Vorlesungsnotizen

Numerik

für ET, ST

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Simon Stingelin

unter Mitwirkung von
Christian Jaeger, Christoph Kirsch, Markus Roos,
Nadin Stahn und Dirk Wilhelm

Version 3. März 2025

Dieses Dokument umfasst in groben Zügen den Inhalt des Numerik Moduls. Das Skript ist nicht als Ersatz für eigene Notizen gedacht und entsprechend auch bei weitem nicht vollständig. Zudem werden im Unterricht weitere Beispiele diskutiert, welche in den vorliegenden Notizen nicht vorhanden sind.

In dem Sinne erheben die Unterlagen auch keinen Anspruch auf Vollständigkeit. Es wird empfohlen zusätzliche Literatur zu studieren. Quellenangaben können im Anhang gefunden werden.

Ein paar Bemerkungen zu mathematischer Software: Es haben sich primär zwei Computer Algebra Programme auf dem Markt etabliert: mathematica und maple. Diese Programme sind sehr hilfreich für das mathematische Verständnis, die Programme sind jedoch nie besser als der Bediener! Ich empfehle Ihnen beide Tools in Ruhe anzuschauen und sich auf eines zu konzentrieren.

Für numerische Rechnungen werden Sie im Studium und auch mit grosser Wahrscheinlichkeit im späteren Arbeitsleben oft python und matlab verwenden. Daher ist es sehr sinnvoll, diese Software schon früh im Studium zu verwenden. Ich empfehle Übungen, in denen etwas numerisch berechnet werden soll, mit python oder matlab zu lösen, um eine gute Praxis in der Anwendung zu erhalten.

Eine interessante Alternative zu matlab ist python, welche sich auch in der Industrie zu etablieren beginnt. Der Vorteil ist, dass python eine OpenSource Software ist und daher keine Kosten verursacht.

Inhaltsverzeichnis

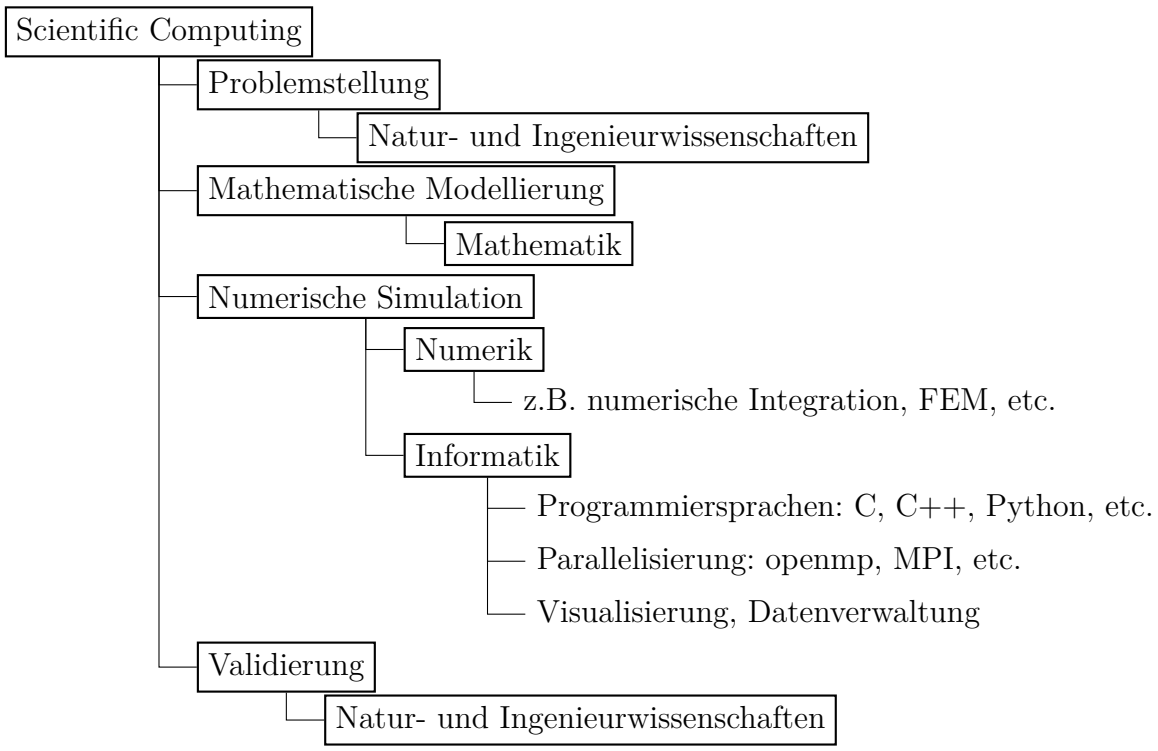
Kapitel 1

Grundlagen der Numerik

1.1 Einleitung

Mit Hilfe der Numerik kann das Verständnis der „realen Welt“ auf virtuellem Weg vertieft und erweitert werden. Dies ist insbesondere in den naturwissenschaftlich/technisch - physikalischen Anwendungsgebieten sehr wichtig, da man in Bereiche vorstossen kann, die experimentell nicht mehr zugänglich sind. Modellrechnungen sind heute integraler Bestandteil in der Entwicklung von Produkten.

Die Numerische Mathematik ist Teil des Wissenschaftlichen Rechnens (Scientific Computing), welches sich aus der Schnittstelle der Bereiche Mathematik, Informatik, Natur- und Ingenieurwissenschaften zusammenstellt.



In der Technik ist man bestrebt, Beobachtungen zu simulieren. Dabei werden verschiedene Fehler gemacht:

Modellfehler: Das mathematische Modell beschreibt in der Regel die Beobachtung nur bis zu einem bestimmten Detaillierungsgrad. Es werden daher idealisierte Rahmenbedingungen geschaffen bzw. beschrieben.

Datenfehler: Modellparameter werden oft aus Messungen bestimmt. Diese wiederum sind aufgrund beschränkter Messgenauigkeit mit Messfehlern behaftet.

Verfahrensfehler: Ein numerisches Lösungsverfahren liefert selbst bei exakter Rechnung häufig nur näherungsweise eine Lösung (Bsp.: Numerische Integration).

Rundungsfehler: Für die Realisierung eines numerischen Verfahrens werden in der Regel Gleitkommazahlen verwendet, welche eine approximative Darstellung einer reellen Zahl darstellt. Es treten daher unvermeidlich Rundungsfehler auf.

Bei einer numerischen Simulation gilt es, diese Fehler zu kennen, kontrollieren und minimieren.

1.2 Einführendes Beispiel

[Jupyter-Notebook] Mit Hilfe eines Pendels soll ein Taktmechanismus zu einer vorgegebenen Taktzeit $T^* > 0$ realisiert werden. Zu bestimmen ist daher die erforderliche Anfangsauslenkung zu einer gegebenen Schwingungsdauer T^*. Die Schwingungsdauer T bezeichnet die Zeit, welches das Pendel benötigt um wieder in die Ausgangslage zurück zu schwingen.

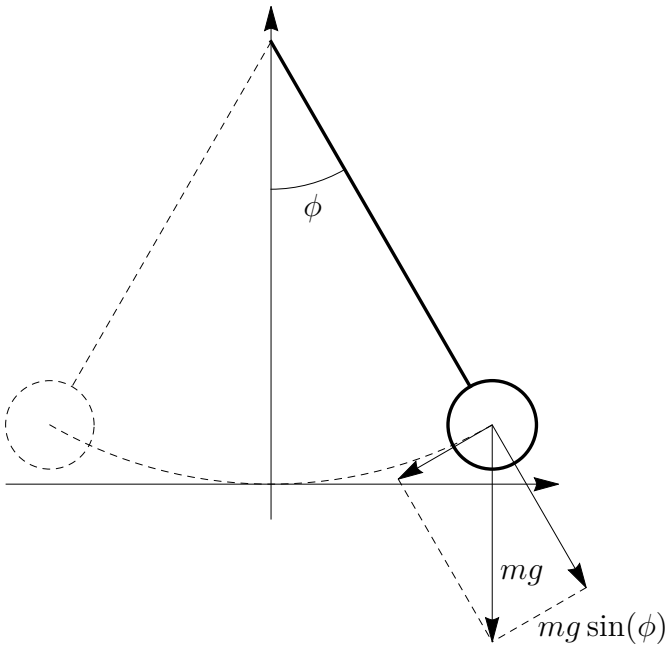


Abbildung 1.1: Mathematisches Pendel

Für die mathematische Modellierung der Aufgabe verwenden wir das Newton'sche Gesetzt:

$$\text{Kraft} = \text{Masse} \times \text{Beschleunigung}.$$

Es gilt daher

$$-m \cdot g \cdot \sin(\phi(t)) = m \cdot l \cdot \ddot{\phi}(t).$$

Mit Hilfe des Anfangswertproblems

$$\begin{aligned} \ddot{\phi}(t) &= -\frac{g}{l} \cdot \sin(\phi(t)) \\ \phi(0) &= \phi_0 \quad \text{und} \quad \dot{\phi}(0) = 0 \end{aligned} \tag{1.1}$$

können wir die Bewegung des Pendels mathematisch modellieren und beschreiben. Die Gravitationskonstante $g \approx 9.81\text{m/s}^2$ und die Länge $l \approx 0.6\text{m}$ sind mit Datenfehler behaftet. Beide Grössen können nicht absolut exakt erfasst werden.

Die Differentialgleichung kann nicht geschlossen gelöst werden. Daher werden numerische Methoden für die Berechnung von $\phi(t)$ benötigt. Für kleine Auslenkungen kann die Näherung

$$\sin(\phi) \approx \phi$$

benutzt werden. Das vereinfachte Modell lautet in dem Fall

$$\begin{aligned} \ddot{\phi}(t) &= -\frac{g}{l} \cdot \phi(t) \\ \phi(0) &= \phi_0 \quad \text{und} \quad \dot{\phi}(0) = 0. \end{aligned} \tag{1.2}$$

Damit wird jedoch ein weiterer systematischer Fehler eingeführt, ein unter Umständen sehr wesentlicher Modellfehler: Die Lösung des vereinfachten Modells (1.2) ist gegeben durch

$$\phi(t) = \phi_0 \cos\left(\sqrt{\frac{g}{l}} \cdot t\right).$$

Offensichtlich scheitert hier das Vorhaben die Taktzeit über die Anfangsauslenkung einzustellen. Die Taktzeit ist in dem Fall unabhängig von der Anfangsauslenkung!

Wir bezeichnen mit $\phi(t, \phi_0)$ die Lösung des Anfangswertproblems (1.1) zur Anfangsauslenkung ϕ_0 und mit $T(\phi_0)$ die Periodizität $T > 0$:

$$\phi(t, \phi_0) = \phi(t + T, \phi_0).$$

Gesucht ist nun $\phi^* > 0$ so, dass

$$T(\phi^*) = T^* \tag{1.3}$$

gilt. Diese Gleichung lässt sicher wiederum nicht algebraisch lösen. Es wird daher eine numerische Methode für die Berechnung einer Lösung benötigt. Mit Hilfe der Analysis können wir jedoch die folgenden Fragen beantworten:

1. Existiert überhaupt eine Lösung des Anfangswertproblems (1.1)?
2. Ist diese Lösung eindeutig?
3. Ist die Lösung stetig vom Anfangswert ϕ_0 abhängig?

Diese Fragen können alle mit ja beantwortet werden. Das heisst also, dass die Funktion $T(\phi_0)$ eine stetige Funktion für $0 < \phi_0 < \pi$ ist. Man kann sogar zeigen, dass die Funktion monoton steigend ist. Die Gleichung (1.3) kann daher zum Beispiel mit dem Bisektions-Verfahren numerisch gelöst werden. Das Verfahren liefert jedoch nur eine numerische Approximation, zum einen will man in endlicher Zeit eine Lösung berechnen und definiert entsprechend einen maximalen Fehler als Abbruchkriterium und zum Andern kann aufgrund der beschränkten Auflösung von Gleitkommazahlen in der Regel nur eine beschränkte Genauigkeit erreicht werden.

1.3 Fehleranalyse

Beschränken wir uns auf die reine Numerik, so setzt sich der Fehler im Resultat von

- Datenfehler und
- Fehler im Algorithmus

zusammen. Datenfehler können in der Regel nicht in der Numerik minimiert werden. Hingegen Fehler im Algorithmus können vermieden bzw. verringert werden, in dem wir geeignete numerische Algorithmen benutzen.

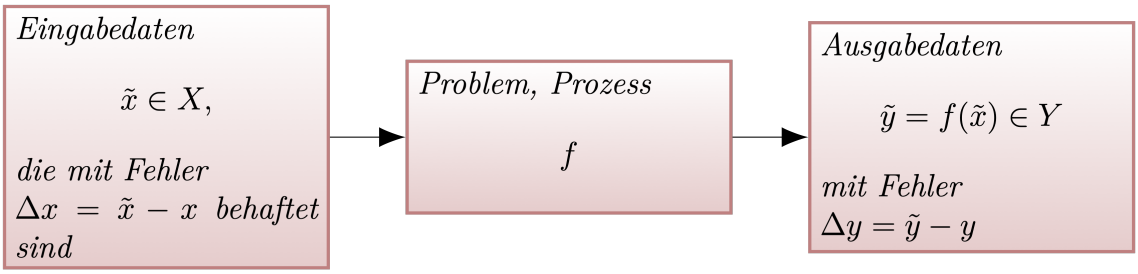
1.3.1 Kondition eines Problems

Die mathematische Analyse der Fehlerverstärkung bei Datenfehlern beruht auf dem Konzept der Kondition eines Problems. Dies ist zunächst unabhängig von einem speziellen Lösungsweg (Algorithmus) und gibt nur an, welche Genauigkeit man bestenfalls (bei exakter Rechnung) bei gestörten Eingangsdaten erwarten kann.

Um dies beschreiben zu können, bezeichnen wir das zu lösende Problem als eine gegeben Funktion

$$\begin{aligned} f : X &\to Y \\ x &\mapsto y = f(x) \end{aligned}$$

Schematisch hat man den folgenden Rahmen:



Es geht nun darum, den Ausgabefehler Δy ins Verhältnis zum Eingabefehler Δx zu setzen.

Definition 1.1 (absoluter, relativer Fehler). Mit dem *absoluten Fehler* seien die Grössen

$$||\Delta x||_X, \quad ||\Delta y||_Y$$

und dem *relativen Fehler*

$$\delta_x = \frac{||\Delta x||_X}{||x||_X}, \quad \delta_y = \frac{||\Delta y||_Y}{||y||_Y}$$

bezeichnet.

Definition 1.2 (relative, absolute Kondition). Mit der *relativen / absoluten Kondition* eines durch f beschriebenen Problems bezeichnet man das Verhältnis

$$\frac{\delta_y}{\delta_x} \quad \text{bzw.} \quad \frac{||\Delta y||_Y}{||\Delta x||_X}$$

des Ausgabefehlers zum Eingabefehler - also die Sensitivität des Problems unter Störung der Eingabedaten.

Bemerkung 1.1. In der Regel wird die relative Kondition verwendet. Ein Problem ist umso besser konditioniert, je kleinere Schranken für δ_y/δ_x mit $\delta_x \to 0$ existieren. ⌟

Beispiel 1.1. Wir betrachten eine reelle, stetig differenzierbare Funktion $f : I \to \mathbb{R}$. Mit der Taylorreihe folgt

$$f(\tilde{x}) = f(x_0) + f'(x_0)(\tilde{x} - x_0) + \mathcal{O}(|\tilde{x} - x_0|^2) \quad \text{für } \tilde{x} \to x_0.$$

Für den relativen Fehler folgt unter Vernachlässigung der Terme höherer Ordnung und insbesondere $|\tilde{x} - x_0|$ klein

$$\underbrace{\frac{f(\tilde{x}) - f(x_0)}{f(x_0)}}_{\text{rel. Fehler der Ausgabe}} \approx \underbrace{f'(x_0) \cdot \frac{x_0}{f(x_0)}}_{\text{Fehlerverstärkung}} \cdot \underbrace{\frac{(\tilde{x} - x_0)}{x_0}}_{\text{rel. Fehler der Eingabe}} . \tag{1.4}$$

Da im Beispiel $X = Y = \mathbb{R}$ gilt, können wir als Norm den Betrag benutzen und erhalten:

Definition 1.3.

$$\left|\frac{f(\tilde{x}) - f(x_0)}{f(x_0)}\right| \leq \kappa_{\text{rel}}(x_0) \cdot \left|\frac{(\tilde{x} - x_0)}{x_0}\right| \quad \text{mit } \kappa_{\text{rel}}(x_0) = \left|f'(x_0) \cdot \frac{x_0}{f(x_0)}\right|. \tag{1.5}$$

Die Zahl $\kappa_{\text{rel}}(\tilde{x})$ heisst *relative Konditionszahl* des Problems f an der Stelle $\tilde{x}$ und beschreibt die maximale Verstärkung des relativen Eingabefehlers.

⌟

Bemerkung 1.2. Das Problem ist umso besser konditioniert, je kleiner $\kappa_{\text{rel}}(x)$ ist.

⌟

Beispiel 1.2. Gegeben sei die Funktion

$$f : \mathbb{R} \to \mathbb{R}, \quad f(x) = e^{3x^2}.$$

Für die relavtive Konditionszahl folgt

$$\kappa_{\text{rel}}(x) = 6x^2.$$

Daraus folgt, dass für kleine $|x|$ die Funktion gut konditioniert ist und entsprechend umgekehrt für $|x|$ gross.

x_0	x	$\delta_x = \frac{x-x_0}{x_0}$	$\delta_y = \frac{f(x)-f(x_0)}{f(x_0)}$	δ_y/δ_x	$\kappa_{\text{rel}}(x_0)$
0.1	0.10001	10^{-4}	$6.00032 \cdot 10^{-6}$	0.0600032	0.06
4	4.0004	10^{-4}	$9.64671 \cdot 10^{-3}$	96.4671	96
0.1	0.101	10^{-2}	$6.03182 \cdot 10^{-3}$	0.0603182	0.06
4	4.04	10^{-2}	1.62426	162.426	96

Ist $|x - x_0|$ zu gross, dann gilt $\delta_y/\delta_x \approx \kappa_{\text{rel}}(x_0)$ nicht mehr.

⌟

Beispiel 1.3 (Integralberechnung über Rekursion). Es soll

$$\int_0^1 x^{30} e^x dx$$

berechnet werden. Mit Hilfe der partiellen Integration erhält man die Rekursion

$$\begin{aligned} I_n &= \int_0^1 x^n e^x dx = x^n e^x \Big|_0^1 - n \int_0^1 x^{n-1} e^x dx = e - n \cdot I_{n-1}, \\ I_1 &= 1 \end{aligned} \tag{1.6}$$

Aus der Rekursionsformel folgt

$$\tilde{I}_n - I_n = -n\,(\tilde{I}_{n-1} - I_{n-1}) = n\,(n-1)\,(\tilde{I}_{n-2} - I_{n-2}) = (-1)^{n-1} n!\,(\tilde{I}_1 - I_1)$$

und entsprechend für den relativen Fehler

$$\frac{|f(\tilde{I}_1) - f(I_1)|}{|f(I_1)|} = \frac{|\tilde{I}_{30} - I_{30}|}{|I_{30}|} = \underbrace{\frac{30!\;|I_1|}{|I_{30}|}}_{\kappa_{\text{rel}}} \cdot \frac{|\tilde{I}_1 - I_1|}{|I_1|}.$$

Das Integral I_{30} können wir abschätzen. Es gilt

$$I_{30} = \int_0^1 x^{30} e^x dx \le \int_0^1 x\, e^x dx = I_1.$$

Für die relative Kondition κ_{rel} folgt daher

$$\kappa_{\text{rel}} = \frac{30!\;|I_1|}{|I_{30}|} \ge 30! \approx 2.65253 \cdot 10^{32}$$

Es ist also zu erwarten, dass die numerische Berechnung von I_{30} scheitern wird. Für I_{30} ist das Resultat berechnet mit floating Zahlen (z.B. in C double) unbrauchbar! ⌟

Bemerkung 1.3. Bei der Ermittlung von Konditionszahlen kommt es in der Regel nicht auf den exakten Zahlenwert an, sondern auf die Grössenordnung. ⌟

1.3.2 Zahlendarstellung, Rundungsfehler und Stabilität

Bei der digitalen Darstellung von Zahlen werden Rundungsfehler gemacht. In der Analysis 1 wurde gezeigt, dass jede beliebige reelle Zahl $x \neq 0$ in der Form

$$x = \pm\Big(\sum_{j=1}^{\infty} d_j\, b^{-j}\Big) \cdot b^e \tag{1.7}$$

darstellen lässt, wobei der ganzzahlige Exponent e so gewählt werden kann, dass $d_1 \neq 0$ gilt. Es gilt der Satz aus der Analysis, hier konkret mit $b = 10$ formuliert:

Satz 1.1. *Jede reelle Zahl x kann als Dezimalbruchentwicklung*

$$x = d_0 + d_1 \cdot 10^{-1} + d_2 \cdot 10^{-2} + d_3 \cdot 10^{-3} + \dots$$

mit $d_0 \in \mathbb{Z}$ und $d_1, d_2, d_3, \dots \in \{0,1,2,3,4,5,6,7,8,9\}$ dargestellt werden.
Der Dezimalbruch

$$x = d_0 + d_1 \cdot 10^{-1} + d_2 \cdot 10^{-2} + d_3 \cdot 10^{-3} + \dots + d_n \cdot 10^{-n}$$

approximiert x (ohne Rundung) mit Genauigkeit 10^{-n}.
x ist genau dann in $\mathbb{Q}$, wenn die Dezimalbruchentwicklung **abbricht** *oder die Koeffizienten d_i* **periodisch** *werden.*

Auf einem Rechner lassen sich nur *endlich* viele Zahlen exakt darstellen, da die Summe nur endlich viele Summanden aufweist. Die Menge dieser Zahlen wird auch als *Maschinenzahlen* bezeichnet. Bei numerischen Berechnungen wird sehr oft die *normalisierte Gleitpunktdarstellung* (floating point representation) verwendet:

Definition 1.4.

$$x = f \cdot b^e,$$

wobei

- $b \in \mathbb{N} \setminus \{1\}$ die *Basis* des Zahlensystems ist,
- der *Exponent* e eine ganze Zahl innerhalb gewisser fester Schranken ist:

$$r \leq e \leq R,$$

- die *Mantisse* f eine feste Anzahl m von Stellen hat:

$$f = \pm 0.d_1 \dots d_m, \quad d_j \in \{0, 1, \dots, b-1\} \quad \forall j.$$

Wobei für $x \neq 0$ stets $d_1 \neq 0$ sei (Eindeutigkeit der Darstellung).

Mit dieser Darstellung erhält man

$$x = \pm\Big(\sum_{j=1}^{m} d_j\, b^{-j}\Big) \cdot b^e.$$

Bemerkung 1.4. Es gilt stets

$$b^{-1} \leq |f| < 1.$$

Beispiel 1.4 (Jupyter-Notebook). Für die Zahl 123.75 folgt mit Basis $b = 2$

$$\begin{aligned} 123.75 &= 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= +(1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + 1 \cdot 2^{-7} \\ &\quad + 1 \cdot 2^{-8} + 1 \cdot 2^{-9}) \cdot 2^7 \end{aligned}$$

Definition 1.5 (IEEE-754). (Quelle Wikipedia) IEEE 754 unterscheidet vier Darstellungen: single, single extended, double und double extended. Die genaue Bitzahl und der Biaswert bleiben dem Implementierer überlassen. Die Grundformate sind vollständig definiert.

Typ	Grösse $(1+r+p)$	Exponent (r)	Mantisse (p)	Werte des Exponenten (e)	Biaswert B
single	32 bit	8 bit	23 bit	$-126 \leq e \leq 127$	127
double	64 bit	11 bit	52 bit	$-1022 \leq e \leq 1023$	1023
single ext.	$\geq$ 43 bit	$\geq$ 11 bit	$\geq$ 31 bit		
double ext.	$\geq$ 79 bit	$\geq$ 15 bit	$\geq$ 63 bit		

Die Anordnung der Bits einer single / double zeigt die nachfolgende Tabelle:

S	Exponent E			Mantisse M		
1	1	...	8	1	...	23
9 Bit				23 Bit		
single, 4 Byte						

S	Exponent E			Mantisse M		
1	1	...	11	1	...	52
3× 1/2-Bytes				13× 1/2-Bytes		
double, 8 Byte						

Die Interpretation hängt vom Exponent ab:

Exponent E	Mantisse M	Bedeutung	Bezeichnung
$E = 0$	$M = 0$	$(-1)^S \times 0$	Null
$0 < E < 2^r - 1$	$M \geq 0$	$(-1)^S \times (1 + M/2^p) \times 2^{E-B}$	normalisierte Zahl
$E = 2^r - 1$	$M = 0$	$\pm$ Inf	Unendlich
$E = 2^r - 1$	$M > 0$	NaN	keine Zahl

Bemerkung 1.5. Weitere Details sind z.B. unter https://de.wikipedia.org/wiki/IEEE_754 zu finden.

Bemerkung 1.6. Die Zahl

$$\text{eps} := \frac{b^{1-m}}{2}$$

wird relative *Maschinengenauigkeit* genannt. Dieses Zahl charakterisiert das *Auflösungsvermögen* des Rechners. Es gilt, dass eps gerade die kleinst mögliche auf dem Rechner darstellbare Zahl ist, die zu 1 addiert von der Rundung noch wahrgenommen wird.

Bemerkung 1.7. Die Addition ist im Gegensatz zur Multiplikation und Division problematisch, da die relativen Fehler enorm verstärkt werden können. Diesen Effekt nennt man *Auslöschung*.

Definition 1.6 (Stabilität). Ein Algorithmus heisst *stabil*, wenn die durch ihn im Laufe der Rechnung erzeugten Fehler in der Grössenordnung des durch die Kondition des Problems bedingten unvermeidbaren Fehlers bleiben.

Um die Fehlerakkumulation in komplexeren Situationen abschätzen zu können, bedient man sich häufig des Prinzips der *Rückwärtsanalyse*:

- Interpretiere sämtliche im Laufe der Rechnung auftretenden Fehler als Ergebnis *exakter* Rechnung zu geeignet gestörten Daten.
- Abschätzungen für diese Störung der Daten, verbunden mit Abschätzungen für die Kondition des Problems, ergeben dann Abschätzungen für den Gesamtfehler.

Bemerkung 1.8. Zusammenfassend lassen sich als Leitlinien einige einfache Grundregeln formulieren:

- Kenntnisse über die Kondition eines Problems sind oft für die Interpretation oder Bewertung der Ergebnisse von entscheidender Bedeutung.
- Multiplikation und Division sind Operationen die für alle Eingangsdaten gut konditioniert sind. Die Subtraktion zweier annähernd gleicher Zahlen ist eine Operation die schlecht konditioniert ist. Dadurch können bei einer solchen Subtraktion Rundungsfehler enorm verstärkt werden. Diesen Effekt nennt man Auslöschung.
- Beim Addieren sollte man die Summanden in der Reihenfolge aufsteigender Beträge addieren. Dadurch erreicht man bei gleicher Rechenzeit ein wesentlich genaueres Ergebnis.
- In einem Algorithmus sollen Auslöschungseffekte möglichst vermieden werden.
- Bei einem stabilen Lösungsverfahren bleiben die im Laufe der Rechnung erzeugten Rundungsfehler in der Größenordnung der durch die Kondition des Problems bedingten unvermeidbaren Fehler.

Kapitel 2

Gleichungssysteme

2.1 Lineare Gleichungen, direkte Lösungsverfahren

2.1.1 Einleitung

Lineare Gleichungssysteme treten in sehr vielen Anwendungen auf und können in den Anforderungen an den Lösungsalgorithmus sehr unterschiedlich sein.

Beispiel 2.1. In der Elektronik können elektrische Netzwerke mit Hilfe der Kirchhoffschen Sätze mathematisch beschrieben werden. Der erste Satz besagt, dass der Gesamtstrom in einem Knoten verschwinden muss:

$$\sum_i I_i = 0.$$

Der zweite Kirchhoffsche Satz besagt, dass die Summe der Spannungen U_j in einer geschlossenen Masche Null sein muss:

$$\sum_j U_j = 0.$$

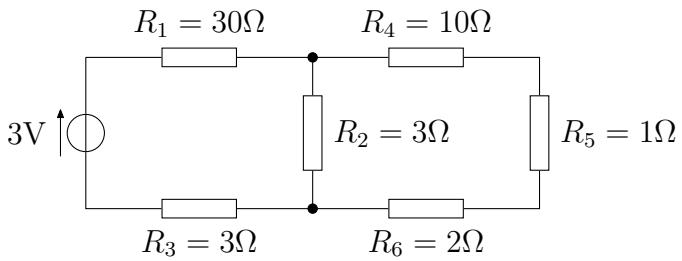


Abbildung 2.1: Elektrisches Netzwerk

In diesem Kapitel werden Lösungsmethoden für lineare Gleichungssysteme behandelt. Die allgemeine Aufgabe lautet:

Definition 2.1 (Lineares Gleichungssystem). Das Problem:
Bestimme $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ zu $A \in \mathbb{R}^{n\times n}$ und $\mathbf{b} = (b_1, \ldots, b_n)^T \in \mathbb{R}^n$ so, dass

$$\begin{array}{ccccc} a_{1,1}\,x_1 & +\ldots+ & a_{1,n}\,x_n & = & b_1 \\ \vdots & & \vdots & & \vdots \\ a_{n,1}\,x_1 & +\ldots+ & a_{n,n}\,x_n & = & b_n \end{array}$$

bzw.

$$A \cdot \mathbf{x} = \mathbf{b} \tag{2.1}$$

gilt. Wird *lineares Gleichungssystem* genannt.

Das Beispiel 2.1 hat uns ein Gleichungssystem von dieser Art geliefert. Für lineare Gleichungssysteme gilt der Satz (vgl. lineare Algebra Vorlesung):

Satz 2.1. *Folgende Aussagen sind äquivalent:*

1. *Das System 2.1 hat für jedes* $\mathbf{b} \in \mathbb{R}^n$ *eine eindeutige Lösung* $\mathbf{x} \in \mathbb{R}^n$.
2. *Die Matrix A in 2.1 hat vollen Rang n.*
3. *Für A in 2.1 hat das* homogene *System* $A \cdot \mathbf{x} = 0$ *nur die triviale Lösung* $\mathbf{x} = 0$.
4. *Es gilt*
$$\det A \neq 0.$$

A heisst regulär, *wenn* $\det A \neq 0$ *und damit die Eigenschaften gelten.*

Wie man es nicht machen sollte

Eine prinzipielle Möglichkeit (2.1) zu lösen, bietet die *Cramersche Regel*. Danach lautet die j-te Komponente x_j der Lösung von (2.1)

$$x_j = \frac{\det A_j}{det A},$$

wobei $A_j \in \mathbb{R}^{n\times n}$ diejenige Matrix ist, die aus A entsteht, wenn die j-te Spalte durch b ersetzt wird. Für das Lösen des Gleichungssystems müssen $n+1$ Determinanten

$$\det A, \det A_1, \ldots, \det A_n$$

berechnet werden.

Die Vorgehensweise ist jedoch numerisch extrem aufwändig und daher im allgemeinen nicht geeignet. Es ergäben sich unter der Verwendung des Laplaceschen Entwicklungssatzes für die Berechnung der Determinanten eine Berechnungszeit (100Mflops) von

n	10	12	14	16	18	20
Rechenzeit	0.4s	1 min	3.6h	41 Tage	38 Jahre	16000 Jahre

Tabelle 2.1: Rechenzeit für Cramersche Regel

Ziel der folgenden Abschnitte ist die Vorstellung ausgewählter Lösungsverfahren. Der Grund für verschiedene Verfahren liegt in der unterschiedlichen Eignung für unterschiedliche Problemmerkmale. Wichtige Merkmale sind Effizienz (Rechenaufwand, Speicherbedarf) und Stabilität.

Die Verfahren beruhen auf der folgenden Beobachtung: Falls C irgend eine reguläre Matrix ist, gilt

$$A x = b \quad \Leftrightarrow \quad C\,A\,x = C\,b. \tag{2.2}$$

In der Tat bedeutet $C\,A\,x = C\,b$ gerade $C\,(A\,x - b) = 0$. Da C regulär ist, ist dies äquivalent zu $A\,x - b = 0$. Die Strategie wird also sein, mit Hilfe von regulären Matrizen das gegebene Gleichungssystem in ein leicht lösbares äquivalentes System umzuformen.

2.1.2 Konditionszahlen linearer Abbildungen

Die Lösung eines linearen Gleichungssystems kann ein schlecht konditioniertes Problem darstellen.

Beispiel 2.2. Das Gleichungssystem

$$\begin{pmatrix} 3.3 & 1.2 \\ 6.9 & 2.5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.1 \\ 2.7 \end{pmatrix}$$

besitzt eine exakte Lösung, welche

$$x \approx \begin{pmatrix} 16.3 \\ -44.0 \end{pmatrix}$$

erfüllt. Das geringfügig verschiedene Gleichungssystem

$$\begin{pmatrix} 3.31 & 1.2 \\ 6.9 & 2.5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.1 \\ 2.7 \end{pmatrix}$$

besitzt eine exakte Lösung, welche

$$x \approx \begin{pmatrix} 98.0 \\ -269.4 \end{pmatrix}$$

erfüllt. Eine minimale Änderung des Eingangsdatums führt zu einer dramatischen Änderung der Ausgangsdaten. Dies zeigt, dass die Numerik linearer Gleichungssysteme bei weitem nicht trivial ist. ⌟

Zur Erinnerung aus der linearen Algebra:

Definition 2.2 (Lineare Abbildung). Eine Abbildung $\mathcal{L} : X \to Y$ heisst *linear*, falls für $x, y \in X$ und $\alpha, \beta \in \mathbb{K}$

$$\mathcal{L}(\alpha x + \beta y) = \alpha \mathcal{L}(x) + \beta \mathcal{L}(y), \quad \text{und} \quad \mathcal{L}(x) = 0 \;\Leftrightarrow\; x = 0$$

gilt.

Bemerkung 2.1. Die Matrix $A \in \mathbb{R}^{n\times n}$ beschreibt eine lineare Abbildung

$$\begin{aligned}\mathcal{L} : \mathbb{R}^n &\to \mathbb{R}^n \\ \mathbf{x} &\mapsto \mathbf{y} = A \cdot \mathbf{x}.\end{aligned}$$

Beweis als Aufgabe. ⌟

Bemerkung 2.2 (Absolute Kondition). Die Menge der linearen Abbildungen bildet bezüglich der üblichen additiven Verknüpfung wieder einen $\mathbb{K}$-Vektorraum. Diesen kann man wieder mit einer Norm ausstatten.

Legt man eine Norm $\|\cdot\|_X$ für X und eine Norm $\|\cdot\|_Y$ für Y fest, so gibt die *Operatornorm* von $\mathcal{L}$ an, wie die Abbildung $\mathcal{L}$ die Einheitskugel $K_{\|\cdot\|_X}(0,1)$ verformt.

Definition 2.3. Sei

$$\|\mathcal{L}\|_{X\to Y} := \sup_{x\in K_{\|\cdot\|_X}(0,1)} \|\mathcal{L}(x)\|_Y = \sup_{\|x\|_X=1} \|\mathcal{L}(x)\|_Y \tag{2.3}$$

die Operatornorm von $\mathcal{L}$.

Für beliebiges $x \in X$ gilt

$$\tilde{x} = \frac{x}{\|x\|_X} \in K_{\|\cdot\|_X}(0,1).$$

Also ist

$$\|\mathcal{L}\|_{X\to Y} := \sup_{x\neq 0} \frac{\|\mathcal{L}(x)\|_Y}{\|x\|_X}$$

zu (2.3) äquivalent. Daraus folgt die Eigenschaft

$$\|\mathcal{L}(x)\|_Y \leq \|\mathcal{L}\|_{X\to Y}\|x\|_X \quad \forall\, x \in X.$$

Man sagt, $\mathcal{L}$ ist *beschränkt*, wenn $\|\mathcal{L}\|_{X\to Y}$ endlich ist. Es gilt

Satz 2.2. *Eine lineare Abbildung ist beschränkt genau dann, wenn sie stetig ist.*

Stetig heisst: Zu jedem $x \in X$ und $\epsilon > 0$ gibt es ein $\delta = \delta(x,\epsilon) > 0$ so, dass $\|\mathcal{L}(x) - \mathcal{L}(x')\|_Y \leq \epsilon$ für alle $\|x - x'\|_X \leq \delta$. Wegen der Linearität gilt

$$\|\mathcal{L}(x) - \mathcal{L}(x')\|_Y = \|\mathcal{L}(x - x')\|_y \leq \|\mathcal{L}\|_{X\to Y}\|x - x'\|_X.$$

Daher, wenn $\mathcal{L}$ beschränkt ist, ist $\mathcal{L}$ sogar Lipschitz-stetig mit der Lipschitz-Konstante $\|\mathcal{L}\|_{X\to Y}$. Die *absolute* Kondition einer linearen Abbildung ist somit gerade durch die entsprechende Operatornorm $\|\mathcal{L}\|_{X\to Y}$ gegeben. ⌟

Bemerkung 2.3. Sei speziell $X = \mathbb{R}^n$, $Y = \mathbb{R}^m$ und $B \in \mathbb{R}^{m\times n}$ eine $(m \times n)$-Matrix. Stattet man sowohl X als auch Y mit der p-Norm für $1 \leq p \leq \infty$ aus, bezeichnet man die entsprechende Operatornorm kurz als $\|B\|_p := \|B\|_{X\to Y}$.

Es gilt

$$\|B\|_\infty = \max_{i=1,\ldots,m} \sum_{k=1}^{n} |b_{i,k}|, \quad \text{(maximale absolute Zeilensumme)} \tag{2.4}$$

sowie

$$\|B\|_1 = \max_{k=1,\dots,n} \sum_{i=1}^{m} |b_{i,k}| \quad \text{(maximale absolute Spaltensumme)}. \tag{2.5}$$

Für $A \in \mathbb{R}^{n\times n}$ gilt

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

wobei A^T die Transponierte von A ist und $\lambda_{\max}$ der grösste Eigenwert. ⌟

Die relative Kondition linearer Abbildungen verlangt die Abschätzung von

$$\frac{\|\mathcal{L}(\tilde{x}) - \mathcal{L}(x)\|_Y}{\|\mathcal{L}(x)\|_X} \quad \text{durch} \quad \frac{\|\tilde{x} - x\|_X}{\|x\|_X}.$$

Satz 2.3 (Relative Kondition). *Sei $\mathcal{L}$ eine bijektiv lineare Abbildung, dann gilt*

$$\frac{\|\mathcal{L}(\tilde{x}) - \mathcal{L}(x)\|_Y}{\|\mathcal{L}(x)\|_Y} \le \kappa(\mathcal{L}) \frac{\|\tilde{x} - x\|_X}{\|x\|_X},$$

wobei

$$\kappa(\mathcal{L}) = \|\mathcal{L}\|_{X\to Y} \|\mathcal{L}^{-1}\|_{Y\to X}.$$

Wir betrachten nun den Spezialfall, dass die lineare Abbildung $\mathcal{L}$ gegeben ist durch

$$\begin{aligned} \mathcal{L} : \mathbb{R}^n &\to \mathbb{R}^n \\ \mathbf{x} &\mapsto \mathbf{y} = A \cdot \mathbf{x}, \end{aligned}$$

mit $A \in \mathbb{R}^{n\times n}$. In dem Fall gilt der Satz:

Satz 2.4. *Sei $x + \Delta x$ die Lösung von $A \cdot (x + \Delta x) = b + \Delta b$. Dann gilt*

$$\frac{|\Delta x|}{|x|} \le \|A^{-1}\| \cdot \|A\| \cdot \frac{|\Delta b|}{|b|}.$$

Beweis. Der Satz folgt direkt aus dem Satz 2.3 mit $\mathcal{L} = A^{-1}$.

Hier im konkreten Kontext die Beweisskizze: Es gilt

$$A \cdot (x + \Delta x) = b + \Delta b \quad \Leftrightarrow \quad A\,\Delta x = \Delta b \quad \Leftrightarrow \quad \Delta x = A^{-1}\Delta b,$$

und somit

$$|\Delta x| = |A^{-1}\Delta b| \le \|A^{-1}\| \cdot |\Delta b|.$$

Andererseits gilt

$$|b| = |Ax| \le \|A\| \cdot |x|$$

und daher

$$\frac{1}{|x|} \le \frac{\|A\|}{|b|}.$$

Aus den beiden Ungleichungen folgt

$$\frac{|\Delta x|}{|x|} \le \|A^{-1}\| \cdot |\Delta b| \cdot \frac{1}{|x|} \le \underbrace{\|A^{-1}\| \cdot \|A\|}_{=:\kappa(A)} \cdot \frac{|\Delta b|}{|b|}.$$

□

Definition 2.4. Die relative Konditionszahl bezüglich der Operatornorm $\|\cdot\|$ ist definiert durch

$$\kappa(A) := \|A^{-1}\| \cdot \|A\|.$$

Bemerkung 2.4. Die Konditionszahl hat folgende Eigenschaften:

1. Die Konditionszahlen von Matrizen hängen von der jeweiligen Norm $\|\cdot\|$ ab.
2. Es gilt stets
$$1 \leq \kappa(A).$$
3. $\kappa(A) = \kappa(A^{-1})$

⌟

Bemerkung 2.5. Aufgrund dessen, dass die Inverse A^{-1} einer Matrix A benötigt wird, ist es im Allgemeinen sehr schwierig und aufwändig, die Konditionszahl $\kappa(A)$ zu berechnen. Ist A schlecht konditioniert, so kann auch die Inverse nicht exakt berechnet werden, womit auch die Konditionszahl nicht berechnet werden kann. Es existieren Methoden, die im Laufe der Lösung eines linearen Gleichungssystems eine Schätzung der Konditionszahl liefern.

⌟

Um die Genauigkeit einer Annäherung $\tilde{x}$ der Lösung eines Gleichungssystems $Ax = b$ zu messen, wird oft als Mass die Grösse des

Definition 2.5. Residuums

$$\tilde{r} = b - A \cdot \tilde{x}$$

verwendet, welches man ohne Kenntnis der exakten Lösung x berechnen kann. Wie aussagekräftig die Grösse des Residuums in Bezug auf den tatsächlichen Fehler ist, hängt wieder von der Kondition ab

$$\kappa(A)^{-1}\frac{|\tilde{r}|}{|b|} \leq \frac{|x - \tilde{x}|}{|x|} \leq \kappa(A)\frac{|\tilde{r}|}{|b|}$$

und kann daher ein schlechtes Mass für den Fehler sein.

Beispiel 2.3. Sei

$$A = \begin{pmatrix} 3 & 1.001 \\ 6 & 1.997 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 6 \end{pmatrix}.$$

Die exakte Lösung des Gleichungssystems $Ax = b$ ist

$$x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Für die Annäherungen

$$\tilde{x} = \begin{pmatrix} 0.99684 \\ 0.00949 \end{pmatrix} \quad \text{und} \quad \hat{x} = \begin{pmatrix} 1.000045 \\ 0.000089 \end{pmatrix},$$

gilt

$$\|\tilde{r}\|_\infty = \|b - A\tilde{x}\|_\infty = 1.95 \cdot 10^{-5}$$
$$\|\hat{r}\|_\infty = \|b - A\hat{x}\|_\infty = 4.48 \cdot 10^{-4}$$

Die Norm des Residuums ist daher für $\tilde{x}$ viel kleiner als für $\hat{x}$. Der Fehler ist jedoch viel grösser:

$$\|x - \tilde{x}\|_\infty = 9.49 \cdot 10^{-3} \gg \|x - \hat{x}\|_\infty = 8.90 \cdot 10^{-5}$$

⌟

2.1.3 Dreiecksmatrizen, Rückwärtseinsetzen

Sogenannte Dreiecksmatrizen ergeben leichter lösbare Systeme.

Definition 2.6 (Dreiecksmatrix). Eine Matrix $R = (r_{i,j})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ heisst *obere* Dreiecksmatrix, falls

$$r_{i,j} = 0 \quad \text{für } i > j,$$

untere Dreiecksmatrix $L = (l_{i,j})_{i,j=1}^n \in \mathbb{R}^{n\times n}$, falls

$$l_{i,j} = 0 \quad \text{für } i < j.$$

Dreiecksmatrizen deren Diagonaleinträge eins sind, heissen *normiert*.

Definition 2.7. Eine Matrix $D = (d_{i,j})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ heisst Diagonalmatrix, falls

$$d_{i,j} = 0 \quad \text{für } i \neq j$$

gilt. Schreibweise

$$D = \operatorname{diag}(d_{1,1}, \ldots, d_{n,n}).$$

Wir betrachten das System

$$\begin{aligned}
r_{1,1}x_1 + r_{1,2}x_2 + \quad &\ldots \quad + r_{1,n}x_n = b_1 \\
r_{2,2}x_2 + \quad &\ldots \quad + r_{2,n}x_n = b_2 \\
&\vdots \\
r_{n-1,n-1}x_{n-1} + r_{n-1,n}x_n &= b_{n-1} \\
r_{n,n}x_n &= b_n
\end{aligned}$$

Das System hat genau dann eine eindeutige Lösung (vgl. Satz 2.1), wenn

$$\det R = r_{1,1} \cdot r_{2,2} \cdot \ldots \cdot r_{n,n} \neq 0,$$

gilt, daher alle Diagonaleinträge ungleich Null sind.

Das System kann durch Rückwärtseinsetzen sukzessive gelöst werden

$$x_n = \frac{b_n}{r_{n,n}}.$$

Einsetzen von x_n in die zweitletzte Gleichung ergibt

$$x_{n-1} = \frac{b_{n-1} - r_{n-1,n}\, x_n}{r_{n-1,n-1}}.$$

Allgemein können wir festhalten

Algorithmus 2.1 (Rückwärtseinsetzen). `Für` $j = n, n-1, \ldots, 1$ `berechne`

$$x_j = \frac{b_j - \sum_{k=j+1}^{n} r_{j,k}\, x_k}{r_{j,j}}, \tag{2.6}$$

`wobei obige Summe für` $j = n$ `als Null interpretiert wird.`

Bemerkung 2.6. Der Rechenaufwand für das Rückwärtseinsetzen ist gegeben durch: Für jedes $j = n-1, \ldots, 1$:

- $n - j$ Multiplikationen / Additionen
- eine Division

und eine Division für $j = n$. Daher

$$\underbrace{\left(\sum_{j=1}^{n-1}(n-j)\right)}_{\text{Multiplikationen / Additionen}} + \underbrace{n}_{\textit{Divisionen}} = \frac{n(n-1)}{2} + n = \frac{1}{2}(n^2+n)$$

Der Rechenaufwand für das Rückwärtseinsetzen beträgt also ca. $\frac{1}{2}n^2$ Operationen.

⌟

Gefragt sind daher Methoden um ein das System

$$A\, x = b, \quad \det A \neq 0$$

auf Dreiecksgestalt zu bringen.

2.1.4 Gauss-Elimination, LU-Zerlegung

Gauss-Elimination ohne Pivotisierung

Die bekannteste Methode das System

$$A\, x = b, \quad \det A \neq 0$$

auf Dreiecksgestalt zu bringen, ist die *Gauss-Elimination*. Man verändert die Lösung nicht, wenn man Vielfache einer der Gleichungen von den anderen subtrahiert. Daraus resultiert das folgenden Vorgehen:

- Sei $a_{1,1} \neq 0$.
- Subtrahiere geeignete Vielfache der 1. Gleichung von den überigen.

$$\overset{A=A^{(1)}}{\begin{pmatrix} \circledast & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & & \vdots \\ * & * & \dots & * \end{pmatrix}} \rightarrow \overset{A^{(2)}}{\left(\begin{array}{c|ccc} * & * & \dots & * \\ \hline 0 & * & \dots & * \\ \vdots & \vdots & \tilde{A}^{(2)} & \vdots \\ 0 & * & \dots & * \end{array}\right)}$$

- Wende das Vorgehen rekursiv auf $\tilde{A}^{(k)}$ an, bis $k = n$.

$$\overset{A^{(2)}}{\left(\begin{array}{c|ccc} * & * & * & \dots & * \\ \hline 0 & \circledast & * & \dots & * \\ 0 & * & * & \dots & * \\ \vdots & \vdots & \vdots & \tilde{A}^{(2)} & \vdots \\ 0 & * & * & \dots & * \end{array}\right)} \rightarrow \overset{A^{(3)}}{\left(\begin{array}{cc|ccc} * & * & * & \dots & * \\ 0 & * & * & \dots & * \\ \hline 0 & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & \tilde{A}^{(3)} & \vdots \\ 0 & 0 & * & \dots & * \end{array}\right)}$$

Die Einträge der Matrix $A^{(k)}$ werden mit $a_{i,j}^{(k)}$ bezeichnet. Der Eintrag $a_{k,k}^{(k)}$, markiert mit $\circledast$, heisst *Pivotelement*. In analoger Weise ist natürlich auch die rechte Seite b umzuformen. Es ist zweckmässig, nicht das lineare Gleichungssystem aufzuschreiben, sondern nur die Matrix A und die Spalte b, die zu einer Matrix $(A\,b)$ zusammengefasst werden. Es ergibt sich der folgende Algorithmus:

Algorithmus 2.2 (Gauss-Elimination).

Gegeben $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, wobei $\det A \neq 0$.
`Für` $j = 1, 2, \dots, n-1$
  `und falls` $a_{j,j}^{(j)} \neq 0$ `ist`
  `für` $i = j+1, j+2, \dots, n$
    `subtrahiere in` $(A\,b)$ `Zeile` j `mit`
    `Faktor` $\ell_{i,j} := \dfrac{a_{i,j}^{(j)}}{a_{j,j}^{(j)}}$ `von Zeile` i.

Definition 2.8. Der Prozess

- Bestimme $(Ab) \rightarrow (Rc)$ gemäss Algorithmus 2.2
- Löse $R\,x = c$ durch Rückwärtseinsetzen, Algorithmus 2.1

heisst *Gauss-Elimination ohne Pivotisierung.*

Es stellt sich abschliessend die Frage, wie die Gauss-Elimination zur Strategie (2.2) passt:

$$A x = b \quad \Leftrightarrow \quad C\,A\,x = C\,b.$$

Wenn der Prozess $A^{(j)} \to A^{(j+1)}$ als Multiplikation des Systems mit einer regulären Matrix interpretiert werden kann, ist der Zusammenhang gezeigt. Mit Hilfe der *Frobenius-Matrix* L_k

$$L_k := \begin{pmatrix} 1 & & & & & \\ & \ddots & & & 0 & \\ & & 1 & & & \\ & & & 1 & & \\ & & & -\ell_{k+1,k} & 1 & \\ & 0 & & \vdots & & \ddots \\ & & & -\ell_{n,k} & 0 & 1 \end{pmatrix}, \tag{2.7}$$

wobei $\ell_{k+1,k}, \ldots, \ell_{n,k} \in \mathbb{R}$ beliebig, kann der Prozess beschrieben werden.

Durch iterative Anwendung der Frobenius Matrix ergibt sich

$$\underbrace{L_{n-1}\, L_{n-2} \ldots L_1\, A}_{=A^{(n)}=:R}\, x = \underbrace{L_{n-1}\, L_{n-2} \ldots L_1\, b}_{=:c}.$$

Bringt man die Frobenius-Matrizen auf die rechte Seite, erhält man eine sehr wichtige Zerlegung der Matrix A:

$$A = (L_{n-1}\, L_{n-2} \ldots L_1)^{-1} R = L_1^{-1}\, L_2^{-1} \ldots L_{n-1}^{-1}\, R =: L\, R.$$

Durch Nachrechnen kann leicht gezeigt werden, dass

$$L_1^{-1}\, L_2^{-1} \ldots L_{n-1}^{-1} = \begin{pmatrix} 1 & & & \\ \ell_{2,1} & \ddots & 0 & \\ \vdots & \ddots & \ddots & \\ \ell_{n,1} & \ldots & \ell_{n,n-1} & 1 \end{pmatrix} \tag{2.8}$$

gilt.

Satz 2.5. *Gilt im Gauss-Alogorithmus stets $a_{j,j}^{(j)} \neq 0$, dann erhält man*

$$A = L\, R,$$

wobei R durch das Resultat R der Gauss-Elimination (Algorithmus 2.2) definiert und L die durch

$$\ell_{i,j} = \frac{a_{i,j}^{(j)}}{a_{j,j}^{(j)}}$$

definierte normierte untere Dreiecksmatrix ist.

Gauss-Elimination mit Pivotisierung

Will man die Matrix $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ mit der Gauss-Elimination zerlegen, scheitert man sofort, da $a_{1,1} = 0$ ist. Es liegt jedoch auf der Hand, dass die beiden Zeilen vertauscht werden können (bedeutet nur eine andere Reihenfolge der Gleichungen). Man erhält in

dem Fall $R = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ und ist damit schon am Ziel. Das Vertauchen von Zeilen ist daher sicherlich notwendig, wenn man auf ein verschwindendes Pivotelement trifft.

Es gibt noch einen weiteren Grund, die Zeilen zu vertauschen. Das folgenden Beispiel zeigt, dass das Vertauschen von Zeilen sehr vorteilhaft sein kann, wenn man *Rundungsfehler* mit einbezieht.

Beispiel 2.4.

$$\begin{pmatrix} 0.00031 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -3 \\ -7 \end{pmatrix}$$

Mit $\ell_{2,1} = 1/0.00031$ ergibt Gauss-Elimination

$$(Rc) = \left(\begin{array}{cc|c} 0.00031 & 1 & -3 \\ 0 & 1 - \frac{1}{0.00031} & -7 - \frac{-3}{0.00031} \end{array} \right).$$

Rückwärtseinsetzen liefert

$$x_2 = \frac{-7 - \frac{-3}{0.00031}}{1 - \frac{1}{0.00031}}, \quad x_1 = \frac{-3 - x_2}{0.00031}$$

Bei 4-stelliger Rechnung mit Abrunden folgt

$$x_2 = \frac{9670}{-3225} \approx -2.998 \quad x_1 = \frac{-3 + 2.998}{0.00031} \approx -6.45$$

Exakte Rechnung ergibt jedoch

$$x_1 = -4.00124\ldots, \quad x_2 = -2.99875\ldots.$$

Betrachtet man die Kondition des Problems

$$\kappa(A) = 4.00$$

so ist das Resultat nicht befriedigend und hat offensichtlich nicht mit dem Problem selber, sondern mit der Lösungsmethode zu tun!

Werden die Gleichungen in der Reihenfolge vertauscht, so erhält man

$$(Rc) = \left(\begin{array}{cc|c} 1 & 1 & -7 \\ 0 & 0.9996 & -2.998 \end{array} \right)$$

und bei 4-stelliger Rechnung mit Abrunden folgt

$$x_1 = -4.000, \quad x_2 = -3.000.$$

Damit akzeptable Werte. ⌟

Bemerkung 2.7. Obwohl $a_{1,1} \neq 0$ gilt, ist hier offensichtlich eine Vertauschung von Zeilen vorteilhaft. ⌟

LU-Zerlegung

Das Vertauschen von Zeilen kann ebenfalls mit Hilfe einer Multiplitkation einer Matrix beschrieben werden. Für das Beispiel 2.4 erhalten wir

$$\underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}_{=P_{1,2}} \cdot \begin{pmatrix} 0.00031 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0.00031 & 1 \end{pmatrix}$$

Die *Permutationsmatrix* $P_{1,2}$ vertauscht die erste und zweite Zeile.

Beispiel 2.5.

$$P_{2,4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

vertauscht die 2. und 4. Zeile. ⌟

Satz 2.6 (LU-Zerlegung). *Zu jeder regulären Matrix A existiert eine Permutationsmatrix P, eine eindeutige untere normierte Dreiecksmatrix L deren Einträge sämtlich betragsmässig durch eins beschränkt sind, und eine eindeutige obere Dreiecksmatrix R, so dass*

$$P\,A = L\,R.$$

Die Matrizen R, L und R ergeben sich aus der Gauss-Elimination mit Spaltenpivotisierung.

Die Implementierung der LU-Zerlegung mit Skalierung und Spaltenpivotisierung kann wie folgt beschrieben werden:

- Bestimme die Diagonalmatrix

  $$D = \mathrm{diag}(d_1, \ \ldots, d_n),$$

  so dass $D\,A$ zeilenweise äquilibriert ist, dh.

  $$d_i = \left(\sum_{k=1}^{n} |a_{i,k}| \right)^{-1}, \quad i = 1, \ldots, n$$

- Wende Gauss-Elimination mit Spaltenpivotisierung auf $D\,A$ an. Im j-ten Schritt der Gauss-Elimination wählt man diejenige Zeile als Pivotzeile, die das betragsmässig grösste Element in der ersten Spalte der $(n+1-j) \times (n+1-j)$ rechten untere Restmatrix hat. Falls diese Pivotzeile und die j-te Zeile verschieden sind, werden sie vertauscht.

Algorithmus 2.3. LU-Zerlegung mit Skalierung und Spaltenpivotisierung

Für $i = 1, 2, \ldots, n$:
  $d_i \leftarrow 1/(\sum_{k=1}^{n} |a_{i,k}|)$;
  Für $j = 1, 2, \ldots, n$:
    $a_{i,j} \leftarrow d_i\, a_{i,j}$; (Skalierung)
Für $j = 1, 2, \ldots, n-1$:
  Bestimme p mit $j \le p \le n$, so dass $|a_{p,j}| = \max_{j \le i \le n} |a_{i,j}|$;
  $r_j = p$;
  Vertausche Zeile j mit Zeile p; (Spaltenpivotisierung)
  Für $i = j+1, \ldots, n$:
    $a_{i,j} \leftarrow \frac{a_{i,j}}{a_{j,j}}$ (neue Einträge in L)
    Für $k = j+1, \ldots n$:
      $a_{i,k} \leftarrow a_{i,k} - a_{i,j} a_{j,k}$: (neue Einträge in R)

Die Werte d_i ergeben die Diagonalmatrix $D = \mathrm{diag}(d_1, \ldots, d_n)$. Die Zahlen r_j entsprechen den elementaren Permutationsmatrizen P_{j,r_j}. Das Produkt aller elementaren Permutationen ist gegeben durch

$$P = P_{n-1,r_{n-1}}\, P_{n-2,r_{n-2}}\, \ldots\, P_{2,r_2}\, P_{1,r_1}.$$

Gemäss Satz 2.6 produziert der Algorithmus 2.3 die Zerlegung

$$P\, D\, A = L\, R.$$

Bemerkung 2.8. Wenn die Betragssummen der Zeilen von A bereits etwa die gleichen Grössenordnungen haben, kann man auf die Skalierung verzichten. ⌟

Bemerkung 2.9. Der Rechenaufwand für die LU-Zerlegung über Gauss-Elimination mit Spaltenpivotisierung beträgt ca.

$$\frac{1}{3}n^3 \quad \text{Operationen.}$$

⌟

Bemerkung 2.10. Merke:

- Skalierung verbessert die *Konditionszahl* der Matrix.
- Pivotisierung verbessert die *Stabilität* der Gauss-Elimination / LU-Zerlegung.

⌟

Anwendungen der LU-Zerlegung

Lösen eines Gleichungssystems Die Lösung von

$$A\, x = b$$

ergibt sich über die Lösung zweier Dreieckssysteme

$$\begin{aligned} A\,x = b \;\Leftrightarrow\; P\,A\,x = P\,b \;&\Leftrightarrow\; L\,\underbrace{R\,x}_{=y} = P\,b \\ &\Leftrightarrow\; L\,y = P\,b \text{ und } R\,x = y \end{aligned} \tag{2.9}$$

Zuerst bestimmt man durch Vorwärtseinsetzen y und danach durch Rückwärtseinsetzen x.

Mehrere rechte Seiten Hat man mehrere Rechteseiten b_j, so benötigt man die LU-Zerlegung mit einem Rechenaufwand von $\sim \frac{1}{3}n^3$ Operationen nur einmal. Das Vorwärts- / Rückwärtseinsetzen benötigt jeweilen nur $\sim n^2$ Operationen.

Berechnen der Inversen Als Anwendungsbeispiel für mehrere rechte Seiten kann die Berechnung der Inversen betrachtet werden: Sei $x_i \in \mathbb{R}^n$ die i-te Spalte der Inversen von A:

$$A^{-1} = (x_1\, x_2\, \ldots x_n).$$

Aus $A\,A^{-1} = \mathbb{1}$ folgt

$$Ax_i = e_i, \quad i = 1, \ldots, n.$$

Berechnung von Determinanten Aus $P\,A = L\,R$ folgt

$$\det P\ \det A = \det L\ \det R = \det R.$$

Für $\det P$ gilt

$$\det P = \det P_{n,r_n} \ldots \det P_{1,r_1} = (-1)^{\#\text{ Zeilenvertauschungen}}$$

und somit

$$\det A = (-1)^{\#\text{ Zeilenvertauschungen}} \cdot \prod_{j=1}^{n} r_{j,j}.$$

Gegenüber dem Laplaceschen Entwicklungssatz ($\sim n!$ Operationen) werden nun nur noch $\sim \frac{1}{3}n^3$ Operationen benötigt.

Nachiteration Aufgrund der Rechnerarithmetik ist es nicht möglich L und R exakt zu berechnen. Sei x_0 der Startwert und $\delta_0 = x - x_0$ der Fehler des Startwerts, wobei mit x die exakte Lösung von $A\,x = b$ bezeichnet sei. Mit $\tilde{L}, \tilde{R}$ sei die Näherung von L, R bezeichnet. Es gilt daher

$$A\,\delta_0 = b - A\,x_0 = r_0. \tag{2.10}$$

Könnte man (2.10) exakt lösen, bekäme man mit $x_0 + \delta_0$ die exakte Lösung. Dies ist nicht möglich. Daher wird mit der berechneten Näherung $\tilde{L}, \tilde{R}$ über (2.9) die Gleichung (2.10) gelöst:

$$\tilde{L}y_0 = r_0, \quad \tilde{R}\delta_0 = y_0.$$

Mit $x_1 = x_0 + \delta_0$ erhält man eine weitere Näherung, die hoffentlich exakter ist.

Algorithmus 2.4 (Nachiteration).
`Sei` r_0 `gegeben.`
`Für` $k = 0, 1, 2, \ldots,$ `berechne:`
$\tilde{L} y_k = r_k \quad \tilde{R}\delta_k = y_k$;
$x_{k+1} = x_k + \delta_k$;
$r_{k+1} = b - A\, x_{k+1}$;

Bemerkung 2.11. Gilt für die näherungsweise LU-Zerlegung $\tilde{L}\tilde{R} = A + \Delta A$, so konvergiert das Verfahren der Nachiteration (bei exakter Rechnung) gegen die Lösung von $A\, x = b$, wenn für irgendeine Norm auf $\mathbb{R}^n$ die Bedingung

$$\|A^{-1}\|\|\Delta A\| < \frac{1}{2}$$

erfüllt ist. Diese Bedingung ist hinreichend, jedoch nicht notwendig. In der Praxis wendet man die Nachiteration - meist mit sehr gutem Erfolg - ohne Überprüfung der Bedingung an. ⌟

Beispiel 2.6 (Finite Differenzen Methode). Wir betrachten das Randwertproblem

$$\begin{aligned} -u''(x) &= 1 \quad \text{für } x \in (0,1) \\ u(0) &= 0 \\ u(1) &= 0. \end{aligned} \tag{2.11}$$

Die analytische Lösung der Differentialgleichung kann einfach durch zweimal integrieren berechnet werden, wir erhalten

$$u(x) = -\frac{1}{2}x^2 + C_1 x + C_2.$$

Aus $u(0) = 0$ folgt $C_2 = 0$ und mit $u(1) = 0$ erhalten wir $C_1 = -1/2$. Damit folgt die analytische Lösung des Randwertproblem (2.11)

$$u(x) = -\frac{1}{2}x(x-1).$$

Ziel des Beispiels ist das Randwertproblem (2.11) numerisch zu berechnen. Dazu benutzen wir die sogenannte Finite Differenzen Methode. Dabei wird der Differenzialquotient durch den Differenzenquotient ersetzt

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

Für die Konvergenz wird vorausgesetzt, dass die Funktion $u(x)$ differenzierbar ist. In dem Fall gilt

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

Sprich: die Ableitung wird diskretisiert.

Um die zweite Ableitung zu erhalten, wird das Vorgehen an den Funktionen $u(x+\Delta)$ und $u(x)$ wiederholt. Es folgt die Diskretisierung der zweiten Ableitung mit Hilfe der finiten Differenzen (bzw. dem Differenzenquotienten)

$$u''(x) \to \frac{\frac{u(x+h)-u(x)}{h} - \frac{u(x)-u(x-h)}{h}}{h} = \frac{u(x+h)-2u(x)+u(x-h)}{h^2} \tag{2.12}$$

Für die numerische Berechnung wird die Funktion $u(x)$ nun diskretisiert. Dazu zerlegt man das Intervall $[0,1]$ in die Teilintervalle

$$[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n],$$

wobei $x_0 = 0 < x_1 < \ldots < x_{n-1} < x_n = 1$ gilt. Wir schreiben

$$u_i = u(x_i) \quad \text{für } i = 0, \ldots, n.$$

Für die Differentialgleichung erhalten wir $n-1$ Gleichungen

$$\begin{aligned} -u_0 + 2u_1 - u_2 &= h^2 \\ &\vdots \\ -u_{n-2} + 2u_{n-1} - u_n &= h^2. \end{aligned} \tag{2.13}$$

Die Diskretisierung der Funktion $u(x)$ führt zu $n+1$ Unbekannten. Das bedeutet zwei Freiheitsgrade müssen Durch die Randbedingung definiert werden, im Beispiel gilt $u_0 = u_n = 0$. Sind die Randbedingungen nicht homogen (daher gleich Null), folgt für das Gleichungssystem (2.13)

$$\begin{aligned} 2u_1 - u_2 &= h^2 + u_0 \\ -u_1 + 2u_2 - u_3 &= h^2 \\ &\vdots \\ -u_{n-3} + 2u_{n-2} - u_{n-1} &= h^2. \\ -u_{n-2} + 2u_{n-1} &= h^2 + u_n. \end{aligned}$$

oder in Matrixschreibweise

$$\begin{pmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & & 0 \\ \vdots & \ddots & & 2 & -1 \\ 0 & \ldots & 0 & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} h^2 + u_0 \\ h^2 \\ \vdots \\ h^2 \\ h^2 + u_n \end{pmatrix} \tag{2.14}$$

Das System kann mit Hilfe der LU-Zerlegung Faktorisiert und durch Vor-/Rückwärtseinsetzen gelöst werden (vgl. Jupyter-Notebook).

⌟

Korrolar 2.1. *Sei $e_h = u(x) - u_h$ der Fehler der Diskretisierten Poisson-Gleichung $-u''(x) = f(x)$ mit Dirichlet Randbedingung. Unter der Voraussetzung $u(x) \in C^4([0,1])$ gilt*

$$\|e_h\|_{2,h} \leq \frac{1}{48} C h^2,$$

wobei

$$C = \|u^{(4)}(x)\|_\infty.$$

Cholesky-Zerlegung

Die LU-Zerlegung kann prinzipiell auf beliebige Gleichungssysteme mit regulären Matrizen angewandt werden. In vielen Anwendungen treten jedoch Matrizen auf, die zusätzliche Struktureigenschaften haben. In vielen Fällen ist die Matrix *positiv definit*.

Definition 2.9. $A \in \mathbb{R}^{n\times n}$ heisst *symmetrisch positiv definit*, falls

$$A^T = A \quad \text{Symmetrie}$$

und

$$x^T A\, x > 0 \quad \text{positiv definit}$$

für alle $x \in \mathbb{R}^n$, $x \neq 0$ gilt.

Beispiel 2.7. Sei $A = B^T B$ mit $B \in \mathbb{R}^{m\times n}$, wobei B vollen Rang hat. Daher die Spalten von B sind linear unabhängig. Es gilt

$$A^T = (B^T B)^T = B^T (B^T)^T = B^T B = A,$$

daher A ist symmetrisch. Sei $x \in \mathbb{R}^n$, dann gilt

$$x^T A\, x = x^T B^T B\, x = (Bx)^T (Bx) = |Bx|^2 \geq 0.$$

Es gilt $x^T Ax = |Bx|^2 = 0$ nur, falls $Bx = 0$ gilt. Da B vollen Rang hat, muss daher $x = 0$ sein. Also gilt $x^T Ax > 0\ \forall x \neq 0$. $A = B^T B$ ist daher positiv definit. ⌟

Satz 2.7. *$A \in \mathbb{R}^{n\times n}$ sei symmetrisch positiv definit. Dann gelten folgende Aussagen:*

1. *A ist invertierbar und A^{-1} ist symmetrisch positiv definit.*
2. *A hat nur strikt positive (reelle) Eigenwerte.*
3. *Jede Hauptuntermatrix von A ist symmetrisch positiv definit.*
4. *Die Determinante von A ist positiv.*
5. *A hat nur strikt positive Diagonaleinträge und der betragsmässig grösste Eintrag von A liegt auf der Diagonalen.*
6. *Bei der Gauss-Elimination ohne Pivotisierung sind alle Pivotelemente strikt positiv.*

Satz 2.8. *Jede symmetrisch positiv definite Matrix $A \in \mathbb{R}^{n\times n}$ besitzt eine eindeutige Zerlegung*

$$A = L\,D\,L^T,$$

wobei L eine normierte untere Dreiecksmatrix und D eine Diagonalmatrix mit Diagonaleinträgen

$$d_{i,i} > 0, \quad i = 1, \ldots, n,$$

ist. Umgekehrt ist jede Matrix der Form $L\,D\,L^T$ mit L, D mit obigen Eigenschaften symmetrisch positiv definit.

Konstruktion der Cholesky-Zerlegung Das Vorgehen sei anhand eines kleinen Beispiels illustriert:

Beispiel 2.8.

$$A = \begin{pmatrix} 2 & 6 & -2 \\ 6 & 21 & 0 \\ -2 & 0 & 16 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ \ell_{2,1} & 1 & 0 \\ \ell_{3,1} & \ell_{3,2} & 1 \end{pmatrix}, \quad D = \begin{pmatrix} d_{1,1} & 0 & 0 \\ 0 & d_{2,2} & 0 \\ 0 & 0 & d_{3,3} \end{pmatrix}$$

Es gilt

$$\begin{aligned} L\,D\,L^T &= \begin{pmatrix} 1 & 0 & 0 \\ \ell_{2,1} & 1 & 0 \\ \ell_{3,1} & \ell_{3,2} & 1 \end{pmatrix} \begin{pmatrix} d_{1,1} & 0 & 0 \\ 0 & d_{2,2} & 0 \\ 0 & 0 & d_{3,3} \end{pmatrix} \begin{pmatrix} 1 & \ell_{2,1} & \ell_{3,1} \\ 0 & 1 & \ell_{3,2} \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} d_{1,1} & 0 & 0 \\ \ell_{2,1}d_{1,1} & d_{2,2} & 0 \\ \ell_{3,1}d_{1,1} & \ell_{3,2}d_{2,2} & d_{3,3} \end{pmatrix} \begin{pmatrix} 1 & \ell_{2,1} & \ell_{3,1} \\ 0 & 1 & \ell_{3,2} \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Die elementweise Auswertung der Gleichung $LDL^T = A$ ergibt

$j = 1$: (1,1)-Element: $d_{1,1} = a_{1,1} = 2$
(2,1)-Element: $\ell_{2,1}d_{1,1} = a_{2,1} = 6 \Rightarrow \ell_{2,1} = 3$
(3,1)-Element: $\ell_{3,1}d_{1,1} = a_{3,1} = -2 \Rightarrow \ell_{3,1} = -1$
$j = 2$: (2,2)-Element: $\ell_{2,1}^2 d_{1,1} + d_{2,2} = a_{2,2} = 21 \Rightarrow d_{2,2} = 3$
(3,2)-Element: $\ell_{3,1}d_{1,1}\ell_{2,1} + \ell_{3,2}d_{2,2} = a_{3,2} = 0 \Rightarrow \ell_{3,2} = 2$
$j = 3$ (3,3)-Element: $\ell_{3,1}^2 d_{1,1} + \ell_{3,2}^2 d_{2,2} + d_{3,3} = a_{3,3} = 16 \Rightarrow d_{3,3} = 2.$

Somit folgt

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -1 & 2 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

⌟

Betrachtet man den *allgemeinen* Fall der Gleichung

$$A = L\,D\,L^T,$$

so folgt für den unteren Dreieckteil

$$a_{i,k} = \sum_{j=1}^{n} \ell_{i,j}d_{j,j}\ell_{k,j} = \sum_{j=1}^{n-1} \ell_{i,j}d_{j,j}\ell_{k,j} + l_{i,k}d_{k,k}, \quad i \geq k,$$

wobei $\ell_{k,j} = 0$ für $j > k$ und $\ell_{k,k} = 1$ gilt. Damit folgt

$$\ell_{i,k}\, d_{k,k} = a_{i,k} - \sum_{j=1}^{k-1} \ell_{i,j} d_{j,j} \ell_{k,j} \quad i \geq k.$$

Für $k = 1$ ist die Summe leer. Da $\ell_{k,k} = 1$ für alle $k = 1, \ldots, n$ gilt, folgt für die erste Spalte $k = 1$:

$$\begin{aligned} \ell_{1,1}\, d_{1,1} = a_{1,1} \quad &\Rightarrow \quad d_{1,1} = a_{1,1} \\ &\qquad\; \ell_{i,1} = \frac{a_{i,1}}{d_{1,1}} \quad \text{für } i > 1. \end{aligned}$$

Hier wurde die Eigenschaft benutzt, dass A nur strikt positive Diagonaleinträge hat. Nun wird vorausgesetzt, dass die Spalten $1, \ldots, k-1$ der Matrix L und D berechnet sind. Es folgt für die k-te Spalte von L und D:

$$\begin{aligned} i = k: \qquad & \ell_{k,k}\, d_{k,k} = a_{k,k} - \sum_{j=1}^{k-1} \ell_{k,j}^2\, d_{j,j} \\ \Rightarrow \quad & d_{k,k} = a_{k,k} - \sum_{j=1}^{k-1} \ell_{k,j}^2\, d_{j,j} \qquad (2.15) \\ i > k: \qquad & \ell_{i,k} = \left(a_{i,k} - \sum_{j=1}^{k-1} \ell_{i,j} d_{j,j} \ell_{k,j} \right) / d_{k,k}, \qquad (2.16) \end{aligned}$$

wobei wiederum die Eigenschaft benutzt wird, dass A nur strikt positive Diagonaleinträge hat. Wir können mit (2.15) und (2.16) das Cholesky-Verfahren wie folgt notieren:

Algorithmus 2.5 (Cholesky-Verfahren).

`Für` $k = 1, 2, \ldots, n$`:`

  `diag` $\leftarrow a_{k,k} - \sum_{j=1}^{k-1} \ell_{k,j}^2\, d_{j,j}$`;`

  `Falls diag` < 0`: Abbruch;  (Matrix ist nicht positiv definit)`

  $d_{k,k} \leftarrow$ `diag;`

  `Für` $i = k+1, \ldots, n$`:`

    $\ell_{i,k} \leftarrow \left(a_{i,k} - \sum_{j=1}^{k-1} \ell_{i,j} d_{j,j} \ell_{k,j} \right) / d_{k,k}$`;`

Bei der Implementierung werde die Einträge von A überschrieben: $\ell_{i,j}$ kommt in den Speicherplatz von $a_{i,j}$ und $d_{i,i}$ kommt in $a_{i,i}$.

Bemerkung 2.12 (Jupyter-Notebook).

- Das Cholesky-Verfahren benötigt ca. $\frac{1}{6}n^3$ Multiplikationen und Additionen, daher ungefähr halb so viele wie die LU-Zerlegung.
- LDL^T entspricht der LU-Zerlegung für $R = DL^T$. Pivotisierung würde die Symmetrie der Matrix zerstören und ist daher weder notwendig, noch sinnvoll.

- Die Lösung des Problems $A\,x = b$ ist gegeben durch

$$L\,y = b, \quad L^T x = D^{-1} y.$$

- Eine weitere Form der Cholesky-Zerlegung ist

$$A = \tilde{L}\,\tilde{L}^T,$$

wobei die Dreiecksmatrix $\tilde{L}$ nicht normiert ist. Es gilt

$$\tilde{L} = LD^{1/2}, \quad D^{1/2} = \mathrm{diag}(\sqrt{d_{1,1}}, \ldots, \sqrt{d_{n,n}}).$$

⌟

LU-Zerlegung einer Tridiagonalmatrix

Ein wichtiger Spezialfall der Bandmatrizen ist die *Tridiagonalmatrix*.

Definition 2.10. Die Matrix

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & 0 & \ldots & 0 \\ a_{2,1} & a_{2,2} & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_{n-1,n} \\ 0 & \ldots & 0 & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

heisst *Tridiagonalmatrix*.

Unter der Voraussetzung, dass die Gauss-Elimination ohne Pivotisierung möglich ist, z.B. im Speziellen bei *symmetrischen positiv definiten* Matrizen, ergibt die elementweise Auswertung der Gleichung $L\,R = A$, wie bei der Herleitung des Cholesky-Verfahrens, folgenden einfachen, sehr schnellen Algorithmus:

Algorithmus 2.6 (LU-Zerlegung einer Tridiagonalmatrix).
`Sei` $r_{1,1} := a_{1,1}$;
`Für` $j = 2, 3, \ldots, n$:
$\ell_{j,j-1} = a_{j,j-1}/r_{j-1,j-1}$
$r_{j-1,j} = a_{j-1,j}$
$r_{j,j} = a_{j,j} - \ell_{j,j-1} r_{j-1,j}$.

Bemerkung 2.13. In dieser Rekursion zur Bestimmung der LU-Zerlegung einer Tridiagonalmatrix ist der Rechenaufwand etwa $2n$ Operationen. Will man mit der berechneten LU-Zerlegung ein System lösen, dann sind zudem noch ca. n Operationen in der Vorwärtssubstitution und $2n$ Operationen in der Rückwärtssubstitution nötig. Insgesamt ergibt sich ein Aufwand von ca. $5n$ Operationen! ⌟

2.2 Lineare Ausgleichsrechnung

2.2.1 Einleitung, Problem

Die Ausgleichsrechnung ist eine mathematische Optimierungsmethode, um für eine Reihe von Messdaten die unbekannten Parameter eines Modells zu bestimmen bzw. schätzen.

Beispiel 2.9. Man betrachte eien einfachen Stromkreis, wobei I die Stromstärke, U die Spannung und R den Widerstand bezeichne. Die Modellannahmen beruhen auf dem Ohmschen Gesetz

$$U = R \cdot I.$$

Man nehme nun an, dass eine Messreihe (U_i, I_i) mit $i = 1, \dots, m$ durchgeführt wurde. Die Aufgabe besteht nun darin aus den Messdaten den Widerstand R zu bestimmen. Theoretisch müsste der gesuchte Wert *alle* Gleichungen

$$U_i = R \cdot I_i, \quad i = 1, \dots, m$$

erfüllen. Da die Messdaten mit (Mess)-Fehlern behaftet sind, werden die Gleichungen

$$R = \frac{U_i}{I_i}$$

unterschiedliche Resultate liefern. Das Vorgehen ist somit nicht Ziel führend.

Ein anderer Ansatz besteht darin den Fehler

$$f(R) := \sum_{i=1}^{m} (R \cdot I_i - U_i)^2 \to \min \tag{2.17}$$

zu minimieren. Da f eine quadratische Funktion ist, kann nur ein Extremum vorliegen, das durch die Nullstelle der Ableitung gegeben ist:

$$f'(R) = \sum_{i=1}^{m} 2(R \cdot I_i - U_i) \cdot I_i = 2R \left(\sum_{i=1}^{m} I_i^2 \right) - 2 \left(\sum_{i=1}^{m} U_i \, I_i \right) = 0.$$

Es folgt somit für den Parameter R

$$R = \frac{\sum_{i=1}^{m} U_i \, I_i}{\sum_{i=1}^{m} I_i^2}. \tag{2.18}$$

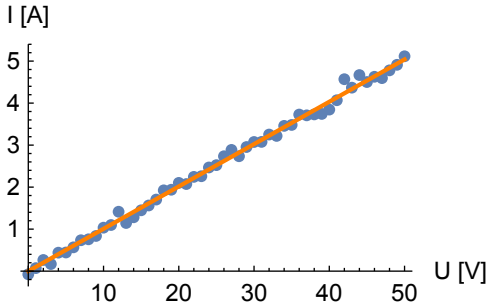


Abbildung 2.2: Messungen (U_i, I_i)

Beispiel 2.10 (vgl. Praktikum 4). Wir betrachten ein weiteres Beispiel. Gegeben sei ein periodischer Vorgang zum Beispiel aus einer Messung (t_i, u_i), für dessen Modellierung durch eine stetige Funktion $f(t)$ mit Periode T eine Linearkombination der trigonometrischen Funktionen

$$\begin{aligned} &\cos(\omega\, k\, t), \quad k = 0, \ldots, n \\ &\sin(\omega\, k\, t), \quad k = 1, \ldots, n \\ &\omega = \frac{2\pi}{T} \end{aligned}$$

verwendet wird. Gesucht sind daher die Koeffizienten a_k und b_k von

$$f_n(t) = \frac{1}{2}a_0 + \sum_{k=1}^{n} \left(a_k \cos(\omega\, k\, t) + b_k \sin(\omega\, k\, t)\right).$$

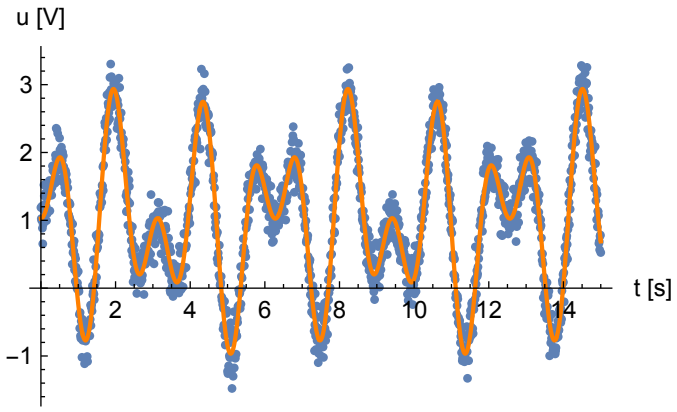


Abbildung 2.3: Messungen (t_i, u_i)

Zur Bestimmung einer geeigneten Approximation müssen $2n + 1$ Koeffizienten berechnet werden. Daher nehmen wir an, dass die Anzahl Messungen m grösser ist als die Anzahl zu bestimmende Koeffizienten

$$2n + 1 < m.$$

In dem Fall kann wieder die Fehlerquadratsumme minimiert werden

$$\sum_{i=1}^{m} (f_n(t_i) - u_i)^2 \to \min.$$

2.2.2 Das lineare Ausgleichsproblem

Aus theoretischen Überlegungen ist bekannt, dass eine bestimmte Grösse $b(t)$ über einen gewissen funktionalen Zusammenhang von einigen Parametern $x_1, \ldots, x_n$ abhängt:

$$b(t) = y(t; x_1, \ldots, x_n).$$

In der Ausgleichsrechnung geht es darum, aus einer Reihe von Messungen die Parameter zu bestimmen, die den gegebenen Prozess möglichst gut beschreiben. Dabei geht man davon aus, dass viel mehr Messungen als Parameter existieren und damit das Gleichungsproblem überbestimmt ist. Wobei das Gleichungssystem aufgrund von Messfehlern im Allgemeinen nicht konsistent ist. Daher versucht man, diejenigen Parameter $x_1, \ldots, x_n$ zu bestimmen, für die

$$\sum_{i=1}^{m} (y(t_i; x_1, \ldots, x_n) - b_i)^2 \to \min . \tag{2.19}$$

gilt. Der Schlüssel zu Entwicklung numerischer Verfahren liegt im Verständnis der Fälle, in denen der Ansatz bzw. das Modell *linear* ist, daher

$$y(t_i; x_1, \ldots, x_n) = a_{i,1}x_1 + \ldots + a_{i,n}x_n, \quad i = 1, \ldots m,$$

wobei die Koeffizienten $a_{i,k}$ gegeben sind.

Definition 2.11. Die Summe

$$\sum_{i=1}^{m} (y(t_i; x_1, \ldots, x_n) - b_i)^2 \tag{2.20}$$

nennt man auch *Summe der Fehlerquadrate* und die damit verbundene optimale Lösung *Kleinste-Quadrate-Lösung*.

Beispiel 2.11. Im Beispiel 2.9 gilt

$$a_{i,1} = I_i$$

und im Beispiel 2.10 ist

$$a_{i,k} = \left(\frac{1}{2}, \cos(1\,\omega\,t_i), \sin(1\,\omega\,t_i), \ldots, \cos(n\,\omega\,t_i), \sin(n\,\omega\,t_i)\right)_{i=1,\ldots,m}$$

⌟

In Matrixform mit $A = (a_{i,j})_{i,j=1}^{m,n} \in \mathbb{R}^{m\times n}$ und $b \in \mathbb{R}^m$ folgt für das Minimierungsproblem (2.19) die äquivalente Form

$$\|A\,x - b\|_2^2 \to \min_{x\in\mathbb{R}^n} .$$

Definition 2.12 (Lineares Ausgleichsproblem). Zu gegebenem $A \in \mathbb{R}^{m\times n}$ und $b \in \mathbb{R}^m$ bestimme man $x^* \in \mathbb{R}^n$ für das

$$\|A\,x^* - b\|_2^2 = \min_{x\in\mathbb{R}^n} \|A\,x - b\|_2^2 \tag{2.21}$$

gilt, wobei $n \ll m$ sei. Der im Allgemeinen nicht verschwindende Vektor $A\,x - b$ nennt man *Residuum* von x.

Beispiel 2.12 (Jupyter-Notebook). Gegeben seien die Messdaten

t	0	1	2	3	4	5
y	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{9}{10}$	$\frac{19}{20}$	$\frac{33}{34}$	$\frac{51}{52}$

mit einer vermuteten Gesetzmässigkeit der Form

$$y = b(t) = \alpha \frac{1}{1+t^2} + \beta.$$

Das zugehörige lineare Ausgleichsproblem in der Form (2.21) ist gegeben durch

$$x = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ \frac{1}{2} & 1 \\ \frac{1}{5} & 1 \\ \frac{1}{10} & 1 \\ \frac{1}{17} & 1 \\ \frac{1}{26} & 1 \end{pmatrix}, \quad b = \begin{pmatrix} \frac{1}{2} \\ \frac{3}{4} \\ \frac{9}{10} \\ \frac{19}{20} \\ \frac{33}{34} \\ \frac{51}{52} \end{pmatrix}.$$

⌟

Bemerkung 2.14. Grundsätzlich könnte man

$$\|Ax - b\| \to \min$$

bezüglich einer beliebigen Norm betrachten. Die Euklidische Norm $\|\cdot\|_2$ lässt jedoch eine schlüssige statistische Interpretation zu und ist zu dem über ein Skalarprodukt ($\|x\|_2 = (x,x)^{1/2}$) definiert. ⌟

Normalgleichungen

Die Lösung von (2.21) lässt sich auf die Lösung des linearen Gleichungssystems

$$A^T A\, x = A^T b$$

reduzieren, welche häufig als *Normalgleichungen* bezeichnet wird. Es gilt der Satz:

Satz 2.9. *$x^* \in \mathbb{R}^n$ ist genau dann Lösung des linearen Ausgleichsproblems (2.21), wenn x^* Lösung der Normalgleichungen*

$$A^T A\, x = A^T b \tag{2.22}$$

ist. Das System der Normalgleichungen hat stets mindestens eine Lösung. Sie ist genau dann eindeutig, wenn Rang$(A) = n$ gilt.

Da die Euklidische Norm über ein Skalarprodukt definiert ist, ist die Funktion

$$f(x) = \|A\,x - b\|_2^2 = (A\,x - b, A\,x - b)_2 = x^T A^T A\,x - 2\,x^T A\,b + b^T b$$

eine quadratische Funktion in $x \in \mathbb{R}^n$. f nimmt daher ein Extremum genau dort an, wo der Gradient

$$\nabla f(x) = 2(A^T Ax - A^T b) = 0$$

gilt. Diese Bedingung sind gerade die Normelgleichungen (2.22).

Bemerkung 2.15 (Geometrische Interpretation). Geometrisch kann das Problem $\|Ax - b\|_2^2 \to \min$ so interpretiert werden, dass die Differenz $b - Ax$ *senkrecht* auf dem Bildraum $\mathrm{im}(A) = \{Ax |\ x \in \mathbb{R}^n\}$ stehen muss. Es gilt also:

$$\|Ax - b\|_2^2 \to \min \ \Leftrightarrow \ Ax - b \perp \mathrm{im}(A), \tag{2.23}$$

vgl. Abbildung 2.4.

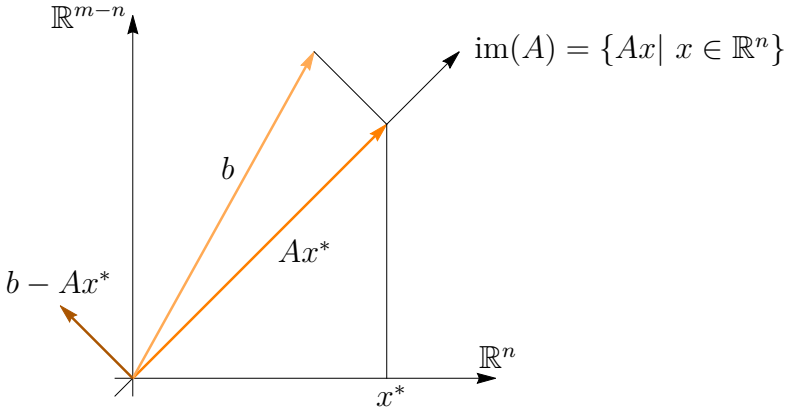


Abbildung 2.4: Geometrische Interpretation des linearen Ausgleichsproblems

Es gilt

$$\begin{aligned} Ax - b \perp \mathrm{im}(A) &\Leftrightarrow \ w^T(Ax - b) = 0 \quad \forall w \in \mathrm{im}(A) \\ &\Leftrightarrow \ (Ay)^T(Ax - b) = 0 \quad \forall y \in \mathbb{R}^n \\ &\Leftrightarrow \ y^T(A^TAx - A^Tb) = 0 \quad \forall y \in \mathbb{R}^n \\ &\Leftrightarrow \ A^TAx - A^Tb = 0, \end{aligned}$$

womit wir gerade die Normalgleichungen erhalten haben. ⌟

Bemerkung 2.16. Falls $A \in \mathbb{R}^{m\times n}$ vollen (Spalten-) Rang n hat, ist die Matrix $A^TA \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit. ⌟

2.2.3 Numerische Lösung des linearen Ausgleichproblems

Lösung der Normalgleichungen

Da die Matrix A^TA symmetrisch positiv definit ist, ergibt sich für die Lösung des linearen Ausgleichsproblems folgende Methode:

Algorithmus 2.7.

- `Berechne` A^TA, A^Tb.
- `Berechne die Cholesky-Zerlegung`

$$LDL^T = A^TA$$

  `von` A^TA.

- Löse
$$Ly = A^T b, \quad L^T x = D^{-1} y$$
durch Vorwärts, bzw. Rückwärtseinsetzen.

Beispiel 2.13 (Jupyter-Notebook). Für das Ausgleichsproblem in Beispiel 2.12 folgt

$$A^T A = \begin{pmatrix} \frac{1274691}{976820} & \frac{4193}{2210} \\ \frac{4193}{2210} & 6 \end{pmatrix}, \quad A^T b = \begin{pmatrix} \frac{2431921}{1953640} \\ \frac{22327}{4420} \end{pmatrix}.$$

Lösen des Gleichungssystems $A^T Ax = A^T b$ liefert die optimalen Parameter

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ 1 \end{pmatrix}$$

Daher folgt

$$y = b(t) = 1 - \frac{1}{2} \cdot \frac{1}{1+t^2}.$$

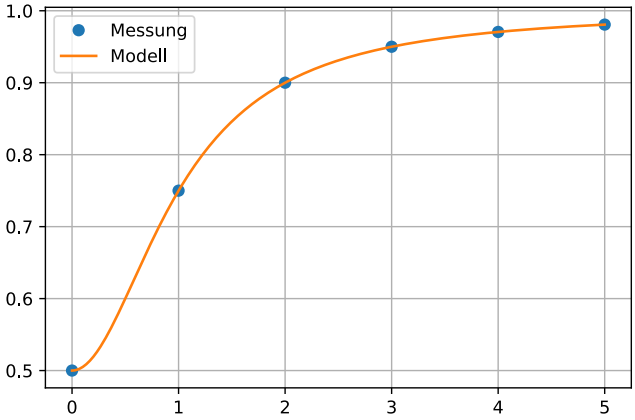


Abbildung 2.5: Lineares Ausgleichsproblem.

Bemerkung 2.17. Der Rechenaufwand für diese Methode setzt sich wie folgt zusammen:

- Berechnung von $A^T A$, $A^T b$: ca. $\frac{1}{2}mn^2$ Operationen,
- Cholesky-Zerlegung von $A^T A$: ca. $\frac{1}{6}n^3$ Operationen,
- Vorwärts- und Rückwärtssubstitution: ca. n^2 Operationen.

Für $m \gg n$ überwiegt der erste Anteil.

Diese Methode hat prinzipiell Defizite:

- Die Berechnung von A^TA ist für grosse m aufwendig und bringt die Gefahr von Genauigkeitsverlust durch Rundungsfehler.
- Bei der Lösung des Systems $A^TAx = A^Tb$ über das Cholesky-Verfahren werden die Rundungsfehler in A^TA und A^Tb verstärkt. Die Methode kann im Extremfall sogar instabil werden.

Die Konditionszahl für Matrizen nach Definition 2.4 muss auf rechteckige Matrizen $A \in \mathbb{R}^{m\times n}$ erweitert werden.

Definition 2.13 (Spektrale Konditionszahl). Die spektrale Konditionszahl der Matrix A ist gegeben durch

$$\kappa_2(A) := \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} \Big/ \min_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \frac{\max_{\|x\|_2=1} \|Ax\|_2}{\min_{\|x\|_2=1} \|Ax\|_2}. \tag{2.24}$$

Formal lässt man $\kappa_2(A) = \infty$ zu. Dies ist genau dann der Fall, wenn die Spalten von A *linear abhängig* sind. Das ist der Fall wenn $m < n$ gilt oder für $m \geq n$ der Rang von A kleiner n ist.

Beispiel 2.14. Sei

$$A = \begin{pmatrix} \sqrt{3} & \sqrt{3} \\ \delta & 0 \\ 0 & \delta \end{pmatrix}, \quad b = \begin{pmatrix} 2\sqrt{3} \\ \delta \\ \delta \end{pmatrix}, \quad 0 < \delta \ll 1.$$

Das lineare Ausgleichproblem hat die exakte Lösung

$$x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

für alle $\delta > 0$. Die Konditionszahl $\kappa_2(A^TA)$ ist in dem Fall gegeben durch

$$\kappa_2(A^TA) = \kappa_2(A)^2$$

wobei

$$\kappa_2(A) = \sqrt{\kappa_2(A^TA)} = \left(\frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}\right)^{1/2} = \sqrt{1 + \frac{6}{\delta^2}} \approx \frac{\sqrt{6}}{\delta}.$$

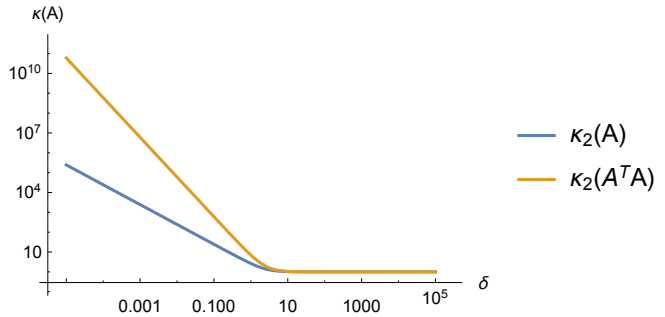


Abbildung 2.6: Lineares Ausgleichsproblem.

Es ist also zu erwarten, dass die Lösung von A^TA schneller numerisch instabil wird, als die Lösung basierend auf A direkt.

⌟

Beispiel 2.15. Das folgenden Beispiel zeigt, dass selbst ein linearer Fit mit einem Polynom 5-ter Ordnung in der Praxis schnell problematisch werden kann.

Seien $m = 1000$ Samples einer Messung gegeben. Nun soll das Messsignal mit einem Polynom 5-ter Ordnung approximiert werden. Die Basisfunktionen sind daher gegeben durch

$$1, k, k^2, \ldots, k^5.$$

und entsprechend folgt für die Matrix $A \in \mathbb{R}^{1000\times 6}$

$$a_{k,\cdot} = \left(1, k, \ldots, k^5\right), \quad k = 1, \ldots, 1000.$$

Für A^TA erhält man (in Mathematica exakt gerechnet)

$$A^TA = \begin{pmatrix} 1000 & 500500 & 333833500 & 250500250000 & 200500333333300 & 167167083333250000 \\ 500500 & 333833500 & 250500250000 & 200500333333300 & 167167083333250000 & 143357642856976190500 \\ 333833500 & 250500250000 & 200500333333300 & 167167083333250000 & 143357642856976190500 & 125500583333041666750000 \\ 250500250000 & 200500333333300 & 167167083333250000 & 143357642856976190500 & 125500583333041666750000 & 111611777777311111333333300 \\ 200500333333300 & 167167083333250000 & 143357642856976190500 & 125500583333041666750000 & 111611777777311111333333300 & 100500749999300000499999850000 \\ 167167083333250000 & 143357642856976190500 & 125500583333041666750000 & 111611777777311111333333300 & 100500749999300000499999850000 & 91409924241424243424241924242500 \end{pmatrix}$$

Für die Konditionszahl $\kappa_2(A^TA)$ folgt in dem Fall

$$\kappa_2(A^TA) \approx 3.35 \cdot 10^{30},$$

und entsprechend für $\kappa_2(A)$

$$\kappa_2(A) \approx 1.83 \cdot 10^{15},$$

womit entsprechende numerische Probleme erwartet werden können. Das Problem ist auch für A schlecht konditioniert, mit Vorbehalt jedoch noch rechenbar, für A^TA sieht es jedoch sehr schlecht aus.

⌟

Bemerkung 2.18. Trotz dieser Nachteile wird obiges Verfahren in der Praxis oft benutzt, insbesondere bei Problemen mit gut konditioniertem A. Im allgemeinen aber ist die im nächsten Abschnitt behandelte Alternative vorzuziehen, da sie stabiler ist und der Rechenaufwand nur wenig höher ist.

⌟

Lösung über QR-Zerlegung

Eine Alternative zur LU-Zerlegung bietet die QR-Zerlegung einer Matrix. Die QR-Zerlegung wird in der Praxis häufig in der Ausgleichsrechnung und Berechnung von Eigenwerten eingesetzt. Die Abgrenzung zu den in Kapitel 2.1 erwähnten Methoden ist gegeben durch

1. Die LU-Zerlegung ist nur für $(n \times n)$-Matrizen sinnvoll, während eine QR-Zerlegung für allgemeine rechteckige $(m \times n)$ Matrizen konstruiert werden kann.

2. Die Methoden zur Berechnung der QR-Zerlegung einer $(n \times n)$ Matrix A sind im Allgemeinen stabiler als die Gauss-Elimintation mit Pivotisierung zur Berechnung einer LU-Zerlegung von A.

3. Der Aufwand zur Berechnung der QR-Zerlgung einer $(n \times n)$-Matrix A ist im Allgemeinen höher als bei der Berechnung einer LU-Zerlegung von A über Gauss-Elimination mit Pivotisierung

	LU-Zerlegung	QR-Zerlegung
Matrix	$(n \times n)$	$(m \times n)$
numerische Stabilität		stabiler
Rechenaufwand	niedriger	

Tabelle 2.2: Vergleich LU / QR Methode

Definition 2.14 (Orthogonale Matrizen). Eine Matrix $Q \in \mathbb{R}^{n\times n}$ heisst *orthogonal*, falls

$$Q^T Q = \mathbb{1} \tag{2.25}$$

gilt.

Bemerkung 2.19. Die Inverse einer solchen Matrix ist demnach gegeben durch

$$Q^{-1} = Q^T.$$

Satz 2.10. *Sei $Q \in \mathbb{R}^{n\times n}$ orthogonal. Dann gilt:*

1. *Q^T ist orthogonal.*
2. *$\|Qx\|_2 = \|x\|_2$ für alle $x \in \mathbb{R}^n$.*
3. *$\kappa_2(Q) = 1$.*
4. *Für beliebiges $A \in \mathbb{R}^{m\times n}$ bzw. $A \in \mathbb{R}^{n\times m}$, gilt $\|A\|_2 = \|QA\|_2 = \|AQ\|_2$.*
5. *Es gilt $\kappa_2(A) = \kappa_2(AQ) = \kappa_2(QA)$*
6. *Sei $\tilde{Q} \in \mathbb{R}^{n\times n}$ orthogonal, dann ist $Q\tilde{Q}$ orthogonal.*

Gelingt es $A \in \mathbb{R}^{m\times n}$ als Produkt

$$A = Q\,R$$

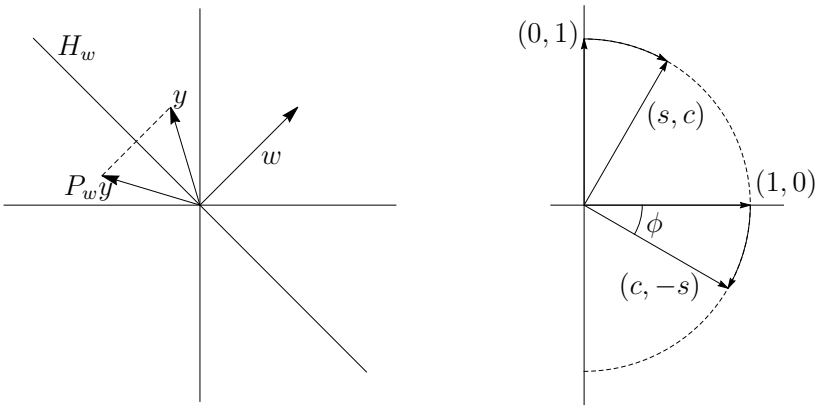
zu schreiben, wobei Q orthogonal und R eine obere Dreiecksmatrix ist, so gilt wegen (2.25)

$$A\,x = b \quad \Leftrightarrow \quad Q\,R\,x = b \quad \Leftrightarrow \quad R\,x = Q^T\,b,$$

daher das Problem reduziert sich wieder auf Rückwärtseinsetzen, falls R, also A invertierbar ist.

Die Verfahren der QR-Zerlegung verfolgen die gleiche Grundidee wie bei der LU-Zerlegung. Die Matrix A wird Schritt für Schritt auf obere Dreiecksform transformiert, in dem man sie mit geeigneten orthogonalen Matrizen Q_i multipliziert. Das Produkt Q der

Q_i ist wegen Satz 2.10 wieder eine orthogonale Matrix. Die obere Dreiecksmatrix ist gegeben durch $R = Q^T A$. Die einzelnen Faktoren Q_i werden meist nach zwei unterschiedlichen Prinzipien konstruiert, der *Householder-Spiegelungen* und *Givens-Rotationen*.



(a) Householder Spiegelung (b) Givens Rotation

Abbildung 2.7: Orthogonale Matrizen: Householder Spiegelung und Givens Rotation

Householder-Transformation [Jupyter-Notebook] Die Householder-Transformation ist gegeben durch die lineare Abbildung

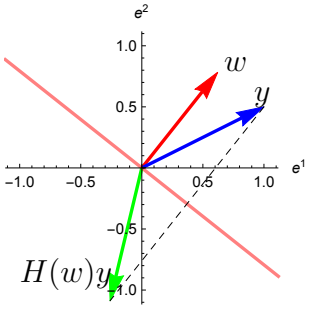
$$\begin{aligned} H : \mathbb{R}^n &\to \mathbb{R}^n \\ y &\mapsto z = H(w) \cdot y \end{aligned}$$

mit

$$H(w) = \mathbb{1} - w \cdot w^T, \quad \text{wobei } \|w\| = \sqrt{2}.$$

Geometrisch beschreibt die Abbildung H eine Spiegelung an der Hyperebene

$$H_w := \{y \in \mathbb{R}^m \mid \langle y, w \rangle = 0\}.$$



Beweis. H ist eine Spiegelung an der Hyperebene.

1. $y \in \mathbb{R}^m$ kann als $y = \alpha w + w^\perp$ mit $\langle w, w^\perp \rangle = 0$ geschrieben werden, wobei für $w^\perp \in \mathbb{R}^m$ nicht zwingend $\|w^\perp\|_2 = \sqrt{2}$ erfüllt sein muss.

2. Es gilt

$$\begin{aligned} H(w)y &= H(w)(\alpha w + w^{\perp}) = (\mathbb{1} - w \cdot w^T) \cdot (\alpha w + w^{\perp}) \\ &= \alpha w + w^{\perp} - w \cdot w^T \cdot \alpha w + w \cdot \underbrace{w^T \cdot w^{\perp}}_{=0} \\ &= \alpha w + w^{\perp} - \alpha w \cdot \underbrace{w^T \cdot w}_{=2} \\ &= -\alpha w + w^{\perp} \end{aligned}$$

3. $H(w)y = -\alpha w + w^{\perp}$ ist somit an der Ebene H_w gespiegelt. □

Die allgemeine Schreibweise der Householder-Transformation für einen beliebigen Vektor w ist gegeben durch

$$H(w) = \mathbb{1} - 2\,\frac{w \cdot w^T}{\langle w, w \rangle}, \tag{2.26}$$

wobei $w \cdot w^T$ das Kroneckerprodukt

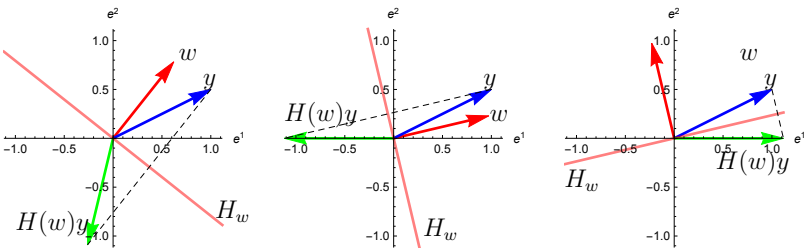
$$w \cdot w^T = (w_i\, w_j)_{i,j=1\ldots m} \in \mathbb{R}^{m \times m}$$

sei.

Die Householder-Transformation auf die QR-Zerlegung angewandet führt zum folgenden Vorgehen: Gegeben sei der Spaltenvektor y einer Matrix $A \in \mathbb{R}^{m \times n}$. Gesucht ist der Vektor w mit welchem die Hyperebene H_w der Spiegelung definiert ist, so dass

$$H(w) \cdot y = \pm \|y\|_2 e^1$$

gilt, wobei mit e^1 der erste Einheitsvektor $e^1 = (1, 0, \ldots, 0)^T$ bezeichne.



Sei $y \in \mathbb{R}^m,\ y \notin \operatorname{span}\{e^1\}$ gegeben.

$$H(w) \cdot y = y - 2\frac{w \cdot w^T}{w^T \cdot w} \cdot y$$

mit

$$w \cdot w^T \cdot y = \left(\sum_{k=1}^{m} (w_i\, w_k)\, y_k \right)_{i,j} = \left(\sum_{k=1}^{m} (w_k\, y_k)\, w_i \right)_{i,j} = w^T \cdot y \cdot w$$

folgt

$$H(w) \cdot y = y - \underbrace{2\frac{w^T \cdot y}{w^T \cdot w}}_{\in \mathbb{R}} \cdot w \overset{\text{Ziel!}}{=} \pm \|y\|_2 e^1$$

Der Vektor w ist daher eine Linearkombination von y und e^1, wir machen daher den Ansatz:

$$w = y + \alpha e^1.$$

Für $H(w) = H(y + \alpha e^1)$ folgt

$$\begin{aligned} H(w)y &= y - 2\frac{(y+\alpha e^1)^T \cdot y}{(y+\alpha e^1)^T \cdot (y+\alpha e^1)} \cdot (y + \alpha e^1) \\ &= y - 2\frac{\|y\|^2 + \alpha\, y_1}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2} \cdot (y + \alpha e^1) \\ &= \underbrace{\left(1 - 2\frac{\|y\|^2 + \alpha\, y_1}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2}\right) \cdot y}_{\text{1. Summand, Ziel} = 0} + \underbrace{2\frac{\|y\|^2 + \alpha\, y_1}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2}\alpha e^1}_{\text{2. Summand}}. \end{aligned}$$

Für den ersten Summanden gilt

$$\begin{aligned} 1 - 2\frac{\|y\|^2 + \alpha\, y_1}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2} &= \frac{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2 - 2\|y\|^2 - 2\alpha\, y_1}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2} \\ &= \frac{\alpha^2 - \|y\|^2}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2} \overset{!}{=} 0 \end{aligned}$$

und es folgt $\alpha = \pm\|y\|$. Für den zweiten Summanden erhalten wir mit $\alpha = \pm\|y\|$

$$\begin{aligned} 2\frac{\|y\|^2 + \alpha\, y_1}{\|y\|^2 + 2\,\alpha\, y_1 + \alpha^2}\alpha e^1 &= \pm 2\frac{\|y\|^2 \pm \|y\|\, y_1}{\|y\|^2 \pm 2\,\|y\|\, y_1 + \|y\|^2}\|y\| e^1 \\ &= \pm\frac{\|y\|^2 \pm \|y\|\, y_1}{\|y\|^2 \pm \|y\|\, y_1}\|y\| e^1 = \pm\|y\| e^1. \end{aligned}$$

Wie gewünscht spiegelt $H(w)$ mit $w = y + \alpha e^1$ den Vektor y auf den Einheitsvektor e^1.

Um Auslöschung in der Berechnung von

$$w = (y_1 \pm \|y\|_2, y_2, \ldots, y_m)^T$$

zu vermeiden, wählt man

$$\begin{aligned} w &= y + \operatorname{sign}(y_1)\|y\|_2 e^1 \\ \text{mit } \operatorname{sign}(s) &= \begin{cases} 1 & \text{für } s \geq 0 \\ -1 & \text{sonst.} \end{cases} \end{aligned}$$

Für die QR-Zerlegung mit Hilfe der Householder-Transformation folgt somit Spaltenweise:

1. Mit $Q_1 = H(w^{(1)})$, $\quad w^{(1)} = y^{(1)} + \operatorname{sign}(y_1^{(1)})\ \|y^{(1)}\|_2 e^1$, wobei $y^{(1)}$ erster Spaltenvektor von A sei, folgt

$$Q_1 \cdot A = \begin{pmatrix} * & \ldots & * \\ 0 & & \\ \vdots & \left[\ \tilde{A}^{(2)}\ \right] & \\ 0 & & \end{pmatrix} = A^{(2)}$$

2. Mit $\tilde{Q}^{(2)} = H(\tilde{w}^{(2)}), \quad \tilde{w}^{(2)} = \tilde{y}^{(2)} + \text{sign}(\tilde{y}_1^{(2)}) \, \|\tilde{y}^{(2)}\|_2 \tilde{e}^1$, wobei $\tilde{y}^{(2)}$ erster Spaltenvektor von $\tilde{A}^{(2)}$ sei, folgt

$$Q_2 = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \tilde{Q}^{(2)} & \\ 0 & & & \end{pmatrix} \text{ und } Q_2 \cdot Q_1 \cdot A = \begin{pmatrix} * & \dots & * \\ 0 & * & \dots & * \\ & 0 & & \\ \vdots & \vdots & \tilde{A}^{(3)} & \\ 0 & 0 & & \end{pmatrix} = A^{(3)} \quad (2.27)$$

Wir erhalten daher

$$Q_n \cdot \ldots \cdot Q_1 A = R$$

und

$$A = Q \cdot R,$$

wobei $Q = (Q_n \cdot \ldots \cdot Q_1)^T = Q_1^T \cdot \ldots \cdot Q_n^T$. Die Submatrizen von Q sind gegeben durch die Householder-Transformation:

$$H(w) = \mathbb{1} - 2 \frac{w \cdot w^T}{\langle w, w \rangle}$$
$$w = y + \text{sign}(y_1) \|y\|_2 e^1$$
$$\text{mit } \text{sign}(s) = \begin{cases} 1 & \text{für } s \geq 0 \\ -1 & \text{sonst.} \end{cases}$$

Zurück zur linearen Ausgleichsrechnung: Der Satz 2.10 zeigt, dass die Multiplikation einer Matrix A mit einer orthogonalen Matrix Q die Euklidische Norm eines Vektors nicht verändert. Die Minimierung von $\|Ax - b\|_2$ ist also für *jede* orthogonale Matrix Q *äquivalent* zur Aufgabe

$$\|Q^T(A\,x - b)\|_2^2 = \|Q^T\,A\,x - Q^T\,b\|_2^2 \to \min\,.$$

Mit $A = Q\,R$ folgt analog zur Normalgleichung die notwendige Bedingung für das Extremum

$$R\,x = Q^T b.$$

Für die Lösung des linearen Ausgleichsproblems folgt die Methode:

Algorithmus 2.8.

- `Berechne die QR-Zerlegung (`R `und` $Q^T b$`)`
- `Löse`

$$R\,x = Q^T b$$

`durch Rückwärtseinsetzen.`

Bemerkung 2.20. Der Rechenaufwand für die Methode ist gegeben durch

- QR-Zerlegung mittels Householder-Spiegelungen, falls $m \gg n$ ca. $m\,n^2$
- Berechnung von $Q^T\,b$ ca. $2\,m\,n$ Operationen

- Rückwärtssubstitution: ca. $\frac{1}{2}n^2$ Operationen.

Die Methode ist im Rechenaufwand um Faktor 2 teurer als die Lösung über die Normelgleichungen, jedoch bedeutend robuster. ⌟

Beispiel 2.16 (Jupyter-Notebook). Abschliessend betrachten wir eine industrielle Anwendung aus der Gas Analytik. In spektrometrischen Messungen hat man oft das Problem, dass die sogenannte Baseline (Störung) der Messung unbekannt ist. Wir gehen davon aus, dass das gesuchte Messsignal ein Puls (z.B. gegeben durch eine Lorentz Shape Funktion) mit einem Drift überlagert ist. Für den Drift nehmen wir an, dass dieser eine polynomiale Form hat. Im Jupyter-Notebook wird basierend auf synthetischen Messwerte die lineare Ausgleichsrechnung angewandt.

Die Ansatzfunktionen bestehen aus einem Polynom und einer Lorentz Shape-Fuktion gegeben durch

$$l(x) = \frac{1}{1+x^2}, \quad x = \frac{t - t_0}{s/2}.$$

Das Beispiel zeigt, dass das Ausgleichsproblem schon für nur 1000 Messungen sehr schlecht konditioniert ist. Die Skalierung der x-Achse spielt hier eine entscheidende Rolle. Durch einen Wechsel von Sample-Nummer zum Intervall $[0, 1]$ reduziert sich die Konditionszahl von $5 \cdot 10^{15}$ auf 25'000! ⌟

2.3 Nichtlineare Gleichungen

2.3.1 Problemstellung

Zum Einstieg in das Thema betrachten wir das folgende Beispiel.

Beispiel 2.17 (Jupyter-Notebook). Gesucht ist eine Lösung des Gleichungssystems:

$$\begin{aligned} 6x &= \cos(x) + 2\,y \\ 8y &= x\,y^2 + \sin(x). \end{aligned} \tag{2.28}$$

Das Gleichungssystem ist aufgrund der nichtlinearen Funktionen $\cos(x), \sin(x), y^2$ nichtlinear. Wir benötigen für das numerische Lösen des Gleichungssystems neue Methoden. Dazu betrachten wir zwei verschiedene Gleichungstypen:

- Fixpunktgleichungen

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{pmatrix}$$

  Auf das Beispiel angewandt folgt

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{6}\left(\cos(x) + 2\,y\right) \\ \frac{1}{8}\left(x\,y^2 + \sin(x)\right) \end{pmatrix}.$$

- Nullstellengleichungen.

$$\begin{pmatrix} g_1(x_1,\dots,x_n) \\ \vdots \\ g_n(x_1,\dots,x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

  Für unser Einstiegsbeispiel folgt wiederum

$$\begin{pmatrix} \cos(x) + 2\,y - 6x \\ x\,y^2 + \sin(x) - 8y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

⌟

Für das Beispiel 2.17 konnte als nichtlineares Gleichungssystem sowohl eine Fixpunkt- wie auch eine Nullstellengleichung notiert werden. Wir definieren ein allgemeines nichtlineares Gleichungssystem als Nullstellengleichung:

Definition 2.15 (Nichtlineares Gleichungssystem).
Zu gegebendem $\mathbf{g} = (g_1,\dots,g_m)^T : \mathbb{R}^n \to \mathbb{R}^m$ finde man $x^* = (x_1^*,\dots,x_n^*)^T \in \mathbb{R}^n$ so, dass

$$\begin{aligned} g_1(x_1^*,\dots,x_n^*) &= 0 \\ &\vdots \\ g_m(x_1^*,\dots,x_n^*) &= 0. \end{aligned} \tag{2.29}$$

Häufig benutzt man die kompakte Schreibweise

$$\mathbf{g}(x^*) = 0,$$

mit $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$. Im Folgenden sei $n = m$.

Bemerkung 2.21. Der Spezialfall $n = m = 1$ kennen wir schon und wird *skalare* Gleichung in *einer* Unbekannten bezeichnet. ⌟

Die Herkunft nichtlinearer Gleichungen ist sehr vielfältig. In der Technik und Naturwissenschaften liegt der Ursprung auch oft bei (partiellen) Differentialgleichungen, wie auch das folgende Beispiel illustriert.

Beispiel 2.18 (Jupyter-Notebook). Als Beispiel für ein nichtlineares Randwertproblem betrachten wir die Gelfand-Gleichung im eindimensionalen auf dem Intervall $[-1, 1]$, wird im 1d auch Bratu's Randwertproblem genannt:

$$\begin{aligned} -u''(x) &= \lambda e^{u(x)} \\ u(-1) &= u(1) = 0. \end{aligned} \tag{2.30}$$

Die Gleichung erscheint in der Thermodynamik im Zusammenhang mit explodierenden thermischen Reaktionen, in der Astrophysik z. B. als Emden-Chandrasekhar-Gleichung.

Für das Randwertproblem (2.30) kann eine analytische Lösung berechnet werden

$$u(x) = \log\left(\frac{\theta\left(1 - \tanh^2\left(\sqrt{\frac{\theta}{2}}\,x\right)\right)}{\lambda}\right),$$

wobei $\theta = \theta(\lambda)$ Lösung der nichtlinearen Gleichung

$$\theta \left(1 - \tanh^2 \left(\sqrt{\frac{\theta}{2}} \right) \right) = \lambda$$

ist. Aus der Abbildung 2.8 der Funktion $\theta(\lambda)$ folgt, dass nur für $\lambda \in [0, \lambda^*]$ Lösungen existieren. Weiter ist zu bemerken dass für $\lambda \in (0, \lambda^*)$ genau zwei Lösungen existieren. Wir bezeichnen die 1. Lösungen (blau) als *untere* und die 2. Lösungen (orange) als *obere Lösungen* (bzw. unterer / oberer Ast).

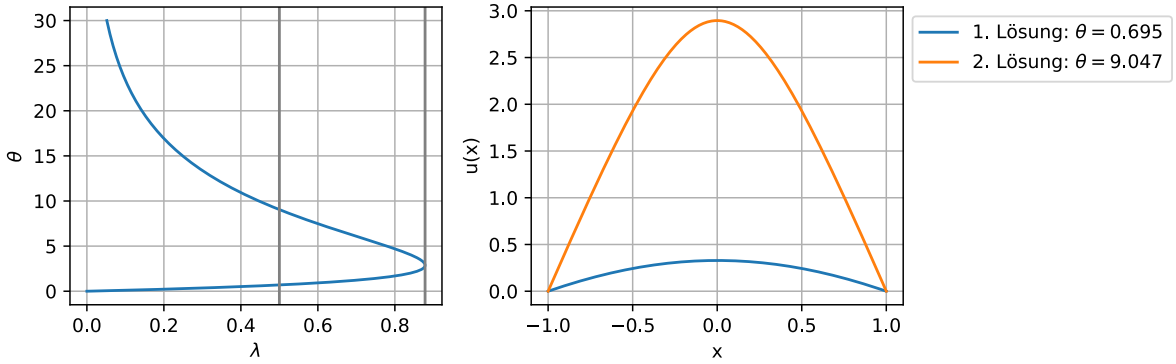


Abbildung 2.8: Für das Randwertproblem (2.30) existieren für $\lambda \in (0, \lambda^*)$ zwei Lösungen. Der Graph rechts zeigt die beiden Lösungen für $\lambda = 0.5$.

Wir diskretisieren das Problem analog zu Beispiel 2.6 mit Hilfe von finiten Differenzen. Dazu zerlegen wir das Intervall $[-1, 1]$ wiederum in die Teilintervalle

$$[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n],$$

wobei $x_0 = -1 < x_1 < \ldots < x_{n-1} < x_n = 1$ gilt und schreiben

$$u_i = u(x_i) \quad \text{für } i = 0, \ldots, n.$$

Damit folgen die $n - 1$ diskretisierten Gleichungen

$$\begin{aligned} -u_0 + 2\,u_1 - u_2 &= h^2 \lambda\, e^{u_1} \\ &\vdots \\ -u_{n-2} + 2\,u_{n-1} - u_n &= h^2 \lambda\, e^{u_{n-1}} \end{aligned} \tag{2.31}$$

oder in der Form von (2.29)

$$\begin{aligned} -u_0 + 2\,u_1 - u_2 - h^2 \lambda\, e^{u_1} &= 0 \\ &\vdots \\ -u_{n-2} + 2\,u_{n-1} - u_n - h^2 \lambda\, e^{u_{n-1}} &= 0. \end{aligned} \tag{2.32}$$

⌟

2.3.2 Fixpunktiteration

Die Fixpunktgleichungen haben den Vorteil einer einfachen Iterationsvorschrift (vgl. Analysis 1 Vorlesung). Sei der *Startwert* x_0 gegeben, dann folgt

$$x_{k+1} = \Phi(x_k), \quad k = 0, 1, 2, 3, \ldots \tag{2.33}$$

Satz 2.11 (Banachscher Fixpunktsatz). *Sei X ein linear normierter Raum mit Norm $\|\cdot\|$. $E \subset X$ sei eine kompakte Teilmenge von X. Die Abbildung Φ sei eine* Selbstabbildung *auf E:*

$$\Phi : E \to E. \tag{2.34}$$

Ferner sei Φ eine Kontraktion *auf E*

$$\|\Phi(x) - \Phi(y)\| \leq L \, \|x - y\| \quad \forall x, y \in E, \text{ mit } L < 1. \tag{2.35}$$

Dann gilt:

1. *Es existiert genau ein Fixpunkt $x^* \in E$ von Φ.*
2. *Für beliebiges $x_0 \in E$ konvergiert*

   $$x_{k+1} = \Phi(x_k), \quad k = 0, 1, 2, \ldots$$

   gegen den Fixpunkt x^*.
3. *A-priori-Fehlerabschätzung*

   $$\|x_k - x^*\| \leq \frac{L^k}{1 - L} \|x_1 - x_0\|. \tag{2.36}$$

4. *A-posteriori-Fehlerabschätzung*

   $$\|x_k - x^*\| \leq \frac{L}{1 - L} \|x_k - x_{k-1}\|. \tag{2.37}$$

Der obige Satz 2.11 ist sehr allgemein. Daher seien hier noch zwei Spezialfälle erwähnt:

Korrolar 2.2. *Sei $X = \mathbb{R}$, $E = [a, b]$ und Φ eine auf E stetig differenzierbare Funktion. Es gelte*

$$\Phi : [a, b] \to [a, b] \quad \textit{(Selbstabbildung)},$$

und

$$\max_{x \in [a,b]} |\Phi'(x)| =: L < 1. \quad \textit{(Kontraktion)}.$$

Dann sind alle Voraussetzungen aus Satz 2.11 erfüllt für $\|\cdot\| = |\cdot|$.

Korrolar 2.3. *Sei $X = \mathbb{R}^n$, $E \subset \mathbb{R}^n$ eine abgeschlossene konvexe Menge, und $\Phi : E \to \mathbb{R}^n$ sei auf E eine stetig differenzierbare Funktion. Es gelte*

$$\Phi : E \to E \quad \textit{(Selbstabbildung)},$$

und bzgl. einer Vektornorm $\|\cdot\|$ auf $\mathbb{R}^n$ gelte für die zugehörige Matrixnorm

$$\max_{x\in E} \|\Phi'(x)\| =: L < 1. \quad \textit{(Kontraktion).} \tag{2.38}$$

Dann sind alle Voraussetzungen aus Satz 2.11 erfüllt.

In (2.38) ist

$$\Phi'(x) = \begin{pmatrix} \frac{\partial}{\partial x_1}\Phi_1(x) & \dots & \frac{\partial}{\partial x_n}\Phi_1(x) \\ \vdots & & \vdots \\ \frac{\partial}{\partial x_1}\Phi_n(x) & \dots & \frac{\partial}{\partial x_n}\Phi_n(x) \end{pmatrix}$$

die Jacobi-Matrix von Φ an der Stelle x.

Korrolar 2.4. *Sei $X = \mathbb{R}^n$, $x^* \in \mathbb{R}^n$, so dass $\Phi(x^*) = x^*$ und Φ stetig differenzierbar in einer Umgebung von x^*. Bezüglich einer Vektornorm $\|\cdot\|$ auf $\mathbb{R}^n$ gelte für die zugehörige Matrixnorm*

$$\|\Phi'(x)\| < 1.$$

Sei $B_\varepsilon := \{x \in \mathbb{R}^n \mid \|x - x^*\| \le \varepsilon\}$. Für $E = B_\varepsilon$ mit $\varepsilon > 0$ hinreichend klein sind alle Voraussetzungen aus Satz 2.11 erfüllt.

Beispiel 2.19 (Jupyter-Notebook). Wir betrachten das eingangs vorgestellte Problem (2.28). Das zugehörige Fixpunkt-Problem ist gegeben durch

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{6}\left(\cos(x) + 2\,y\right) \\ \frac{1}{8}\left(x\,y^2 + \sin(x)\right) \end{pmatrix}.$$

Sei

$$\Phi(x, y) = \begin{pmatrix} \frac{1}{6}\left(\cos(x) + 2\,y\right) \\ \frac{1}{8}\left(x\,y^2 + \sin(x)\right) \end{pmatrix}$$

wir zeigen, dass auf $E = [0,1]\times[0,1]$ eine eindeutige Lösung besitzt. Um den Fixpunktsatz 2.11 anwenden zu können, wählen wir $X = \mathbb{R}^2$ und zeigen, dass Φ eine Selbstabbildung und Kontraktion ist.

Selbstabbildung Zu zeigen ist, dass $\Phi([0,1]^2) \subset [0,1]^2$ gilt, was mit Hilfe von Monotonie-Überlegungen sofort folgt:

$$\begin{aligned} 0 \le \quad & \frac{1}{6}\left(\cos(x) + 2\,y\right) \le \frac{1}{6}\left(\cos(0) + 2\cdot 1\right) < \frac{3}{6} \quad < 1 \\ 0 \le \quad & \frac{1}{8}\left(x\,y^2 + \sin(x)\right) \le \frac{1}{8}\left(1\cdot 1^2 + \sin(1)\right) < \frac{2}{8} \quad < 1. \end{aligned}$$

Kontraktion Wir zeigen $\max_{(x,y)\in[0,1]^2} \|\Phi'(x,y)\|_\infty < 1$: Für die Jacobi-Matrix folgt

$$\Phi'(x,y) = \begin{pmatrix} -1/6\sin(x) & 1/3 \\ 1/8y^2 + 1/8\cos(x) & 1/4xy \end{pmatrix}$$

und damit

$$\|\Phi'(x,y)\|_\infty = \max\left\{\frac{1}{6}\sin(x) + \frac{1}{3}, \frac{1}{8}y^2 + \frac{1}{8}\cos(x) + \frac{1}{4}xy\right\} \le \max\left\{\frac{1}{2}, \frac{1}{2}\right\} = \frac{1}{2} < 1$$

Im Jupyter-Notebook wird für $(x_0, y_0) = (0.5, 0.5)$ die Fixpunkt-Folge

$$(x_{n+1}, y_{n+1}) = \Phi(x_n, y_n) \quad \text{für } n = 0, 1, \ldots$$

berechnet. ⌟

Definition 2.16 (Konvergenzordnung). Eine konvergente Folge $\{x_k\}_{k\in\mathbb{N}} \subset \mathbb{R}^n$ mit Grenzwert x^* hat die Konvergenzordnung p, falls für ein $k_0 \in \mathbb{N}$

$$\|x_{k+1} - x^*\| \leq c\|x_k - x^*\|^p$$

für alle $k \geq k_0$ gilt, wobei

$$0 < c < 1 \quad \text{falls} \quad p = 1.$$

Beispiel 2.20 (Jupyter-Notebook). Für den Fall $c = 0.1$ und $\|x_0 - x^*\| = 1$ kann der Unterschied zwischen der linearen und quadratischen Konvergenz sehr gut veranschaulicht werden (vgl. auch Abbildung 2.9):

k	0	1	2	3	4	5
$p = 1$	1×10^{-1}	1×10^{-2}	1×10^{-3}	1×10^{-4}	1×10^{-5}	1×10^{-6}
$p = 2$	1×10^{-1}	1×10^{-3}	1×10^{-7}	1×10^{-15}	1×10^{-31}	1×10^{-63}

Tabelle 2.3: Entwicklung des Fehlers im Fall einer linearen bzw. quadratischen Konvergenzordnung.

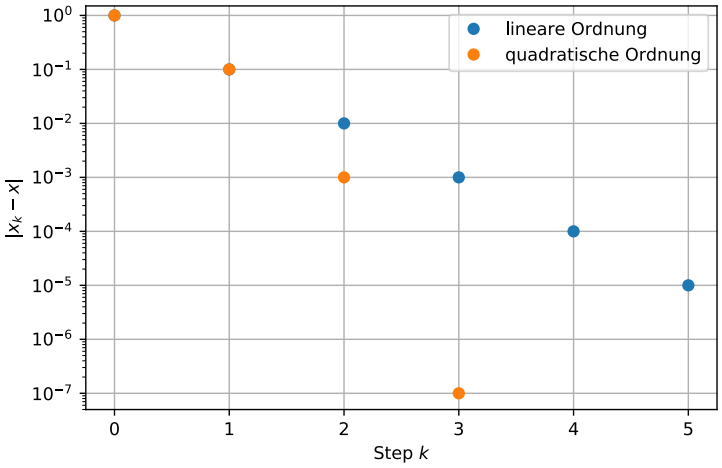


Abbildung 2.9: Lineare und quadratische Konvergenzordnung.

⌟

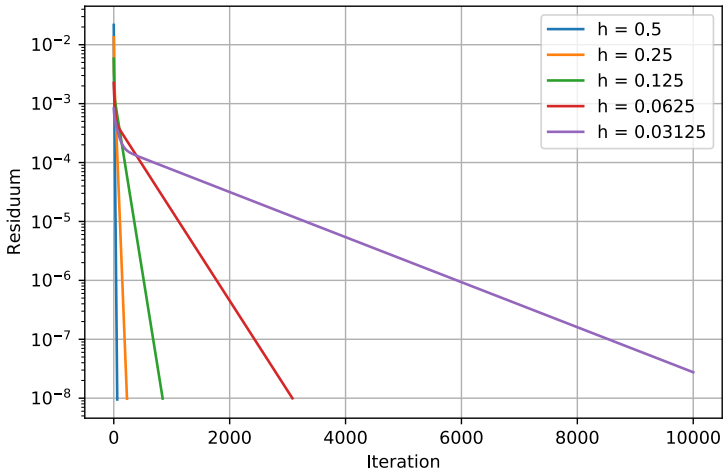


Abbildung 2.10: Konvergenz Analyse der Fixpunktiteration für die mit finite Differenzen diskretisierte Gelfand-Gleichung.

Bemerkung 2.22. Sei $x_{k+1} = \Phi(x_k)$, $k = 0, 1, \ldots$ eine konvergente Fixpunktiteration mit Fixpunkt x^* und zweimal stetig differenzierbarem Φ. Mit Hilfe der Taylor-Entwicklung

$$\Phi(x_k) = \Phi(x^*) + \Phi'(x^*)(x_k - x^*) + \mathcal{O}(\|x_k - x^*\|^2)$$

folgt

$$x_{k+1} - x^* = \Phi(x_k) - \Phi(x^*) = \Phi'(x^*)(x_k - x^*) + \mathcal{O}(\|x_k - x^*\|^2).$$

Im Normalfall, wenn $0 \neq \|\Phi'(x^*)\| < 1$ gilt, hat die Fixpunktiteration die Konvergenzordnung $p = 1$ (lineare Konvergenz). Quadratische Konvergenz hat man höchstens dann, wenn $\Phi'(x^*) = 0$ gilt. ⌟

Beispiel 2.21 (Jupyter-Notebook). Wir versuchen als abschliessendes Beispiel für die Fixpunktiteration numerisch eine Lösung der Gelfand-Gleichung (2.30) zu berechnen. Die diskrete Gleichung (2.32) kann als Fixpunktgleichung geschrieben werden

$$\begin{aligned} u_1 &= \frac{1}{2}\left(u_0 + u_2 + h^2\lambda\, e^{u_1}\right) \\ &\vdots \\ u_{n-1} &= \frac{1}{2}\left(u_{n-2} + u_n + h^2\lambda\, e^{u_{n-1}}\right) \end{aligned} \tag{2.39}$$

Im Jupyter-Notebook wird versucht, Lösungen mit Hilfe einer einfachen Fixpunkt Iteration zu berechnen. Wie man beobachten kann (Abb. 2.10), wird die Konvergenz mit feiner werdenden Diskretisierung immer schlechter. Mit Hilfe der formulierten Fixpunktiteration können nur die unteren Lösungen (vgl. Abb. 2.8 links) berechnet werden. Der obere Bereich ist in dieser Formulierung instabil und führt sofort zur Divergenz der Fixpunktiteration. ⌟

2.3.3 Das Newton-Verfahren für Systeme

Gegeben sei ein Gleichungssystem von n Gleichungen mit n Unbekannten $x_1, \ldots, x_n$:

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

Mit $\mathbf{x} = (x_1, \ldots, x_n)^T$ und $\mathbf{f} = (f_1, \ldots, f_n)^T$ folgt die kompakte Schreibweise des Gleichungssystems

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{2.40}$$

Sei $D \subset \mathbb{R}^n$ der Definitionsbereich von $\mathbf{f}$, daher $\mathbf{f} : D \to \mathbb{R}^n$ und $\mathbf{f}$ sei stetig differenzierbar.

Gesucht sind Punkte $\mathbf{x} \in D$, die die Gleichung (2.40) erfüllen. Hat man eine gute Näherhungslösung, so kann man mit dem Newton-Verfahren versuchen, beliebig genaue Lösungen zu berechnen. Das Newton-Verfahren für $\mathbf{f}(\mathbf{x}) = 0$ im $\mathbb{R}^n$ ist analog zum Newton-Verfahren für Funktionen einer Variablen.

Das Newton Verfahren basiert auf der Idee des iterativen Lösen der *linearisierten Gleichung* und ist für skalare Gleichungen $f(x) = 0$ gegeben durch

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}.$$

Da für Abbildungen $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ die Ableitung $\mathbf{f}'(\mathbf{x})$ durch die Jacobi-Matrix gegeben ist, kann nicht einfach dividiert werden. Die Idee bleibt jedoch dieselbe: Gesucht ist die Lösung der linearisierten Gleichung. Die Linearisierung von $\mathbf{f}$ im Punkt $\mathbf{x}_0$ ist gegeben durch

$$\boldsymbol{\tau}_{\mathbf{x}_0}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{f}'(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0).$$

Gesucht ist daher die Lösung des linearen Gleichungssystem $\boldsymbol{\tau}_{\mathbf{x}_0}(\mathbf{x}) = \mathbf{0}$

$$\mathbf{f}(\mathbf{x}_0) + \mathbf{f}'(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) = \mathbf{0}.$$

Unter der Annahme, dass $\mathbf{f}'(\mathbf{x}_0)$ regulär (dh. invertierbar) ist, folgt

$$\mathbf{x} = \mathbf{x}_0 - \mathbf{f}'(\mathbf{x}_0)^{-1} \cdot \mathbf{f}(\mathbf{x}_0).$$

Damit folgt die Rechenvorschrift des Newton-Verfahren

$$\begin{aligned} \mathbf{x}_0 \quad &\text{gegeben,} \\ \mathbf{x}_{n+1} &= \mathbf{x}_n - \mathbf{f}'(\mathbf{x}_n)^{-1} \cdot \mathbf{f}(\mathbf{x}_n). \end{aligned}$$

In der numerischen Umsetzung berechnet man **nicht** die Inverse der Jacobi-Matrix, sondern die Korrektur $\boldsymbol{\delta}_n$

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \underbrace{\mathbf{f}'(\mathbf{x}_n)^{-1} \cdot \mathbf{f}(\mathbf{x}_n)}_{=\boldsymbol{\delta}_n},$$

und aktualisiert damit den Lösungsvektor $\mathbf{x}_{n+1}$. Es folgt damit

$$\begin{aligned} \mathbf{f}'(\mathbf{x}_n) \cdot \boldsymbol{\delta}_n &= \mathbf{f}(\mathbf{x}_n) \\ \mathbf{x}_{n+1} &= \mathbf{x}_n - \boldsymbol{\delta}_n. \end{aligned}$$

Algorithmus 2.9 (Newton-Iteration).
`Gegeben: Startwert` $\mathbf{x}^0$.
`Für` $n = 0, 1, 2, \ldots$:

- `Berechne`

$$\mathbf{b}_n = \mathbf{f}(\mathbf{x}_n), \quad \boldsymbol{A}_n = \mathbf{f}'(\mathbf{x}_n)$$

- `Löse das lineare Gleichungssystem in` $\boldsymbol{\delta}_n$

$$\boldsymbol{A}_n \cdot \boldsymbol{\delta}_n = \mathbf{b}_n.$$

- `Setze (Newton Schritt)`

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \boldsymbol{\delta}_n.$$

Bemerkung 2.23. Das Newton-Verfahren für Systeme lässt sich als Fixpunktiteration

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k) \quad \text{mit} \quad \Phi(\mathbf{x}) = x - (\mathbf{f}'(\mathbf{x}))^{-1} \cdot \mathbf{f}(\mathbf{x})$$

auffassen. ⌟

Beispiel 2.22 (Jupyter-Notebook). Als Einführende Anwendung betrachten wir das Einstiegsbeispiel 2.17. Gesucht ist eine Lösung des Gleichungssystems (2.28), wir betrachten daher die Nullstellengleichung

$$\begin{pmatrix} \cos(x) + 2\,y - 6x \\ x\,y^2 + \sin(x) - 8y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

und definieren die linke Seite als

$$\begin{aligned} \mathbf{g} &: \mathbb{R}^2 \to \mathbb{R}^2 \\ (x, y) &\mapsto \mathbf{g}(x, y) = \begin{pmatrix} \cos(x) + 2\,y - 6x \\ x\,y^2 + \sin(x) - 8y \end{pmatrix}. \end{aligned}$$

Die zugehörige Jacobi-Matrix ist gegeben durch

$$\begin{aligned} \mathbf{g}' &: \mathbb{R}^2 \to \mathbb{R}^{2\times 2} \\ (x, y) &\mapsto \mathbf{g}'(x, y) = \begin{pmatrix} -\sin(x) - 6 & 2 \\ y^2 + \cos(x) & 2xy - 8 \end{pmatrix}. \end{aligned}$$

Für den Startwert $(0.5, 0.5)$ erhält man in 5 Iterationsschritte die Lösung $(0.1713, 0.02132)$ mit einem Residuum von 10^{-16}, sprich Maschinengenauigkeit. In der Abbildung 2.11 ist der Unterschied einer linearen (Fixpunkt Iteration) und quadratischen (Newton Verfahren) Fehlerordnung gut zu sehen.

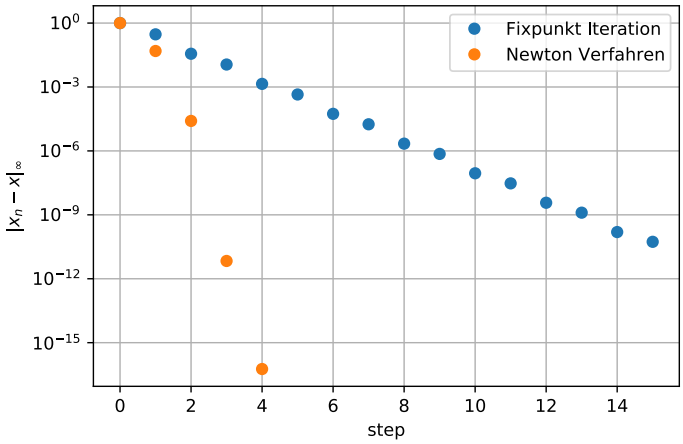


Abbildung 2.11: Vergleich Konvergenzordnung der Fixpunkt-Iteration und des Newton-Verfahren für das Einstiegsbeispiel 2.17

Das Schwierigkeit in der Anwendung des Newton-Verfahrens besteht in der Regel darin, gute Startwerte zu finden. Zweidimensionale Systeme können oft graphisch einfach visualisiert werden, wie im folgenden Beispiel illustriert wird.

Beispiel 2.23 (Jupyter-Notebook). Gesucht ist die Lösung des quadratischen Gleichungssystems

$$\begin{aligned} x^2 + 2y^2 &= 4 \\ 2x^2 + 2xy + 2x + 4(y-1)^2 &= 1. \end{aligned} \tag{2.41}$$

Um mit Hilfe des Newton-Verfahren eine Lösung des Systems numerisch berechnen zu können, muss das System als ein Nullstellenproblem geschrieben werden. Es folgt in dem Fall sehr leicht das Gleichungssystem

$$\mathbf{f}(x,y) = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

mit den beiden Abbildungen

$$\begin{aligned} f_1(x,y) &= x^2 + 2y^2 - 4 \\ f_2(x,y) &= 2x^2 + 2xy + 2x + 4(y-1)^2 - 1. \end{aligned} \tag{2.42}$$

Das System kann graphisch visualisiert werden, in dem die Contourgraphen der beiden Abbildungen f_1, f_2 für das Niveau 0 dargestellt wird.

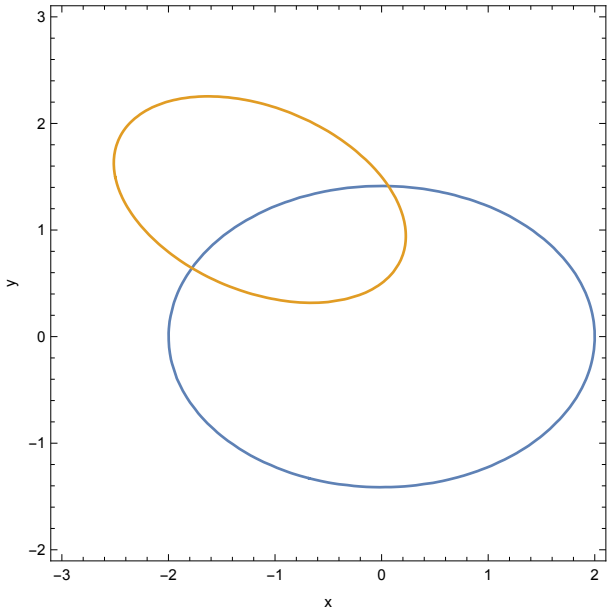


Abbildung 2.12: Niveaumenge der Abbildungen (2.42).

In der Abbildung 2.12 ist die Visualisierung zu sehen. Die beiden Schnittpunkte der Niveaulinien der beiden Abbildungen sind Lösung des Gleichungssystems (2.41). Mit der Jacobi-Matrix

$$\mathbf{f}'(x,y) = \begin{pmatrix} 2x & 2y \\ 4x+2y+2 & 2x+8(y-1) \end{pmatrix}$$

kann der Algorithmus 2.9 mit geeigneten Startwerte aus dem Graphen angewandt werden. Für den Startwert $(x_0, y_0) = (-1.5, 0.8)$ erhält man die Lösung $(-1.78112, 0.64328)$ und mit $(x_0, y_0) = (0.2, 1.2)$ folgt die Lösung $(0.063798, 1.41349)$. ⌟

Der Vollständigkeit halber und um die Voraussetzungen für die Konvergenz und das Konvergenzverhalten zu verdeutlichen, sei hier der ein Konvergenzsatz für das Newton-Verfahren erwähnt:

Satz 2.12. *Sei $D \subset \mathbb{R}^n$ offen und konvex, $\mathbf{f} : D \to \mathbb{R}^n$ eine stetig differenzierbare Funktion mit invertierbarer Jacobi-Matrix $\mathbf{f}'(\mathbf{x})$ für alle $\mathbf{x} \in D$. Sei β so, dass*

$$\|(\mathbf{f}'(\mathbf{x}))^{-1}\| \leq \beta \quad \forall \mathbf{x} \in D.$$

Ferner sei $\mathbf{f}'(\mathbf{x})$ auf D Lipschitz-stetig mit einer Konstanten γ, daher

$$\|\mathbf{f}'(\mathbf{x}) - \mathbf{f}'(\mathbf{y})\| \leq \gamma \|\mathbf{x} - \mathbf{y}\| \quad \mathbf{x}, \mathbf{y} \in D.$$

Weiterhin existiere eine Lösung $\mathbf{x}^* \in D$ von $\mathbf{f}(\mathbf{x}) = 0$. Der Startwert $\mathbf{x}_0$ erfülle $\mathbf{x}^0 \in B_\epsilon(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^n | \ \|\mathbf{x}^* - \mathbf{x}\| < \epsilon\}$ mit $\epsilon > 0$ hinreichend klein so, dass $B_\epsilon(\mathbf{x}^*) \subset D$ und

$$\epsilon \leq \frac{2}{\beta\,\gamma}.$$

Dann bleibt die durch das Newton-Verfahren definierte Folge $\{\mathbf{x}^k\}_{k=0}^{\infty}$ innerhalb $B_\epsilon(x^*)$

und konvergiert quadratisch gegen $\mathbf{x}^*$*:*

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq \frac{\beta\,\gamma}{2}\|\mathbf{x}_k - \mathbf{x}^*\|^2, \quad k = 0, 1, 2, \ldots.$$

Bemerkung 2.24. Bei geeigneten Startparametern konvergiert das Newton-Verfahren quadratisch. ⌟

Beispiel 2.24 (Jupyter-Notebook). Als abschliessendes Beispiel kommen wir wieder auf die Gelfand-Gleichung zurück. Wir berechnen nun mit Hilfe des Newton-Verfahrens die numerische Lösung der Nullstellengleichung (2.32) und definieren

$$\begin{aligned}\mathbf{g} &: \mathbb{R}^{n-1} \to \mathbb{R}^{n-1}\\ \mathbf{u} \mapsto \mathbf{g}(\mathbf{u}) &= \begin{pmatrix} -u_0 + 2\,u_1 - u_2 - h^2\lambda\, e^{u_1} \\ \vdots \\ -u_{n-2} + 2\,u_{n-1} - u_n - h^2\lambda\, e^{u_{n-1}} \end{pmatrix}.\end{aligned}$$

Da $u_0 = u_n = 0$ gegeben sind, benötigen wir die Jacobi-Matrix von $\mathbf{g}$ bezüglich den Variablen

$$\mathbf{u} = (u_1, \ldots, u_{n-1})^T.$$

Es folgt

$$\begin{aligned}\mathbf{g}' &: \mathbb{R}^{n-1} \to \mathbb{R}^{(n-1)\times(n-1)}\\ \mathbf{u} \mapsto \mathbf{g}'(\mathbf{u}) &= \begin{pmatrix} (2 - h^2\lambda\, e^{u_1}) & -1 & & 0 \\ -1 & & & \\ & & \ddots & \\ & & & -1 \\ 0 & & -1 & (2 - h^2\lambda\, e^{u_{n-1}}) \end{pmatrix}.\end{aligned}$$

Für unterschiedliche Startwerte können wir mit Hilfe des Newton-Verfahren beide Lösungen berechnen (Abb. 2.13). Die Konvergenz ist auch für h klein sehr gut. Mit ein paar wenigen (<5) Newton Iterationen erreicht man ein Residuum im Bereich der numerischen Rechenauflösung.

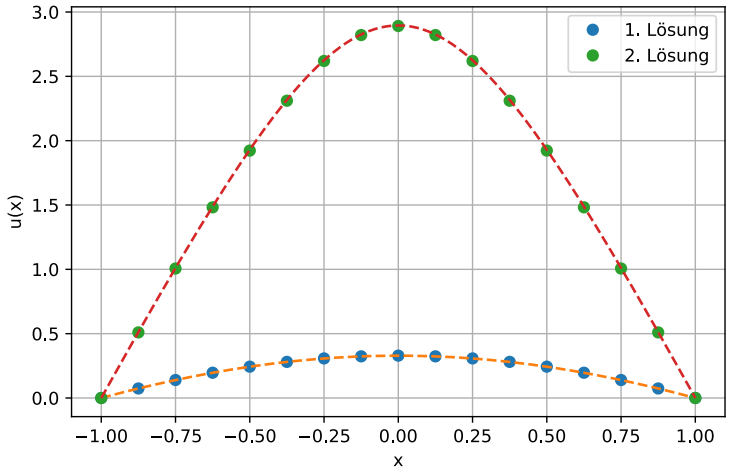


Abbildung 2.13: Numerische Lösungen für $\lambda = 1/2$ der Gelfand-Gleichung mit Hilfe des Newton-Verfahrens berechnet.

Das vereinfachte Newton-Verfahren

Das Berechnen der Jacobi-Matrix in jedem Iterationsschritt kann sehr numerisch teuer sein. Daher benutzt man oft auch das vereinfachte Newton-Verfahren:

Algorithmus 2.10 (Vereinfachtes Newton-Verfahren).
Gegeben: Startwert $\mathbf{x}_0$.
Berechne LU-Zerlegung von $\mathbf{f}'(x_0)$
Für $k = 0, 1, 2, \ldots$:

- Berechne $\mathbf{f}(\mathbf{x}_k)$
- Löse das lineare Gleichungssystem in $\mathbf{s}_k$

$$\mathbf{f}'(\mathbf{x}_0)\mathbf{s}_k = \mathbf{f}(\mathbf{x}_k).$$

- Setze (Newton Schritt)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{s}_k.$$

In dem Fall verwendet man die *numerische* Differentation

$$\frac{\partial f_i(x)}{\partial x_j} \approx \frac{f_i(x + h\, e^j) - f_i(x)}{h}, \quad \text{mit } e^j = (0, \ldots 0, 1, 0, \ldots, 0)^T.$$

Die numerische Berechnung der Jacobi-Matrix muss jedoch nicht immer numerisch effizienter sein.

Bemerkung 2.25. Typischerweise reduziert sich die Konvergenzgeschwindigkeiten der modifizierten Newton-Verfahren vgl. Abbildung 2.14.

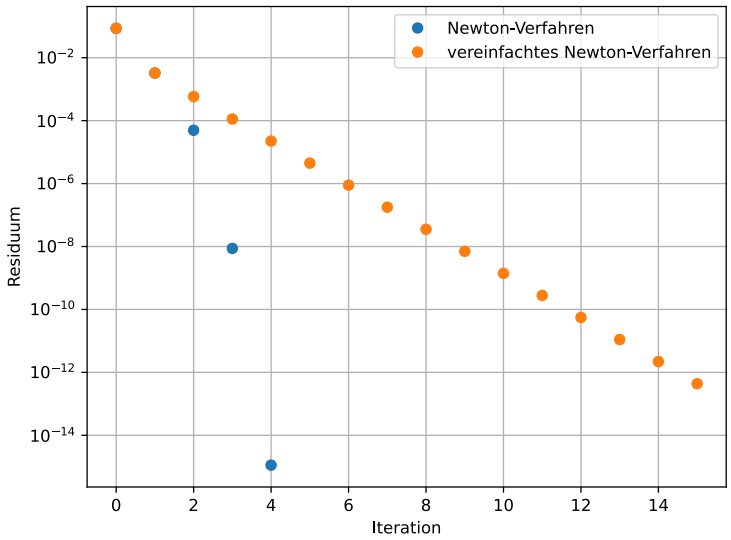


Abbildung 2.14: Vergleich der Konvergenzgeschwindigkeit für das Beispiel 2.24 (vgl. Jupyter-Notebook).

2.4 Nichtlineare Ausgleichsrechnung

2.4.1 Problemstellung

Wie im Kapitel 2.2 betrachten wir die Aufgabe, aus gegebenen Daten (Messungen) $b_i,\ i = 1,\dots,m,\ m > n$, auf eine von gewissen unbekannten Parametern $x_1,\dots,x_n$ abhängige Funktion

$$b(t) = y(t; x_1,\dots,x_n)$$

zu schliessen. Die Funktion $y(t; x_1,\dots,x_n)$ dient als Modell für den Zusammenhang der in den Messungen beobachtet wird. Die Parameter $x_i,\ 1,\dots,n$ sind so zu bestimmen, dass sie „optimal“ im Sinne der Gauss'schen Fehlerquadratmethode sind, daher

$$\sum_{i=1}^{m}(y(t_i, x_1^*,\dots,x_n^*) - b_i)^2 = \min_{x\in\mathbb{R}^n}\sum_{i=1}^{m}(y(t_i, x_1,\dots,x_n) - b_i)^2.$$

Falls die Parameter *linear* in y eingehen, so führt dies auf die lineare Ausgleichsrechnung. Hängt y von einigen Paramtern *nichtlinear* ab, so ergibt sich ein *nichtlineares Ausgleichsproblem*.

Beispiel 2.25. Elektromagnetische Schwingungen spielen eine zentrale Rolle in elektrischen Systemen. Jedes mechanische System unterliegt im Prinzip Schwingungsvorgängen, deren Verständnis schon im Hinblick auf Resonanzen enorm wichtig ist. Schwingungen sind aufgrund von Widerstands- bzw. Reibungseffekten in der Regel *gedämpft*. Für ein mechanisches System mit rückstellenden Kräften und Dämpfung lautet die entsprechende Differentialgleichung

$$\ddot{u} + \frac{b}{m}\dot{u} + \frac{D}{m}u = 0,$$

wobei m die Masse, D die Federkonstante und b eine Dämpfungskonstante sei. Lösungen dieser Differentialgleichung haben die Form

$$u(t) = u_0\, e^{-\tau t}\sin(\omega\, t + \varphi),$$

wobei u_0 eine Amplitude, τ die Abklingkonstante, ω die Kreisfrequenz und φ den Nullphasenwinkel bezeichnet. Das *Modell* sei also gegeben durch

$$y(t; x_1, x_2, x_3, x_4) = x_1\, e^{-x_2 t}\sin(x_3 t + x_4), \tag{2.43}$$

mit den Parametern $x_1 = u_0$, $x_2 = \tau$, $x_3 = \omega$ und $x_4 = \varphi$. Die Messdaten für b_i sin in Tabelle 2.4 gegeben.

t_i	0.1	0.3	0.7	1.2	1.6	2.2	2.7	3.1	3.5	3.9
b_i	0.558	0.569	0.176	-0.207	-0.133	0.132	0.055	-0.090	-0.069	0.027

Tabelle 2.4: Messungen b_i der Schwingung.

Der nach der Methode der kleinsten Fehlerquadrate zu minimierende Ausdruck ist

$$\sum_{i=1}^{10}\left(x_1\, e^{-x_2 t_i}\sin(x_3\, t_i + x_4) - b_i\right)^2 =: \|F(x_1, x_2, x_3, x_4)\|_2^2,$$

wobei $F : \mathbb{R}^4 \to \mathbb{R}^{10}$ durch

$$F_i(x_1, x_2, x_3, x_4) = x_1 e^{-x_2 t_i} \sin(x_3\, t_i + x_4) - b_i, \quad i = 1, \ldots, 10,$$

gegeben ist. ⌟

Definition 2.17 (Nichtlineare Ausgleichsproblem). Sei

$$F : \mathbb{R}^n \to \mathbb{R}^m, \quad F_i(x) := y(t_i, x) - b_i, \quad i = 1, \ldots, m,$$

mit $n \ll m$ gegeben, dann ist das *nichtlineare Ausgleichsproblem* gegeben durch: Bestimme $x^* \in \mathbb{R}^n$ so, dass

$$\|F(x^*)\|_2^2 = \min_{x \in \mathbb{R}^n} \|F(x)\|_2^2 \tag{2.44}$$

gilt. Oder mit der Funktion $\phi : \mathbb{R}^n \to \mathbb{R}$ gegeben durch

$$\phi(x) = \frac{1}{2} F(x)^T F(x)$$

äquivalent

$$\phi(x^*) = \min_{x \in \mathbb{R}^n} \phi(x).$$

Die Funktion ϕ hat im Punkt x^* ein lokales Minimum genau dann, wenn

1. der Gradient null ist, dh. $\nabla\phi(x^*) = 0$ gilt und
2. die Hess'sche Matrix $\phi''(x^*) \in \mathbb{R}^{n \times n}$ symmetrisch positiv definit ist.

Bezeichnet $F'(x) \in \mathbb{R}^{m \times n}$ die Jacobi-Matrix von F an der Stelle x

$$F'(x) = \left(\frac{\partial F_i(x)}{\partial x_j} \right)_{1 \le i \le m,\, 1 \le j \le n}$$

und $F_i''(x)$ die Hessesche Matrix von F_i,

$$F_i''(x) = \left(\frac{\partial^2 F_i(x)}{\partial x_j \partial x_k} \right)_{1 \le j,k \le n} \in \mathbb{R}^{n \times n},$$

dann lässt sich durch Nachrechnen bestätigen, dass

$$\nabla\phi(x) = F'(x)^T F(x) \tag{2.45a}$$

$$\phi''(x) = F'(x)^T F'(x) + \sum_{i=1}^{m} F_i(x) F_i''(x) \tag{2.45b}$$

gilt.

Beispiel 2.26. Für das Beispiel 2.25 folgt $F'(x)$

$$F'(x)^T = \begin{pmatrix} e^{-x_2 t_i} \sin(x_3 t_i + x_4) \\ -t_i\, x_1 e^{-x_2 t_i} \sin(x_3\, t_i + x_4) \\ t_i\, x_1 e^{-x_2 t_i} \cos(x_3\, t_i + x_4) \\ x_1 e^{-x_2 t_i} \cos(x_3\, t_i + x_4) \end{pmatrix}_{1 \le i \le 10} \in \mathbb{R}^{4 \times 10}$$

und somit

$$\nabla\phi(x) = \begin{pmatrix} e^{-x_2 t_1}\sin(x_3 t_1 + x_4) & \dots & e^{-x_2 t_{10}}\sin(x_3 t_{10} + x_4) \\ -t_1\, x_1 e^{-x_2 t_1}\sin(x_3\, t_1 + x_4) & \dots & -t_{10}\, x_1 e^{-x_2 t_{10}}\sin(x_3\, t_{10} + x_4) \\ t_1\, x_1 e^{-x_2 t_1}\cos(x_3\, t_1 + x_4) & \dots & t_{10}\, x_1 e^{-x_2 t_{10}}\cos(x_3\, t_{10} + x_4) \\ x_1 e^{-x_2 t_1}\cos(x_3\, t_1 + x_4) & \dots & x_1 e^{-x_2 t_{10}}\cos(x_3\, t_{10} + x_4) \end{pmatrix} \cdot \begin{pmatrix} x_1 e^{-x_2 t_1}\sin(x_3\, t_1 + x_4) - b_1 \\ \vdots \\ x_1 e^{-x_2 t_{10}}\sin(x_3\, t_{10} + x_4) - b_{10} \end{pmatrix}$$

2.4.2 Gauss-Newton-Verfahren

Sei $x_k \in \mathbb{R}^n$ eine bekannte Annäherung der gesuchten Lösung des nichtlinearen Ausgleichproblems (2.44). Die Idee besteht nun darin, ein Minimum der linearen Approximation mittels der Taylorentwicklung

$$F(x) = F(x_k) + F'(x_k)\underbrace{(x - x_k)}_{=s} + \mathcal{O}(\|x - x_k\|_2^2)$$

zu suchen:

Finde $s_k \in \mathbb{R}^n$ so, dass

$$\|F'(x_k)\cdot s_k + F(x_k)\|_2^2 = \min_{s\in\mathbb{R}^n} \|F'(x_k)\cdot s + F(x_k)\|_2^2 \tag{2.46}$$

und setze das neue x_{k+1}

$$x_{k+1} = x_k + s_k.$$

Bemerkung 2.26. Das lineare Ausgleichsproblem

$$\|F'(x_k)s_k + F(x_k)\|_2^2 = \min_{s\in\mathbb{R}^n} \|F'(x_k)\cdot s + F(x_k)\|_2^2$$

hat eine eindeutige Lösung $s_k \in \mathbb{R}^n$ nur dann, wenn die Matrix $F'(x_k)$ vollen Rang n hat. Falls sie das nicht hat ist nicht alles verloren, wir gehen aber an der Stelle grundsätzlich davon aus (vgl. [1] Abschnitt 4.7).

Algorithmus 2.11 (Gauss-Newton).
`Wähle Startwert` $x_0 \in \mathbb{R}^n$.
`Für` $k = 0, 1, 2, \dots$:
  `Berechne` $F(x_k)$, $F'(x_k)$;
  `Löse das lineare Ausgleichsproblem (2.46);`
  `Setze` $x_{k+1} = x_k + s_k$;

Als Abbruchkriterium für diese Methode wird häufig

$$\|F'(x_k)^T \cdot F(x_k)\|_2 \le \varepsilon$$

verwendet, wobei ε eine vorgegebene Toleranz sei. Die Idee besteht darin, dass von ϕ die Ableitung $\|\nabla\phi(x_k)\|_2 = \|F'(x_k)^T \cdot F(x_k)\|_2$ Null sein muss.

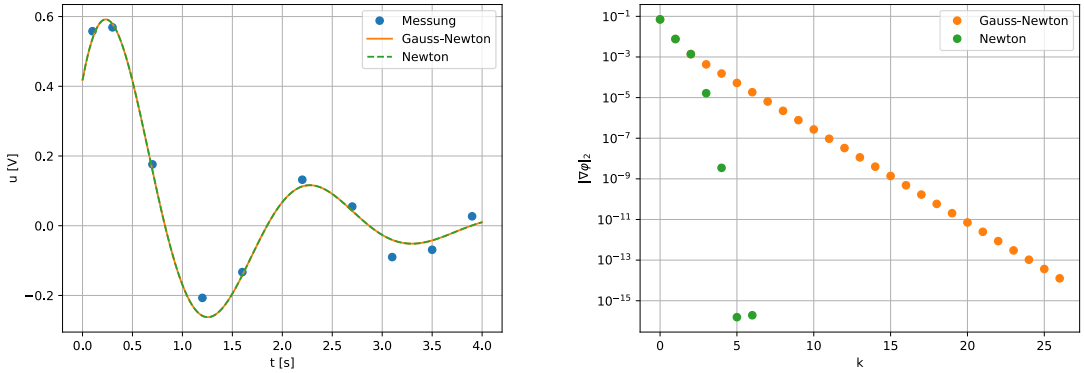
Bemerkung 2.27. Eine Alternative ist das *Newton-Verfahren* zu Bestimmung einer Nullstelle von $f(x) := \nabla\phi(x)$. Mit Hilfe von (2.45) ergibt dies die Iterationsvorschrift

$$\begin{aligned} x_{k+1} &= x_k - (\phi''(x_k))^{-1} \cdot \nabla\phi(x_k) \\ &= x_k - \left(F'(x_k)^T \cdot F'(x_k) + \sum_{i=1}^{m} F_i(x_k) F_i''(x_k) \right)^{-1} \cdot (F'(x_k)^T \cdot F(x_k)) \end{aligned}$$

Die Gauss-Newton-Methode kann als ein *modifiziertes* Newton-Verfahren zur Bestimmung einer Nullstelle von $\nabla\phi$ interpretiert werden, wobei in $\phi''(x^k)$ der Term mit den zweiten Ableitungen von F weggelassen wird. Wegen der Vernachlässigung dieses Terms geht die quadratische Konvergenz der Newton-Methode verloren.

Die Gauss-Newton-Iteration ist in der Regel nur linear konvergent. Der Rechenaufwand für die zweiten Ableitungen kann sehr gross sein, so dass das Gauss-Newton unter dem Strich schneller sein kann. ⌟

Beispiel 2.27 (Jupyter-Notebook). Fortsetzung des Beispiels 2.25. Die Gauss-Newton Methode wird mit dem Startwert $x^0 = (1, 1, 3, 1)^T$ auf das Problem angewandt. In der Abbildung 2.15 sind die Resultate dargestellt. Beim Newton-Verfahren wurde für die ersten drei Iterationsschritte das Gauss-Newton Verfahren durch weglassen der Summe $\sum_{i=1}^{m} F_i(x) F_i''(x)$ benutzt.



(a) Lösungen (b) Konvergenz

Abbildung 2.15: Lösung und Konvergenzgeschwindigkeiten mit $x_0 = (1, 1, 3, 1)^T$.

Mit dem Startwert $x_0 = (1, 1, 3, 1)^T$ erhalten wir die Lösung

$$u(t) \approx 0.735356 \cdot e^{-0.796202 \cdot t} \sin(3.074499 \cdot t + 0.604181).$$

⌟

Bemerkung 2.28. Der Konvergenzradius der Newton-Iteration ist in der Regel sehr klein. Daher werden bei der Anwendung der Newton-Methode die ersten Iterationsschritt oft mit einer Methode mit linearer Konvergenzrate (z.B. Gauss-Newton-Methode) durchgeführt und danach auf die Newton-Iteration zur Beschleunigung gewechselt. Zu dem wird der Newton-Schritt oft auch gedämpft

$$x_{k+1} = x_k - \delta_k s_k, \quad \text{mit } 0 < \delta_k \in \mathbb{R} \text{ so, dass } \phi(x_{k+1}) < \phi(x_k) \tag{2.47}$$

gilt. Die Gleichung (2.47) liefert im Algorithmus 2.9 eine zusätzliche innere Schleife. Eine mögliche Umsetzung ist mit dem Algorithmus 2.12 gegeben. Die Konvergenzgeschwindigkeit ist das eine, eine stabile numerische Methode das andere, oft wichtigere! ⌟

Algorithmus 2.12 (Gauss-Newton gedämpft).
`Wähle Startwert` $x_0 \in \mathbb{R}^n$.
`Für` $k = 0, 1, 2, \ldots$:
  `Berechne` $F(x_k)$, $F'(x_k)$;
  `Löse das lineare Ausgleichsproblem (2.46);`
  $\delta_k = 1$;
  `Solange` $\phi(x_k + \delta_k s_k) > \phi(x_k)$ `und` $\delta_k > \delta_{\min}$:
    $\delta_k = \delta_k/2$;
  `Setze` $x_{k+1} = x_k + \delta_k s_k$;

2.4.3 Levenberg-Marquardt-Verfahren

Beim Levenberg-Marquardt-Verfahren wird die Korrektur s_k durch folgende Minimierungsaufgabe festgelegt: Finde $s_k \in \mathbb{R}^n$ so, dass

$$\|F'(x_k) \cdot s_k + F(x_k)\|_2^2 + \mu^2 \|s_k\|_2^2 \to \min, \tag{2.48}$$

wobei $\mu > 0$ ein zu wählender Parameter ist. Als neue Näherung wird

$$x_{k+1} = x_k + s_k$$

gesetzt. Die Minimierungsaufgabe (2.48) ist äquivalent zu

Finde $s_k \in \mathbb{R}^n$ so, dass

$$\left\| \begin{pmatrix} F'(x_k) \\ \mu \mathbb{1} \end{pmatrix} \cdot s_k + \begin{pmatrix} F(x_k) \\ 0 \end{pmatrix} \right\|_2^2 \to \min . \tag{2.49}$$

Das folgt direkt aus

$$\left\| \begin{pmatrix} F'(x_k) \\ \mu \mathbb{1} \end{pmatrix} \cdot s_k + \begin{pmatrix} F(x_k) \\ 0 \end{pmatrix} \right\|_2^2 = \|F'(x_k) \cdot s_k + F(x_k)\|_2^2 + \mu^2 \|s_k\|_2^2.$$

Der grosse Vorteil des linearen Ausgleichproblem (2.49) im Vergleich zur Aufgabe (2.46) beim Gauss-Newton-Verfahren ist, dass für $\mu > 0$ die Matrix $\begin{pmatrix} F'(x_k) \\ \mu \mathbb{1} \end{pmatrix}$ stets vollen Rang hat. Die Minimierungsaufgabe hat daher immer eine eindeutige Lösung s^k.

Satz 2.13. *Sei $x_k \in \mathbb{R}^n$ mit $\nabla\phi(x_k) \neq 0$. Sei s_k die Levenberg-Marquardt-Korrektur aus (2.49). Für μ hinreichend gross gilt*

$$\phi(x_k + s_k) < \phi(x_k).$$

Bemerkung 2.29. Daher ist die Folge $\{x_k\}_{k=0,1,2,\ldots}$ für ϕ eine minimierende Folge. Um die Konvergenz zu gewährleisten, muss μ hinreichend gross gewählt werden. Dies kann dazu führen, dass die Konvergenz sehr langsam wird. In der Praxis werden heuristische Kriterien für die Wahl von μ benutzt. ⌟

Wahl von μ

Ein mögliches Verfahren für die Wahl von μ ist die folgende sogenannte *Trust Region* Methode: Sei

$$\rho_\mu := \frac{\|F(x_k)\|_2^2 - \|F(x_k + s_k)\|_2^2}{\|F(x_k)\|_2^2 - \|F(x_k) + F'(x_k) \cdot s_k\|_2^2} =: \frac{\Delta R(x_k, s_k)}{\Delta \tilde{R}(x_k, s_k)}.$$

und $0 < \beta_0 < \beta_1 < 1$ (z.B. $\beta_0 = 0.2$, $\beta_1 = 0.8$).

$\rho_\mu \leq \beta_0$:	s_k wird nicht akzeptiert, μ wird vergrössert und eine neue Korrektur s_k berechnet.
$\beta_0 < \rho_\mu < \beta_1$:	s_k wird akzeptiert, bei der Berechnung von s_{k+1} wird als Anfangswert dasselbe μ verwendet.
$\beta_1 \leq \rho_\mu$:	s_k wird akzeptiert, bei der Berechnung von s_{k+1} wird μ jedoch verkleinert.

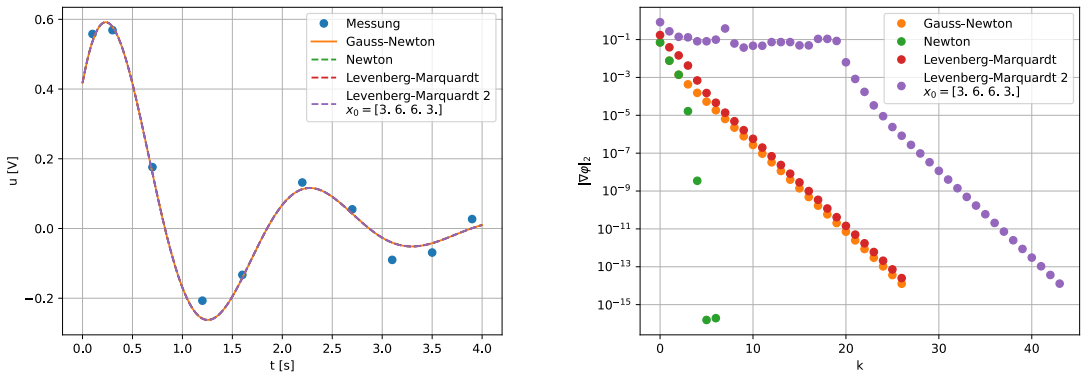
Algorithmus 2.13 (Levenberg-Marquardt).
Wähle Startwert x_0 und Anfangswert für den Parameter μ;
Für $k = 0, 1, 2, \ldots$:

1. Berechne $F(x_k)$, $F'(x_k)$;
2. Löse das lineare Ausgleichproblem

$$\left\| \begin{pmatrix} F'(x_k) \\ \mu \mathbb{1} \end{pmatrix} \cdot s_k + \begin{pmatrix} F(x_k) \\ 0 \end{pmatrix} \right\|_2^2 \to \min$$

3. Teste, ob Korrektur s_k akzeptabel ist?
   Nein: μ anpassen und Schritt 2 wiederholen;
4. $x_{k+1} = x_k + s_k$.

Beispiel 2.28. In der Abbildung 2.16 sind die Resultate der Levenberg-Marquardt Methode angewandt auf das Beispiel 2.25 zu sehen. Die Konvergenzgeschwindigkeit des Levenberg-Marquardt-Verfahren ist wie beim Gauss-Newton-Verfahren linear. ⌟



(a) Lösungen (b) Konvergenz

Abbildung 2.16: Konvergenzvergleich für das Beispiel 2.25. Mit Rot markiert ist eine zusätzliche Levenberg-Marquardt Folge für den Startwert $3 \cdot x_0$. Für diesen Startwert konvergieren sowohl das Gauss-Newton Verfahren wie auch die Newton Verfahren nicht. In der Praxis kann Aufgrund der beschränkten Rechenauflösung die Konvergenz der Levenberg-Marquardt-Methode nicht für einen beliebigen Startwert erwartet werden.

Kapitel 3

Einführung numerische Verfahren für Anfangswertprobleme

3.1 Einleitung

Häufig lässt sich für das Anfangswertprobleme

$$\begin{aligned} y'(x) &= f(x,y) \quad x \in [a,b] \subset \mathbb{R}_0^+ \\ y(x_0) &= y_0 \end{aligned} \tag{3.1}$$

mit einer gegebenen Funktion $f : [a,b] \times \mathbb{R} \to \mathbb{R}$ und gegebenem Anfangswert $y_0 \in \mathbb{R}$ keine analytische Lösung herleiten, wohingegen die Existenz einer Lösung durch den Satz von Picard-Lindelöf [vgl. Analysis 3] gegeben ist. Das folgende Kapitel soll eine Übersicht numerischer Methoden zum Lösen von Anfangswertprobleme erster Ordnung geben.

Wir betrachten im Folgenden eine äquidistante Unterteilung des Zeitintervalls $[a,b]$ gemäss

$$a = x_0 < x_1 < \ldots < x_n = b$$

mit

$$x_k = x_0 + k \cdot h, \quad h = \frac{b-a}{n}, \quad k = 0, \ldots, n.$$

Ausgehend von

$$y_0 = y(x_0)$$

werden sukzessive Näherungen

$$y_k \quad \text{an } y(x_k), \quad k = 1, 2, \ldots, n$$

bestimmt.

Zur Herleitung numerischer Verfahren werden zwei grundlegend verschiedene Ansätze betrachtet und unterscheiden daher in Integrations- und Differenzenmethoden. Die Klassifizierung ist nicht strikt, da Verfahren existieren, die sowohl auf der Basis einer numerischen Quadratur als auch auf der Grundlage einer Differenzenbildung hergeleitet werden können.

Integrationsmethoden Bei derartigen Algorithmen wird eine Integration der Differentialgleichung (3.1) beginnend bei $[x_0, x_{k+m}]$, $m \in \{1, \ldots, n\}$ sukzessive über jedes Teilintervall

$$[x_k, x_{k+m}], \quad k = 0, \ldots, n-m$$

vorgenommen. Mit

$$y(x_{k+m}) - y(x_k) = \int_{x_k}^{x_{k+m}} y'(s)ds = \int_{x_k}^{x_{k+m}} f(s, y(s))ds$$

ergibt sich das Verfahren durch Verwendung einer numerischen Quadraturformel für das Integral über f.

Beispiel 3.1. Für $m = 1$ kann das Integral zum Beispiel durch

$$\int_{x_k}^{x_{k+m}} f(s, y(s))ds \approx h\, f(x_k, y_k)$$

approximiert werden. Es folgt das *explizite Euler Verfahren*

Explizites Euler Verfahren

$$y_{k+1} = y_k + h \cdot f(x_k, y_k). \tag{3.2}$$

Differenzenmethoden Bei den Differenzenmethoden wird die Differentialgleichung zu einem Zeitpunkt x_k betrachtet und der vorliegende Differentialquotient durch einen geeigneten Differenzenquotienten approximiert. Mit

$$y'(x_k) \approx \frac{y(x_{k+1}) - y(x_k)}{h}$$

ergibt sich durch Einsetzen in die Differentialgleichung die Darstellung

$$\frac{y_{k+1} - y_k}{h} = f(x_k, y_k)$$

und folglich wiederum das explizite Euler Verfahren

$$y_{k+1} = y_k + h\, f(x_k, y_k) \quad , k = 0, \ldots, n-1.$$

3.2 Einschrittverfahren

3.2.1 Einführung

Das bereits vorgestellte Euler Verfahren gehört in die Gruppe der sogenannten Einschrittverfahren. Diese Verfahrensklasse ist dadurch charakterisiert, dass zur Berechnung der Grösse y_{k+1} stets die Verfügbarkeit des letzten Näherungswertes y_k ausreichend ist.

Definition 3.1 (Einschrittverfahren). Lässt sich eine Methode unter Verwendung einer *Verfahrensfunktion*

$$\phi : [a, b] \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}_0^+ \to \mathbb{R}$$

in der Form

$$y_{k+1} = y_k + h \cdot \phi(x_k, y_k, y_{k+1}, h) \tag{3.3}$$

schreiben, so sprechen wir von einem *Einschrittverfahren*. Dabei wird zwischen

- *impliziten* Verfahren, falls $\phi = \phi(x_k, y_k, y_{k+1}, h)$ und
- *expliziten* Verfahren, falls $\phi = \phi(x_k, y_k, h)$

unterschieden.

Nutzen wir im Kontext der Integrationsmethoden mit $m = 1$ das Polynom $q \in \Pi_0$, $q(x_{k+1}) = f(x_{k+1}, y(x_{k+1}))$ und wird die Rechteckregel unter Auswertung des Integranden am rechten Rand des Intervalls $[x_k, x_{k+1}]$ eingesetzt, so erhalten wir

Implizites Euler Verfahren

$$y_{k+1} = y_k + h \cdot f(x_{k+1}, y_{k+1}). \tag{3.4}$$

Die zugehörige Verfahrensfunktion lautet folglich

$$\phi(x_k, y_k, y_{k+1}, h) = f(x_{k+1}, y_{k+1}),$$

weshalb vom *impliziten Euler Verfahren* gesprochen wird, welches analog zur expliziten Variante ein Einschrittverfahren repräsentiert.

Beispiel 3.2 (Jupyter-Notebook). Wir betrachten das Beispiel 4.10 aus der Analysis 2 Vorlesung. Gegeben sei ein R-L-Schwingkreis (vgl. Abb. 3.1).

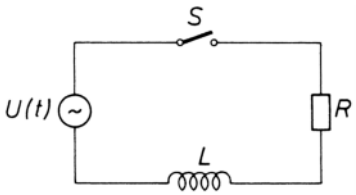


Abbildung 3.1: R-L-Schwingkreis

Das dazu gehörige Anfangswertproblem ist gegeben durch

$$\begin{aligned} \frac{di}{dt} &= -\frac{R}{L} i + \frac{U_0}{L} \sin \omega t \\ i(0) &= 0. \end{aligned}$$

Für die analytische Lösung folgt

$$i(t) = \frac{U_0}{R^2 + \omega^2 L^2} \left(\omega L e^{-\frac{R}{L} t} + R \sin \omega t - \omega L \cos \omega t \right), \quad t > 0.$$

Die Funktion $f : [a, b] \times \mathbb{R} \to \mathbb{R}$ ist gegeben durch

$$f(t, i) = -\frac{R}{L} i + \frac{U_0}{L} \sin(\omega t).$$

Entsprechend folgt für das explizite bzw. implizite Euler Verfahren

1. Explizites Euler Verfahren

$$i_{k+1} = i_k + h \cdot \left(-\frac{R}{L} i_k + \frac{U_0}{L} \sin(\omega k\, h) \right)$$

2. Implizites Euler Verfahren

$$i_{k+1} = i_k + h \cdot \left(-\frac{R}{L} i_{k+1} + \frac{U_0}{L} \sin(\omega (k+1)\, h) \right)$$

Anwendung der Euler Verfahren auf ein Modellproblem

Zur Untersuchung von Stabilitätseigenschaften ist das Modellproblem

$$y'(x) = \lambda\, y(x), \quad y(0) = 1, \quad \lambda \in \mathbb{R}$$

sehr geeignet. Es hat die exakte Lösung

$$y(x) = e^{\lambda x}.$$

Über den Parameter λ kann das Lösungsverhalten gesteuert werden, für $\lambda < 0$ ergeben sich abklingende, für $\lambda > 0$ anwachsende Lösungen. Der zweite Vorteil dieses einfachen Testproblems ist, dass für viele Verfahren die Näherungslösungen in ihrer Abhängigkeit von λ und h exakt dargestellt werden können und damit das Verhalten der Verfahren für unterschiedliche Parameterwerte untersucht werden kann. (Siehe auch Jupyter-Notebook).

Das *explizite* Euler Verfahren (3.2) ergibt die Näherungswerte

$$\begin{aligned} u_0 &= 1 \\ u_1 &= 1 + \lambda\, h = e^{\lambda h} + C\, h \\ u_2 &= (1 + \lambda\, h)^2 \\ &\vdots \\ u_j &= (1 + \lambda\, h)^j \end{aligned}$$

Damit folgt für die Konvergenz des Verfahrens: Für ein festes $\bar{x} \in \mathbb{R}$ lassen wir $h \to 0$ und $j \to \infty$ streben, so dass immer $\bar{x} = j \cdot h$ gilt, also $h = \frac{\bar{x}}{j}$. Dann gilt für den Näherungswert u_j von $y(\bar{x})$

$$u_j = u_j(h) = \left(1 + \lambda \frac{\bar{x}}{j} \right)^j \to e^{\lambda \bar{x}} \quad \text{mit} \quad j \to \infty.$$

Wird ein festes $h > 0$ benutzt, muss dies jedoch nicht zwingend der Fall sein. Ist $\lambda < 0$ so stark negativ, dass $|1 + \lambda h| > 1$ gilt, so wächst $|u_j|$ oszillatorisch, während die exakte Lösung gegen Null abklingt. Das Verfahren wird daher numerisch instabil.

Das *implizite* Euler Verfahren (3.4) ergibt

$$\begin{aligned} u_0 &= 1 \\ u_1 &= 1 + \lambda h u_1 \Rightarrow \; u_1 = \frac{1}{1 - \lambda h} \\ u_2 &= \frac{1}{(1 - \lambda h)^2} \\ &\vdots \\ u_j &= \frac{1}{(1 - \lambda h)^j} \end{aligned}$$

Hier liegt Konvergenz in der gleichen Güte wie beim expliziten Euler Verfahren vor. Darüber hinaus ist aber die numerische Lösung für endliches $h > 0$ immer stabil, da im wichtigen Fall $\lambda < 0$ mit einer abklingenden Lösungsfunktion auch die Werte u_j abklingen. Dieses von h unabhängige Stabilitätsverhalten nennt man *absolute Stabilität*.

3.2.2 Diskretisierungsfehler, Konvergenz

Zum Vergleich der Güte unterschiedlicher Algorithmen ist der *lokale Diskretisierungsfehler* von zentraler Bedeutung.

Definition 3.2 (lokaler Diskretisierungsfehler). Sei $y : [a, b] \to \mathbb{R}$ Lösung der Differentialgleichung (3.1), dann heisst

$$\eta(x, h) := y(x) + h\,\phi(x, y(x), y(x + h), h) - y(x + h) \tag{3.5}$$

für $x \in [a, b]$, $0 \le h \le b - x$, *lokaler Diskretisierungsfehler* der zur Verfahrensfunktion ϕ gehörigen Einschrittmethode zum Zeitpunkt $x + h$ bezüglich der Schrittweite h.

Der lokale Diskretisierungsfehler beschreibt die innerhalb eines Zeitschrittes entstehende Abweichung von der Lösungskurve. Obwohl innerhalb eines Zeitschrittverfahrens der Wert y_1 üblicherweise nicht mehr auf der Lösungskurve des zugrundeliegenden Anfangswertproblems liegt und folglich eine explizite Auswertung der lokalen Fehlerentwicklung auch bei Kenntnis der Lösung y nicht möglich ist, erweist sich diese lokale Fehlerbetrachtung als wesentlich für die folgende Ordnungsanalyse des *globalen Gesamtfehlers* (vgl. Abbildung 3.2a).

Definition 3.3 (Konsistenz). Ein Einschrittverfahren heisst *konsistent von der Ordnung* $p \in \mathbb{N}$ zur Differentialgleichung (3.1), wenn

$$\eta(x, h) \le C\,h^{p+1}, \quad h \to 0$$

für alle $x \in [a, b]$, $0 \le h \le b - x$, gilt. Im Fall $p = 1$ spricht man auch einfach von *Konsistenz*.

Bemerkung 3.1. Konsistenz bedeutet insbesondere

$$\lim_{h\to 0}\phi(x,y(x),y(x+h),h) = \lim_{h\to 0}\frac{y(x+h)-y(x)}{h} - \underbrace{\lim_{h\to 0}\frac{\eta(x,h)}{h}}_{=0}$$
$$= y'(x) = f(x,y(x))$$

für alle $x \in [a,b)$. Folglich spiegelt diese Eigenschaft eine hinreichende Bedingung dafür wider, dass die Verfahrensfunktion sinnvoll in Bezug auf die Differentialgleichung gewählt wurde.

Satz 3.1. *Die Euler Verfahren (3.2) und (3.4) sind konsistent zur Differentialgleichung (3.1) erster Ordnung.*

Beweis. Wir beschränken uns auf das explizite Euler Verfahren. Der Beweis für das implizite funktioniert analog.

Aus einer Taylor-Entwicklung folgt für eine zweimal stetig differenzierbare Lösung y die Darstellung

$$y(x+h) = y(x) + h\,y'(x) + \frac{h^2}{2}y''(\xi)$$
$$= y(x) + h\,f(x,y(x)) + \frac{h^2}{2}y''(\xi)$$

für ein $\xi \in [x, x+h]$. Einsetzen in die Definition des lokalen Verfahrensfehlers liefert für das explizite Euler Verfahren mit

$$\phi(x,y(x),y(x+h),h) = f(x,y(x))$$

den Nachweis durch

$$\eta(x,h) = y(x) + h\,f(x,y(x)) - y(x+h)$$
$$= \frac{h^2}{2}y''(\xi) = C\,h^2, \quad h\to 0.$$

□

Bei der Nutzung numerischer Verfahren ist der Anwender üblicherweise nicht vorrangig an dem lokalen Fehlerterm, sondern an dem Gesamtfehler zu einem gegebenen Zeitpunkt interessiert.

Definition 3.4 (globaler Diskretisierungsfehler). Sei $y : [a,b] \to \mathbb{R}$ Lösung der Differentialgleichung (3.1) und y_k der durch ein Einschrittverfahren mit der Schrittweite h erzeugte Näherungswert an $y(x_k)$, $x_k = a + k\,h \in [a,b]$, dann heisst

$$e(x_k,h) := y(x_k) - y_k \tag{3.6}$$

der *globale Diskretisierungsfehler* zum Zeitpunkt x_k.

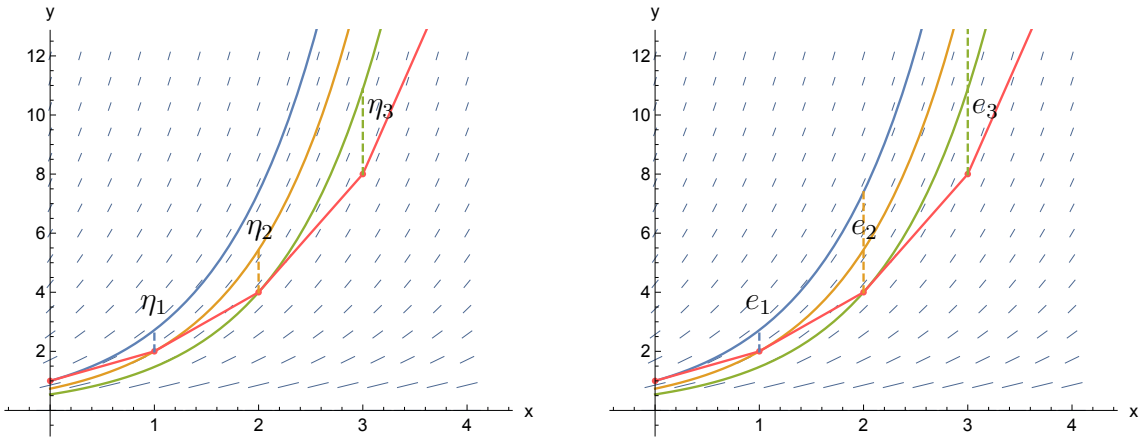
Definition 3.5 (Konvergenz). Ein Einschrittverfahren heisst *konvergent von der Ordnung* $p \in \mathbb{N}$ zum Anfangswertproblem (3.1), wenn

$$e(x_k, h) \leq C\, h^p, \quad h \to 0$$

für alle x_k gilt. Im Fall

$$\lim_{h \to 0} e(x_k, h) = 0$$

für alle x_k spricht man auch von *Konvergenz*.



(a) lokaler (b) globaler

Abbildung 3.2: Diskretisierungsfehler (Jupyter-Notebook)

Satz 3.2 (Konvergenz). *Sei ϕ die Verfahrensfunktion eines Einschrittverfahrens zur Lösung des Anfangswertproblems (3.1). Genügt*

$$\phi = \phi(x, y, \tilde{y}, h)$$

bezüglich der zweiten und dritten Komponente einer Lipschitzbedingung gemäss

$$|\phi(x, u, \tilde{y}, h) - \phi(x, v, \tilde{y}, h)| \leq L_1 |u - v|$$

und

$$|\phi(x, y, u, h) - \phi(x, y, v, h)| \leq L_2 |u - v|$$

mit $L_1, L_2 > 0$, dann gilt mit

$$L = \max\{L_1, L_2\} \quad \textit{und} \quad \eta(h) = \max_{x \in [a,b]} |\eta(x, h)|$$

für den globalen Diskretisierungsfehler zum Zeitpunkt x_k im Fall

$$h < \frac{1}{L}$$

die Abschätzung

$$|e(x_k,h)| \le \left(|e(x_0,h)| + \frac{(x_k-x_0)}{1-h\,L}\frac{\eta(h)}{h}\right) e^{2\frac{x_k-x_0}{1-h\,L}L} \tag{3.7}$$

für $k = 0, 1, \ldots n$.

Beweis. Aus der Definition des lokalen Diskretisierungsfehlers ergibt sich

$$y(x_{k+1}) = y(x_k) + h\,\phi(x_k, y(x_k), y(x_{k+1}), h) - \eta(x_k,h),$$

womit wir direkt die Darstellung des globalen Diskretisierungsfehlers in der Form

$$\begin{aligned}
e(x_{k+1},h) &= y(x_{k+1}) - y_{k+1} \\
&= y(x_k) + h\,\phi(x_k, y(x_k), y(x_{k+1}), h) - \eta(x_k,h) - y_{k+1} \\
&= y(x_k) + h\,\phi(x_k, y(x_k), y(x_{k+1}), h) - \eta(x_k,h) \\
&\quad - y_k - h\,\phi(x_k, y_k, y_{k+1}, h) \\
&= e(x_k,h) - \eta(x_k,h) \\
&\quad + h\,(\underbrace{\phi(x_k, y(x_k), y(x_{k+1}), h) - \phi(x_k, y_k, y(x_{k+1}), h)}_{\le L_1|y(x_k)-y_k|} \\
&\qquad \underbrace{\phi(x_k, y_k, y(x_{k+1}), h) - \phi(x_k, y_k, y_{k+1}, h)}_{\le L_2|y(x_{k+1})-y_{k+1}|})
\end{aligned}$$

erhalten. Unter Verwendung der Lipschitz-Bedingung folgt

$$|e(x_{k+1},h)| \le |e(x_k,h)| + h\,L(|e(x_k,h)| + |e(x_{k+1},h)|) + |\eta(x_k,h)|.$$

Da $h < \frac{1}{L} \;\Rightarrow\; 1 - h\,L > 0$ und somit

$$|e(x_{k+1},h)| \le \underbrace{\left(1 + \frac{2h\,L}{1-h\,L}\right)}_{\ge 0} |e(x_k,h)| + \underbrace{\frac{1}{1-h\,L}}_{\ge 0}\underbrace{|\eta(x_k,h)|}_{\le \eta(h)}$$

für $k = 0, \ldots, n-1$. Sukzessives Anwenden der Ungleichung führt zu

$$\begin{aligned}
|e(x_{k+1},h)| &\le \left(1 + \frac{2\,h\,L}{1-h\,L}\right)^k |e(x_0,h)| + \sum_{j=0}^{k-1}\left(1 + \frac{2\,h\,L}{1-h\,L}\right)^{k-1} \frac{\eta(h)}{1-h\,L} \\
&\le \left(1 + \frac{2\,h\,L}{1-h\,L}\right)^k \left(|e(x_0,h)| + \frac{k\,\eta(h)}{1-h\,L}\right)
\end{aligned}$$

Mit $k = \frac{x_k-x_0}{h}$ folgt

$$|e(x_{k+1},h)| \le \underbrace{\left(1 + \frac{2h\,L}{1-h\,L}\right)^{\frac{x_k-x_0}{h}}}_{\le e^{2\frac{x_k-x_0}{1-h\,L}L}} \left(|e(x_0,h)| + \frac{x_k-x_0}{1-h\,L}\cdot\frac{\eta(h)}{h}\right).$$

Per Induktion kann der letzte Schritt nachgewiesen werden. □

Bemerkung 3.2. Bei einem expliziten Einschrittverfahren vereinfacht sich die Abschätzung (3.7) zu

$$|e(x_k,h)| \leq \left(|e(x_0,h)| + (x_k - x_0)\frac{\eta(h)}{h} \right) e^{(x_k-x_0)L}.$$

Es ist zu sehen, dass die rechte Seite sowohl bezüglich des Zeitpunktes x_k wie auch bezüglich der Lipschitz-Konstanten L exponentiell wächst.

In der Bemerkung 3.1 wurde der Zusammenhang zwischen der Verfahrensfunktion ϕ und der rechten Seite der Differentialgleichung aufgezeigt. Es ist ersichtlich, dass sich eine grosse minimale Lipschitz-Konstante L_f mit

$$|f(x,u(x)) - f(x,v(x))| \leq L_f |u(x) - v(x)|$$

in einer grossen Lipschitz-Konstanten L der Verfahrensfunktion ϕ widerspiegelt. Derartige Differentialgleichungen werden als *steif* bezeichnet. ⌟

Korrolar 3.1. *Ist ein Einschrittverfahren konsistent zu Differentialgleichung (3.1) von der Ordnung p und stimmt der Startwert der Methode mit dem Anfangswert überein, so ist das Verfahren konvergent von der Ordnung p. Kurz:*

$$\textit{Konsistenz } p\textit{-ter Ordnung} \quad \Rightarrow \quad \textit{Konvergenz } p\textit{-ter Ordnung}$$

3.2.3 Verfahren nach Runge

Um die Genauigkeit des expliziten Euler Verfahrens zu verbessern, betrachten wir das Verhalten des Abschneidefehlers bei zwei unterschiedlichen Zeitschritten. Sei $y_k = y(x_k)$ und $h > 0$ gegeben, dann folgt

$$y_{k+1} = y_k + h\, f(x_k, y_k)$$

bzw. für die halbe Schrittweite $h/2$

$$\begin{aligned} \tilde{y}_{k+1/2} &= \tilde{y}_k + \frac{h}{2} f(x_k, \tilde{y}_k) \\ \tilde{y}_{k+1} &= \tilde{y}_{k+1/2} + \frac{h}{2} f(x_{k+1/2}, \tilde{y}_{k+1/2}) \\ &= y_k + \frac{h}{2} \cdot \left(f(x_k,y_k) + f\left(x_{k+1/2}, y_k + \frac{h}{2} f(x_k,y_k) \right) \right) \end{aligned} \tag{3.8}$$

Eine Taylorentwicklung im Punkt x_k der Lösungsfunktion $y(x)$ liefert

$$\begin{aligned} y(x_{k+1}) &= y(x_k) + h \underbrace{y'(x_k)}_{=f(x_k,y(x_k))} + \frac{h^2}{2} \underbrace{y''(x_k)}_{=\frac{d}{dx} f(x_k,y(x_k))} + C\,h^3 \\ &= y(x_k) + hf(x_k,y(x_k)) + \frac{h^2}{2}(\partial_x f(x_k,y(x_k)) + \partial_y f(x_k,y(x_k)) \cdot \underbrace{y'(x_k)}_{=f(x_k,y(x_k))}) + C\,h^3 \\ &= y(x_k) + hf(x_k,y(x_k)) + \frac{h^2}{2}(\partial_x f(x_k,y(x_k)) + \partial_y f(x_k,y(x_k)) \cdot f(x_k,y(x_k))) + C\,h^3. \end{aligned}$$

Somit ergibt sich für die Näherungslösung y_{k+1} die Abweichung zur Lösung in der Form

$$y(x_{k+1}) - y_{k+1} = \frac{h^2}{2}(f_x + f_y \cdot f)(x_k, y(x_k)) + C \cdot h^3. \tag{3.9}$$

Analog erhalten wir unter Berücksichtigung von

$$f\left(x_k + \frac{h}{2}, y(x_k) + \frac{h}{2} f(x_k, y(x_k))\right) = f(x_k, y(x_k)) + \frac{h}{2}(f_x + f_y \cdot f)(x_k, y(x_k)) + C \cdot h^2$$

für $\tilde{y}_{k+1}$ gemäss (3.8) den Fehler

$$y(x_{k+1}) - \tilde{y}_{k+1} = \frac{h^2}{4}(f_x + f_y \cdot f)(x_k, y(x_k)) + C \cdot h^3. \tag{3.10}$$

Mittels einer gewichteten Kombination der Verfahren werde wir die quadratischen Anteile des Abschneidefehlers eliminieren und hierdurch ein konsistentes Verfahren zweiter Ordnung erzeugen. Motiviert durch die Fehlerdarstellungen (3.9) und (3.10) nutzen wir

$$2\tilde{y}_{k+1} - y_{k+1} = y(x_{k+1}) + C \cdot h^3,$$

wodurch sich das sogenannte verbesserte Polygonzugverfahren

$$\begin{aligned} y_{k+1/2} &= y_k + \frac{h}{2} f(x_k, y_k) \\ y_{k+1} &= 2\tilde{y}_{k+1} - y_{k+1} \\ &= y_k + h\, f(x_k + h/2, y_k + y_{k+1/2}) \end{aligned}$$

ergibt. Diese auch nach ihrem Erfinder Carl Runge (1905) benannte Methode kann unter Verwendung der Steigungen $r_k, k = 1, 2$ in der Form

Runge-Verfahren

$$\begin{aligned} r_1 &= f(x_k, y_k) \\ r_2 &= f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} r_1\right) \\ y_{k+1} &= y_k + h\, r_2 \end{aligned} \tag{3.11}$$

geschrieben werden. Aus der Herleitung folgt der Satz

Satz 3.3. *Das* Runge-Verfahren *(3.11) besitzt die Verfahrensfunktion*

$$\phi(x_k, y_k, h) = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} f(x_k, y_k)\right)$$

und ist konsistent zur Differentialgleichung (3.1) von zweiter Ordnung.

Bemerkung 3.3. Verwendung der Trapezregel im Rahmen der Integration liefert

$$\begin{aligned} y(x_k + h) &= y(x_k) + \int_{x_k}^{x_k+h} f(x, y(x))dx \\ &= y(x_k) + \frac{h}{2} \cdot \Big(f(x_k, y(x_k)) + f(x_k + h, y(x_k + h))\Big) + C\, h^3, \end{aligned}$$

so dass unter Berücksichtigung von $x_{k+1} = x_k + h$ mit

$$y_{k+1} = y_k + \frac{h}{2} \cdot (f(x_k, y_k) + f(x_{k+1}, y_{k+1})) \tag{3.12}$$

die *implizite Trapezmethode* zweiter Ordnung vorliegt.

3.2.4 Runge-Kutta-Verfahren, Butcher Schema

Das Runge-Verfahren kann auch als Integrationsmethode hergeleitet werden. Der Einsatz verbesserter Quadraturformeln (vgl. Jupyter-Notebook) in Verbindung mit einer zusätzlichen numerischen Integration zur Approximation der benötigten Lösungswerte an Zwischenstellen kann auf Verfahren höherer Ordnung führen.

Im Abschnitt wird eine verallgemeinerte Vorgehensweise dargestellt, die auf die Klasse der Runge-Kutta-Verfahren führt. Neben der Herleitung einer grossen Vielfalt weiterer impliziter und expliziter Methoden können auch die bereits vorgestellten Algorithmen (Euler Verfahren, Trapezregel) in diese Gruppe von Einschrittverfahren eingeordnet werden.

Betrachte das Integrationsintervall $[x_k, x_{k+1}]$ mit $x_{k+1} = x_k + h$ und die Stützstellen

$$\xi_j = x_k + c_j\, h, \quad c_j \in [0,1], \quad j = 1, \ldots, s, \tag{3.13}$$

dann lautet die allgemeine Form der zugehörigen interpolations Quadraturformel Q für die Funktion $g : [x_k, x_{k+1}] \to \mathbb{R}$

$$Q(g) = h \cdot \sum_{j=1}^{s} b_j g(\xi_j) \approx \int_{x_k}^{x_{k+1}} g(x)dx,$$

wobei $b_1, \ldots, b_s \in \mathbb{R}$ als Gewichte bezeichnet werden.

Eine Grundforderung besteht bei Quadraturformeln darin, dass das Integral für eine konstante Funktion exakt berechnet wird, daher

$$h\,c = \int_{x_k}^{x_{k+1}} g(x)dx = Q(g) = h \sum_{j=1}^{s} b_j \underbrace{g(\xi_j)}_{=c} = h\,c \sum_{j=1}^{s} b_j. \tag{3.14}$$

Daraus folgt sofort, dass

$$\sum_{j=1}^{s} b_j = 1$$

erfüllt sein muss. Für das Anfangswertproblem (3.1) folgt

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} f(x, y(x))dx \approx h \sum_{j=1}^{s} b_j f(\xi_j, y(\xi_j))$$

mit $\xi_j = x_k + c_j h$.

Das *s-stufige Runge-Kutta-Verfahren* folgt durch geeignete Wahl der Funktionswerte $y(\xi_j)$ durch Näherungswerte k_j:

$$y_{k+1} = y_k + h \sum_{j=1}^{s} b_j f(x_k + c_j h, k_j)$$

mit den Knoten c_j und den Gewichten b_j.

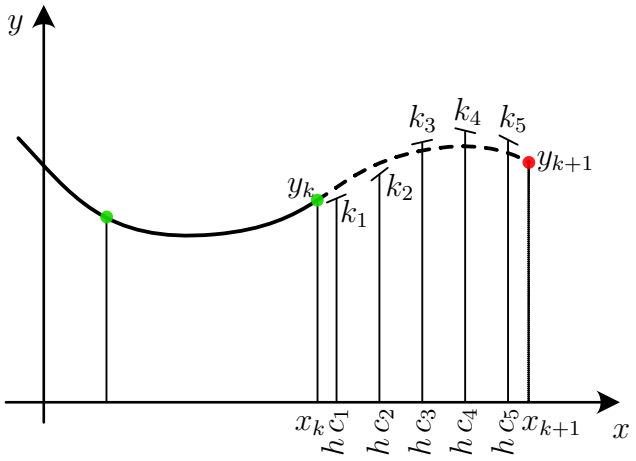


Abbildung 3.3: Quadratur Formel Beispiel für $s = 5$.

Beispiel 3.3. Die bereits bekannten Verfahren lassen sich wie folgt beschreiben

	Stufenzahl s	Gewichte $b_j, j = 1, \ldots, s$	Knoten $c_j, j = 1, \ldots, s$	Näherungen Gleichung	Vgl.
Explizites Euler Verfahren	1	$b_1 = 1$	$c_1 = 0$	$k_1 = y_k$	(3.2)
Implizites Euler Verfahren	1	$b_1 = 1$	$c_1 = 1$	$k_1 = y_{k+1}$	(3.4)
Implizite Trapezregel	2	$b_1 = b_2 = \frac{1}{2}$	$c_1 = 0, c_2 = 1$	$k_1 = y_k, k_2 = y_{k+1}$	(3.12)

Tabelle 3.1: Bekannte Verfahren

Die Hilfsgrössen k_j können wiederum durch Quadraturformeln approximiert werden. Es gilt

$$k_j \approx y(x_k + c_j\, h) = y(x_k) + \int_{x_k}^{x_k + c_j\, h} f(x, y(x))dx. \tag{3.15}$$

Zur Vermeidung zusätzlicher Funktionsauswertungen betrachten wir die schon in (3.13) festgelegten Stützstellen, womit sich analog zu (3.14) die Darstellung

$$k_j = y_k + h \sum_{\ell=1}^{s} a_{j,\ell} f(x_k + c_\ell h, k_\ell) \quad j = 1, \ldots, s$$

ergibt. Da auch die zur Approximation des in (3.15) auftretenden Integrals genutzten Quadraturformeln der Forderung (3.14) genügen sollen, erhalten wir die Bedingung

$$h \sum_{\ell=1}^{s} a_{j,\ell} = x_j + c_j h - t_j = c_j h,$$

daher

$$\sum_{\ell=1}^{s} a_{j,\ell} = c_j, \quad j = 1, \ldots, s.$$

Die auftretenden Koeffizienten $a_{j,\ell}, b_j, c_j$, $\ell = 1, \ldots s$ charakterisieren die unterschiedlichen Runge-Kutta-Verfahren. Schreibt man

$$\mathbf{A} = \begin{pmatrix} a_{11} & \ldots & a_{1s} \\ \vdots & & \vdots \\ a_{s1} & \ldots & a_{ss} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_s \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_s \end{pmatrix}$$

so ist ein s-stufiges Runge-Kutta-Verfahren vollständig durch die Angabe des zugehörigen, wie folgt erklärten *Butcher-Tableau* $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ gemäss

Butcher-Tableau

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} \quad \text{respektive} \quad \begin{array}{c|ccc} c_1 & a_{11} & \ldots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \ldots & a_{ss} \\ \hline & b_1 & \ldots & b_s \end{array} \tag{3.16}$$

festgelegt.

s-stufiges Runge-Kutta-Verfahren (Stufenform)

$$k_j = y_k + h \sum_{\ell=1}^{s} a_{j,\ell} f(x_k + c_\ell h, k_\ell), \quad j = 1, \ldots, s \tag{3.17a}$$

$$y_{k+1} = y_k + h \sum_{j=1}^{s} b_j \, f(x_k + c_j \, h, k_j) \tag{3.17b}$$

Bemerkung 3.4. Im Fall einer strikten linken unteren Dreiecksmatrix $\mathbf{A}$ können die Näherungswerte k_j sukzessive in der Reihenfolge $j = 1, 2, \ldots s$ durch eine explizite Verfahrensvorschrift berechnet werden (*explizite Runge-Kutta-Verfahren*). Weist $\mathbf{A}$ nichtverschwindende Einträge im oberen Dreiecksbereich auf, so sind in der Regel nicht alle Grössen k_j explizit berechenbar. Man bezeichnet das Verfahren in dem Fall als *implizit*.

⌟

Beispiel 3.4. Wir betrachten verschiedene Butcher-Tableau und die zugehörigen Runge-Kutta-Verfahren:

a) Das Butcher-Tableau

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

beschreibt ein explizites einstufiges Runge-Kutta-Verfahren. Es folgt mit

$$k_1 = y_k + h \underbrace{a_{11}}_{=0} f(x_k + c_1 h, k_1) = y_k$$

und

$$y_{k+1} = y_k + h \underbrace{b_1}_{=1} f(x_k + \underbrace{c_1}_{=0} h, k_1) = y_k + h f(x_k, y_k)$$

das explizite Euler Verfahren.

b) Analog liefert

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

wegen

$$k_1 = y_k + h \underbrace{a_{11}}_{=1} f(x_k + \underbrace{c_1}_{=1} h, k_1)$$

und

$$y_{k+1} = y_k + h \underbrace{b_1}_{=1} f(x_k + \underbrace{c_1}_{=1} h, k_1) = y_k + h f(x_{k+1}, y_{k+1})$$

das implizite Euler Verfahren.

c) Beim Runge-Kutta-Verfahren

$$y_{k+1} = y_k + h\, f(x_k + h/2, y_k + h/2\, f(x_k, y_k))$$

wird die Auswertung an den Zeitpunkten $\xi_1 = x_k$ und $\xi_2 = x_k + h/2$ benötigt. Daher muss $s = 2$, $c_1 = 0$ und $c_2 = 1/2$ gewählt werden. Es folgt

$$\begin{aligned} k_1 &= y_k \\ k_2 &= y_k + h \underbrace{a_{21}}_{=1/2} f(x_k, k_1) \\ y_{k+1} &= y_k + h \underbrace{b_2}_{=1} f(x_k + \underbrace{c_2}_{=1/2} h, k_2) \end{aligned}$$

womit das zugehörige Butcher-Tableau durch

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

gegeben ist. Es liegt ein explizites Runge-Kutta-Verfahren vor.

d) Das Butcher-Tableau

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

repräsentiert die implizite Mittelpunktregel und kann in der Form

implizite Mittelpunktregel

$$\begin{aligned} k_1 &= y_k + \frac{1}{2} h\, f(x_k + h/2, k_1) \\ y_{k+1} &= y_k + h\, f(x_k + h/2, k_1) \end{aligned} \tag{3.18}$$

geschrieben werden.

Bemerkung 3.5. Für die numerische Berechnung ist das in Gleichung (3.17) gegebene s-stufige Runge-Kutta Verfahren nicht optimal. Mit der Definition

$$r_j = f(x_k + c_j h, k_j), \quad j = 1, \ldots, s \tag{3.19}$$

kann das Verfahren von der sogenannten *Stufenform* in die *Steigungsform* transformiert werden, welche bedeutend weniger Funktionsauswertungen erfordert. Für (3.19) folgt mit (3.17a)

$$r_j = f(x_k + c_j h, k_j) = f\Big(x_k + c_j h, y_k + h \sum_{\ell=1}^{s} a_{j,\ell} \underbrace{f(x_k + c_\ell h, k_\ell)}_{=r_\ell}\Big).$$

Damit erhalten wir die *Steigungsform* des s-stufigen Runge-Kutta-Verfahrens

s-stufiges Runge-Kutta-Verfahren (Steigungsform)

$$r_j = f\Big(x_k + c_j h, y_k + h \sum_{\ell=1}^{s} a_{j,\ell}\, r_\ell\Big), \quad j = 1, \ldots, s \tag{3.20a}$$

$$y_{k+1} = y_k + h \sum_{j=1}^{s} b_j\, r_j. \tag{3.20b}$$

Klassische Runge-Kutta-Verfahren 4. Ordnung

Das sehr häufig verwendete sogenannte *klassische Runge-Kutta-Verfahren* stellt eine explizite vierstufige Methode der Konsistenzordnung $p = 4$ dar. Die Grundidee dieses Verfahrens liegt in der Nutzung der Simpson-Formel zu numerischen Integration, wobei in der Erweiterung eine doppelte Auswertung am mittleren Knoten $c = \frac{1}{2}$ vorgenommen wird und das hiermit gekoppelte Gewicht $b = \frac{2}{3}$ zu gleichen Teilen verteilt wird. Das Butcher-Tableau lautet

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} .$$

Für das s-stufige Runge-Kutta-Verfahren nach (3.17) folgt

$$k_1 = y_k \tag{3.21a}$$
$$k_2 = y_k + \frac{h}{2} f(x_k, k_1) \tag{3.21b}$$
$$k_3 = y_k + \frac{h}{2} f(x_k + \frac{1}{2} h, k_2) \tag{3.21c}$$
$$k_4 = y_k + h\, f(x_k + \frac{1}{2} h, k_3) \tag{3.21d}$$
$$y_{k+1} = y_k + \frac{h}{6}\Big(f(x_k, k_1) + 2f(x_k + \frac{h}{2}, k_2) + 2f(x_k + \frac{h}{2}, k_3) + f(x_k + h, k_4)\Big). \tag{3.21e}$$

Um unnötige Funktionsauswertungen zu vermeiden wechseln wir in die Steigungsform. Wir erhalten analog zur Bemerkung 3.5

$$y_{k+1} = y_k + \frac{h}{6}\Big(\underbrace{f(x_k, k_1)}_{r_1} + 2\underbrace{f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}h\, r_1)}_{r_2} + 2\underbrace{f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}h\, r_2)}_{r_3} + \underbrace{f(x_k + h, y_k + h\, r_3)}_{r_4}\Big)$$

die *Steigungsform* des Runge-Kutta-Verfahrens. Für das klassische Runge-Kutta-Verfahren folgt:

klassisches Runge-Kutta-Verfahren

$$\begin{aligned} r_1 &= f(x_k, y_k) \\ r_2 &= f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}h\, r_1) \\ r_3 &= f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}h\, r_2) \\ r_4 &= f(x_k + h, y_k + h\, r_3) \\ y_{k+1} &= y_k + \frac{h}{6}(r_1 + 2\, r_2 + 2\, r_3 + r_4). \end{aligned} \tag{3.22}$$

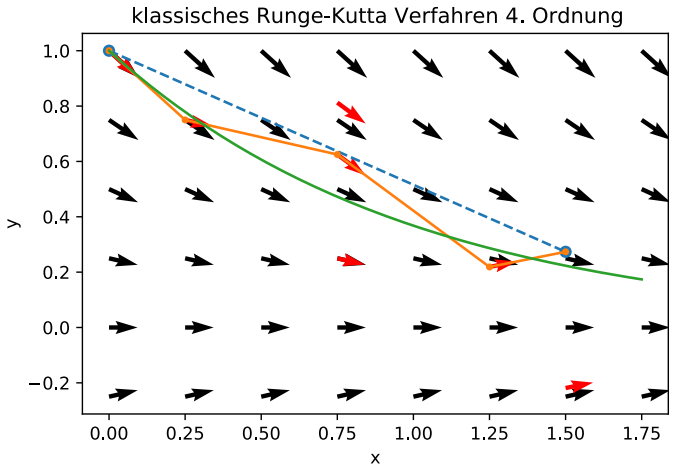


Abbildung 3.4: Klassisches Runge-Kutta Verfahren 4. Ordnung

3.2.5 Heun-Verfahren

Den Bereich der expliziten Runge-Kutta-Verfahren soll mit dem bekannten *Heun-Verfahren* abgeschlossen werden. Die Methode basiert auf der impliziten Trapezregel

$$y_{k+1} = y_k + \frac{1}{2}h\left(f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right),$$

wobei zur Vermeidung der impliziten Auswertung zunächst ein Zeitschritt mit dem expliziten Euler Verfahren durchgeführt wird und somit y_{k+1} auf der rechten Seite durch

$$y_k + hf(x_k, y_k)$$

ersetzt wird. Somit ergibt sich in der klassischen Form eines Runge-Kutta-Verfahrens die Darstellung

Heun-Verfahren

$$\begin{aligned} k_1 &= y_k \\ k_2 &= y_k + hf(x_k, k_1) \\ y_{k+1} &= y_k + \frac{1}{2}h(f(x_k, k_1) + f(x_{k+1}, k_2)). \end{aligned} \tag{3.23}$$

Das zugehörige Butcher-Tableau ist gegeben durch

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Das Verfahren nach Heun ist ein Verfahren zweiter Ordnung. Die Modifikation der Trapezregel unter Nutzung eines vorgezogenen Zeitschrittes auf der Basis des expliziten Euler Verfahrens führt nicht zu einer Reduktion der Ordnung.

Die Methode basiert auf dem *Prädikator-Korrektor-Verfahren*. Zunächst wird mit dem Euler Verfahren eine Näherungslösung k_2 ermittelt (Prädikator), um im abschliessenden Schritt die Näherung zu korrigieren (Korrektor).

3.2.6 Verfahren nach Hammer und Hollingsworth (optional)

Ohne auf die Details (vgl. Jupyter-Notebook) einzugehen, soll hier noch ein zweistufiges *implizites* Verfahren mit maximaler Ordnung $p = 4$ erwähnt werden. Das Verfahren nach Hammer und Hollingsworth ist durch das folgende Butcher-Tableau gegeben:

$$\begin{array}{c|cc} \frac{1}{2} - \frac{1}{6}\sqrt{3} & \frac{1}{4} & \frac{1}{4} - \frac{1}{6}\sqrt{3} \\ \frac{1}{2} + \frac{1}{6}\sqrt{3} & \frac{1}{4} + \frac{1}{6}\sqrt{3} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Ein allgemeines zweistufiges Verfahren ist durch das Butcher-Tableau

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} \\ c_2 & a_{21} & a_{22} \\ \hline & b_1 & b_2 \end{array}$$

gegeben. Für das Verfahren folgt demnach

$$\begin{aligned} k_1 &= y_k + h\left(a_{11} f(x_k + c_1\, h, k_1) + a_{12} f(x_k + c_2\, h, k_2)\right) \\ k_2 &= y_k + h\left(a_{21} f(x_k + c_1\, h, k_1) + a_{22} f(x_k + c_2\, h, k_2)\right) \\ y_{k+1} &= y_k + h\left(b_1 f(x_k + c_1\, h, k_1) + b_2 f(x_k + c_2\, h, k_2)\right) \end{aligned}$$

bzw. mit den Werten von Hammer und Hollingsworth eingesetzt

Verfahren nach Hammer und Hollingsworth

$$\begin{aligned} k_1 &= y_k + h\left(\frac{1}{4}\cdot f\left(x_k + \frac{3-\sqrt{3}}{6}\,h, k_1\right) + \frac{3-2\sqrt{3}}{12}\cdot f\left(x_k + \frac{3+\sqrt{3}}{6}\,h, k_2\right)\right) \\ k_2 &= y_k + h\left(\frac{3+2\sqrt{3}}{12}\cdot f\left(x_k + \frac{3-\sqrt{3}}{6}\,h, k_1\right) + \frac{1}{4}\cdot f\left(x_k + \frac{3+\sqrt{3}}{6}\,h, k_2\right)\right) \\ y_{k+1} &= y_k + \frac{1}{2}\cdot h\left(f\left(x_k + \frac{3-\sqrt{3}}{6}\,h, k_1\right) + f\left(x_k + \frac{3+\sqrt{3}}{6}\,h, k_2\right)\right) \end{aligned} \tag{3.24}$$

3.3 Übersicht der Numerischen Verfahren

Verfahrensname	Gleichung	Ordnung p	Abkürzung
Explizites Euler Verfahren	(3.2)	1	EE
Runge-Verfahren	(3.11)	2	Runge
Heun-Verfahren	(3.23)	2	Heun
Klassisches Runge-Kutta-Verfahren	(3.22)	4	ERK4

(a) Explizite Einschrittverfahren

Verfahrensname	Gleichung	Ordnung p	Abkürzung
Implizit Euler Verfahren	(3.4)	1	IE
Implizite Mittelpunktregel	(3.18)	2	IM
Implizite Trapezregel	(3.12)	2	IT
Implizites Verfahren nach Hammer und Hollingsworth	(3.24)	4	IHH

(b) Implizite Einschrittverfahren

Tabelle 3.2: Übersicht numerische Verfahren für Anfangswertprobleme der Form $y'(x) = f(x, y(x)), y(0) = y_0$.

Zur Untersuchung des prognostizierten Konvergenzverhalten der vorgestellten numerischen Verfahren betrachten wir das Problem

$$\begin{aligned} y'(t) &= k\,(c_0 - y(t)) \\ y(0) &= 0 \end{aligned} \tag{3.25}$$

mit $k \in \mathbb{R}^+$ und $c_0 \in \mathbb{R}_0^+$. Für $k = \frac{R}{L}$ und $c_0 = \frac{U_0}{R}$ beschreibt die Differentialgleichung das Einschwingverhalten eines R-L-Gliedes an einer DC-Spannungsquelle. Die analytische Lösung ist gegeben durch

$$y(t) = c_0(1 - e^{-k,t})$$

mit der Eigenschaft

$$\lim_{t\to\infty} y(t) = c_0 \left(= \frac{U_0}{R}\right).$$

Bei Festlegung der Konstanten $k = 0.3$ und $c_0 = 10$ betrachten wir das Maximum der absoluten Differenz zwischen dem numerischen Wert y_{num} und der analytischen Lösung y gemäss

$$\epsilon_{\Delta t} = \max_{t\in[0,T]} |e(t, \Delta t)| = \max_{t\in[0,T]} |y(t) - y_{\text{num}}(t, \Delta t)|$$

zum Zeitpunkt $t = 5$ bei variierter Zeitschrittweite $\Delta t \in \{1, 0.1, 0.01, 0.001, 0.0001\}$. Der Wert $y_{\text{num}}(t, \Delta t)$ entspricht stets der Grösse y_k mit $k = \frac{t}{\Delta t}$.

	EE	Runge / Heun	ERK4
Δt	$p = 1$	$p = 2$	$p = 4$
10^0	0.635696597405992	0.069239652594008	0.000317429687035
10^{-1}	0.055883667535561	0.000564398486584	0.000000025459431
10^{-2}	0.005525101107902	0.000005530623804	0.000000000002459
10^{-3}	0.000551888151156	0.000000055194048	0.000000000000303
10^{-4}	0.000055182603101	0.000000000554749	0.000000000003613
	IE	IM/IT	IHH
Δt	$p = 1$	$p = 2$	$p = 4$
10^0	0.489335847335735	0.027725034983046	0.000041385908482
10^{-1}	0.054499595695614	0.000275922699853	0.000000004138659
10^{-2}	0.005511301451693	0.000002759097175	0.000000000000517
10^{-3}	0.000551750191186	0.000000027590739	0.000000000001856
10^{-4}	0.000055181223733	0.000000000269092	0.000000000005781

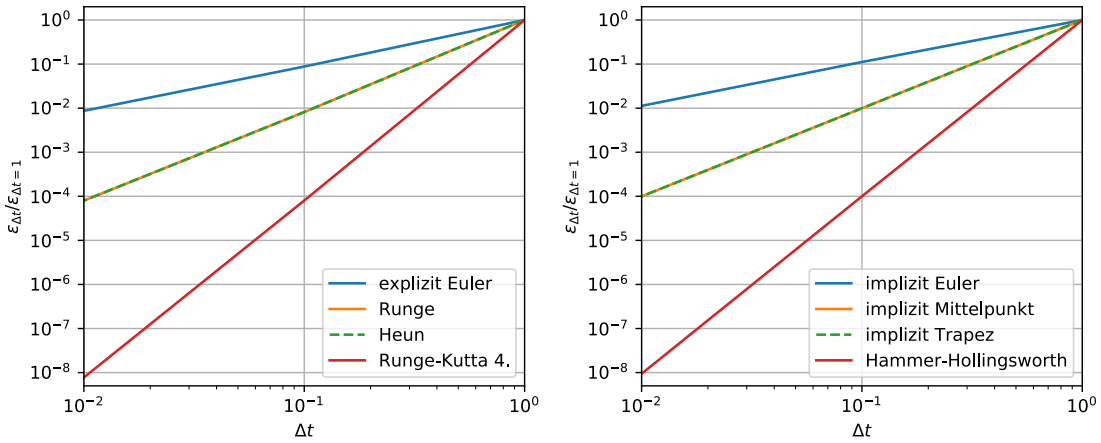


Abbildung 3.5: Konvergenzrate

In der Abbildung 3.6 ist zu beobachten, dass der Fehler der RK4, IHH und IAM3 Verfahren für $\Delta t < 10^{-1}$ wieder grösser wird. Dabei stellt sich natürlich die Frage, was

die Ursache des Problems ist? Zumal die Konvergenzordnung der Verfahren gut mit der Theorie übereinstimmt (vgl. Abb. 3.5).

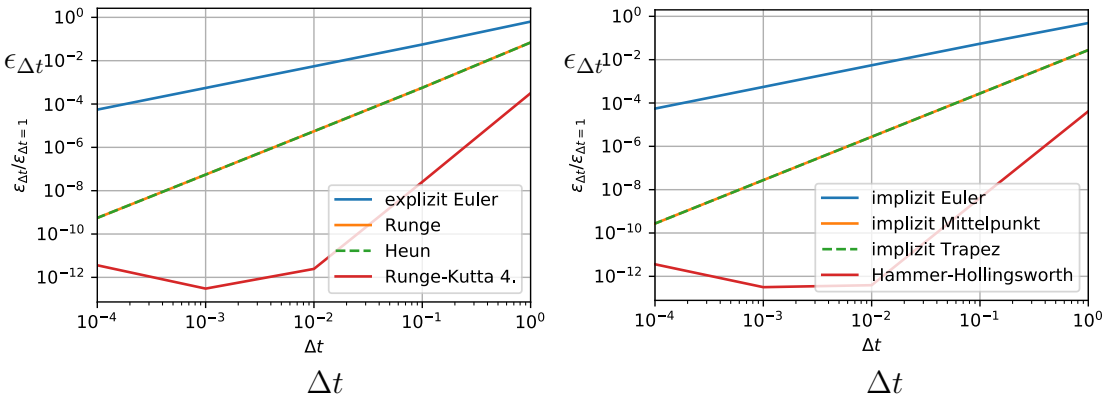


Abbildung 3.6: Benchmark

3.3.1 Rechenfehler bei der Durchführung des Verfahrens

Grundsätzlich gibt es drei Typen von Ursachen, die zu einem Abweichen der exakten von der berechneten Lösung führen können:

(i) *Datenfehler* bei den Anfangsbedingungen, d.h. der verwendete Startwert x_0 ist nicht gleich $x(t_0)$

(ii) *Diskretisierungsfehler*, d.h. der Fehler, der durch die Verwendung der diskreten Approximationsverfahren entsteht

(iii) *Rechenfehler* bei der Durchführung des Algorithmus, die z.B. durch Rundungseffekte bedingt sein können

Meistens liegt der Fokus bei der Fehleranalyse auf der offensichtlichen Fehlerquelle (ii), aber (i) und (iii) müssen auch berücksichtigt werden.

Hier geht es um die Frage der Stabilität des Verfahrens, d.h. ob ein bei einem Rechenschritt gemachter Fehler in den Folgeschritten abnimmt oder höchstens von der gleichen Grössenordnung bleibt. Wir gehen nicht auf diese Diskussion ein und erwähnen nur eine Erweiterung der Formel (3.7) auf den Fall mit Fehlertermen r_k. Wir betrachten also anstatt (3.3) das modifizierte Verfahren

$$\tilde{y}_{k+1} = \tilde{y}_k + h \cdot \phi(x_k, \tilde{y}_k, h) + r_k, \quad k = 0, 1, 2, \ldots, \tag{3.26}$$

wobei r_k der bei der Auswertung von $\phi(x_k, \tilde{y}_k, h)$ auftretende Fehler ist, z.B. durch Rundungsfehler. Für die durch (3.26) definierte Folge $(\tilde{y}_k)_{k\in\mathbb{N}}$ gilt dann, unter ähnlichen Annahmen wie in Satz 3.2, für den durch (3.26) entstehenden Fehler $\tilde{e}_k = |y(x_k) - \tilde{y}_k|$ die

Abschätzung

$$\max_{0\le k\le n}|\tilde{e}_k| = \left(\overbrace{|y(x_0)-y_0|}^{\text{Fehler i)}} + \sum_{k=0}^{n-1} \left(\overbrace{\frac{|\eta_{k,h}|}{h}}^{\text{Fehler ii)}} + \overbrace{|r_k|}^{\text{Fehler iii)}} \right) \right) e^{L(x_n-x_0)}.$$

Für das Verfahren nach Hammer und Hollingsworth (IHH) wurden numerische Berechnungen für $h \in [10^{-6}, 1]$ durch geführt und die maximalen Fehler berechnet. Um die Fehlergrössen zu bestimmen benutzen wir das Fehlermodell

$$e(h) = \frac{a}{h} + b \cdot h^p$$

und berechnen mit Hilfe der nichtlinearen Ausgleichsrechnung (log-Daten benutzen) die Parameter a, b, p, wobei $p \approx 4$ zu erwarten ist. Es folgt (vgl. Abb. 3.7)

$$e(h) \approx \underbrace{\frac{6.78213 \cdot 10^{-16}}{h}}_{\text{Fehler iii)}} + \underbrace{0.0000428638 \cdot h^{4.0263}}_{\text{Fehler ii)}}.$$

Der Datenfehler ist im Beispiel Null (Anfangsbedingungen werden exakt erfüllt) und der Rechenfehler nimmt mit $1/h$ zu für kleiner werdendes h. Daraus ergibt sich eine optimale Schrittweite mit der das Problem mit kleinstem Fehler numerisch berechnet werden kann.

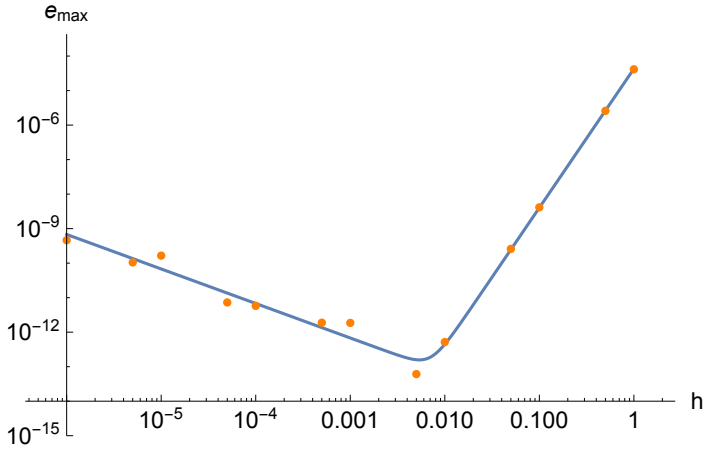


Abbildung 3.7: Maximaler Fehler des Verfahren nach Hammer und Hollingsworth (IHH) für $h \in [10^{-6}, 1]$.

3.3.2 Schrittweitensteuerung bei Einschrittverfahren

Im Verlauf der numerischen Behandlung eines Differentialgleichungssystems kann sich der Diskretisierungsfehler mit wachsendem x stark ändern. Deshalb spielt eine dem Problem angepasste Schrittweite h_k eine wichtige Rolle. Grundsätzlich soll die Schrittweite über eine *Fehlerschätzung* gesteuert werden. Da die Abschätzungen des globalen Fehlers schwer zu bestimmen sind (vgl. Satz 3.2) und darüber hinaus den tatsächlichen Fehler meist stark überschätzen, ist eine Schätzung

$$T(x,h) \approx |\eta(x,h)|$$

des Betrags des lokalen Fehlers günstiger. Es gibt verschiedene Strategien dies umzusetzten.

An dieser Stelle gehen wir auf eine bewährte Methode ein: Parallelrechnung mit zwei Verfahren verschiedener Ordnung oder Stufe. Zur Aufwandsminimierung werden dabei *eingebettete Verfahren* verwendet. Das sind Verfahren, bei denen die Koeffizienten gemeinsamer Stufe übereinstimmen, so dass dann die Werte k_ℓ nur einmal berechnet werden müssen:

Runge-Kutta-Verfahren 4. und 5. Ordnung mit Fehlerschätzung nach Fehlberg

$$\begin{aligned}
r_1 &= f(x_k, y_k)\\
r_2 &= f\left(x_k + \frac{2}{9}h, y_k + \frac{2}{9}h\,r_1\right)\\
r_3 &= f\left(x_k + \frac{1}{3}h, y_k + \frac{1}{12}h\,r_1 + \frac{1}{4}h\,r_2\right)\\
r_4 &= f\left(x_k + \frac{3}{4}h, y_k + \frac{69}{128}h\,r_1 - \frac{243}{128}h\,r_2 + \frac{135}{64}h\,r_3\right)\\
r_5 &= f\left(x_k + h, y_k - \frac{17}{12}h\,r_1 + \frac{27}{4}h\,r_2 - \frac{27}{5}h\,r_3 + \frac{16}{15}h\,r_4\right)\\
y_{k+1} &= y_k + h\left(\frac{1}{9}r_1 + \frac{9}{20}r_3 + \frac{16}{45}r_4 + \frac{1}{12}r_5\right)
\end{aligned} \tag{3.27}$$

Dieses fünfstufige Verfahren 4. Ordnung wird eingebettet mit der Erweiterung

$$\begin{aligned}
r_6 &= f\left(x_k + \frac{5}{6}h, y_k + \frac{65}{432}h\,r_1 - \frac{5}{16}h\,r_2 + \frac{13}{16}h\,r_3 + \frac{4}{27}h\,r_4 + \frac{5}{144}h\,r_5\right)\\
\tilde{y}_{k+1} &= y_k + h\left(\frac{47}{450}r_1 + \frac{12}{25}r_3 + \frac{32}{225}r_4 + \frac{1}{30}r_5 + \frac{6}{25}r_6\right).
\end{aligned} \tag{3.28}$$

Schätzwert des lokalen Fehlers der Methode (3.27) ist der Betrag der Differenz $\tilde{y}_{k+1} - y_{k+1}$

$$T(x_{k+1}, h) = \frac{h}{300}|-2\,r_1 + 9\,r_3 - 64\,r_4 - 15\,r_5 + 72\,r_6|. \tag{3.29}$$

Eine mögliche Strategie ist in Abbildung 3.8 veranschaulicht.

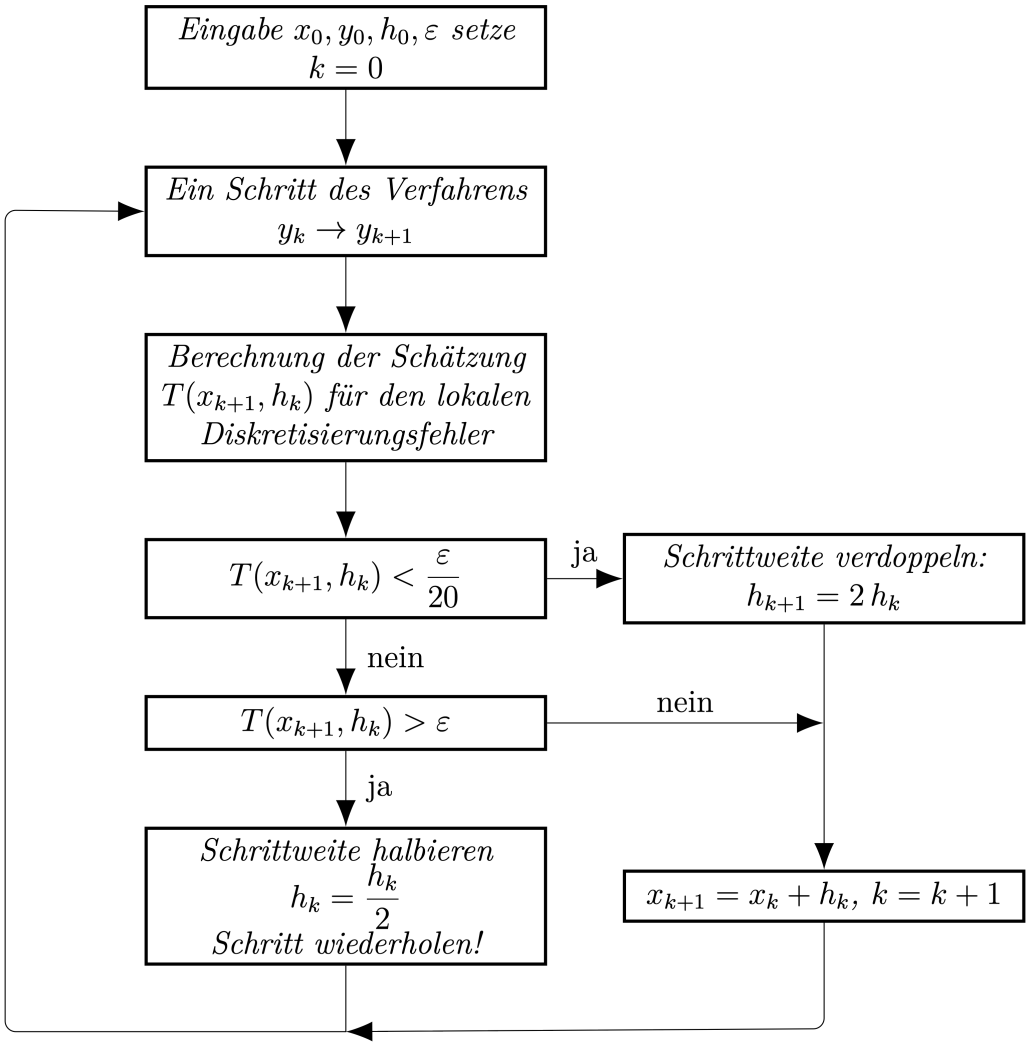


Abbildung 3.8: Schema einer möglichen Strategie zur Schrittweitensteuerng (vgl. [7]).

Beispiel 3.5. Es soll die Differentialgleichung

$$y' = -(\sin(x^3) + 3x^3\cos(x^3))\,y, \quad y(0) = 1 \tag{3.30}$$

mit dem Runge-Kutta-Verfahren (3.27) und der beschriebenen Schrittweitensteuerung gelöst werden. Die analytische Lösung ist gegeben durch

$$y(x) = \exp(-x\,\sin(x^3)).$$

Sie ähnelt einem glatten Einschwingvorgang, dem eine in Frequenz und Amplitude wachsende Schwingung folgt, siehe Abbildung 3.9.

In der Tabelle 3.3 sind die numerischen Resultate für verschiedene Toleranzen ε zu sehen. Der effektive maximale Fehler $e_{\max}$ weicht natürlich davon ab, sollte jedoch mit kleiner werdenden Toleranz ebenfalls kleiner werden (vgl. Abb. 3.9 rechts unten).

ε	$h_{\min}$	$h_{\max}$	# Schritte	$e_{\max}$
0.100	0.01562500	0.500	33	9.36
0.010	0.00781250	0.500	54	1.09
0.001	0.00781250	0.500	76	8.12e-02
1e-04	0.00390620	0.250	124	1.21e-02
1e-05	0.00195310	0.250	192	4.26e-03
1e-06	0.00195310	0.125	304	5.72e-04
1e-07	0.00097656	0.125	484	6.25e-05

Tabelle 3.3: Numerische Resultate für verschiedene Toleranzen ε.

Im Vergleich dazu die Berechnung mit konstanter Schrittweite h mit dem Runge-Kutta-Verfahren nach Fehlberg (3.27): Um mit konstanter Schrittweite einen maximalen Fehler von $e_{\max} = 6.25 \cdot 10^{-5}$ zu erreichen, ist eine Schrittweite von $h_{\text{const}} = 0.003378$ notwendig (889 Schritte, 1.835×mehr Schritte). Mit 484 Schritten ($h_{\text{const}} = 0.0061983$) wird ein maximaler Fehler von $e_{\max} = 2.697 \cdot 10^{-4}$ erreicht (4.3×grösser Fehler).

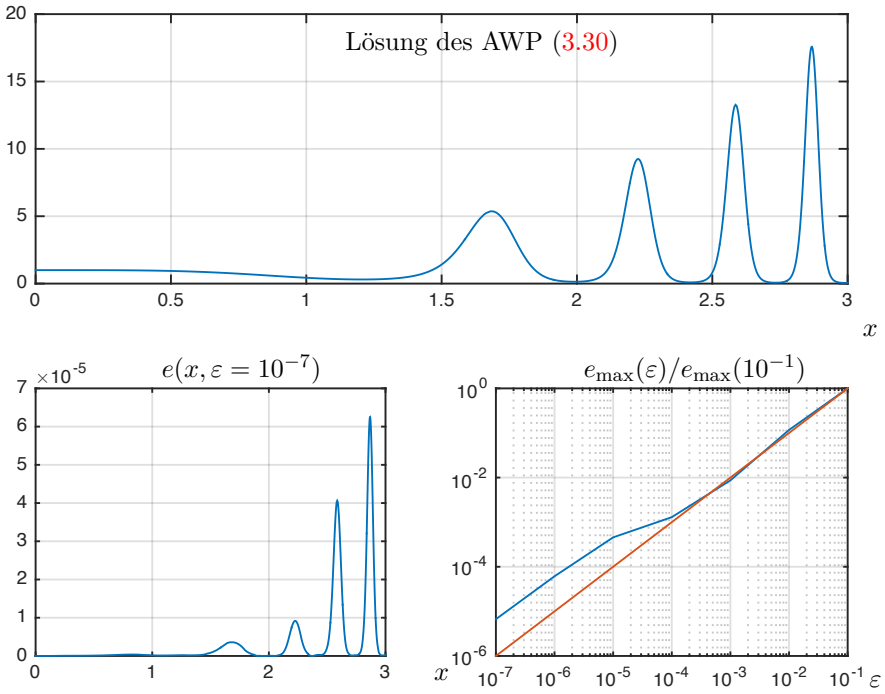


Abbildung 3.9: Schwingung mit wachsender Amplitude und Frequenz.

3.4 Numerische Methoden für Systeme

3.4.1 Runge-Kutta-Einschrittverfahren

Im Kapitel 3.2 sind wir von skalaren Anfangswertproblemen der Form

$$y'(x) = f(x, y) \quad x \in [a, b] \subset \mathbb{R}_0^+$$
$$y(x_0) = y_0$$

aus gegangen. Die behandelten Methoden lassen sich direkt auf Differentialgleichungssysteme

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}) \quad x \in [a, b] \subset \mathbb{R}_0^+$$
$$\mathbf{y}(x_0) = \mathbf{y}_0$$

übertragen, wobei $\mathbf{y} : [a, b] \to \mathbb{R}^n$ und $\mathbf{f} : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n$ gilt.

Beispiel 3.6. Wir betrachten das mathematische Pendel. Ein Massenpunkt der Masse m sei an einem gewichtlosen Faden der Länge ℓ befestigt (vgl. Abb. 3.10). Es soll der Verlauf des Ausschlagwinkels φ als Funktion der Zeit t bestimmt werden. Wie üblich bezeichne g die Erdbeschleunigung. Aus der Abbildung ist ersichtlich, dass die Schwerkraft gegeben ist durch

$$m\, g\, \sin(\varphi(t)).$$

Die Trägheitskraft in dieser Richtung lautet

$$m\, \ell\, \ddot{\varphi}(t).$$

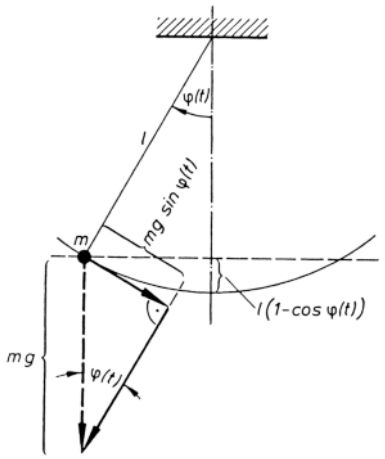


Abbildung 3.10: Mathematisches Pendel

Aus der Kräftegleichgewichtsbedingung folgt die Differentialgleichung zweiter Ordnung

$$m\, \ell\, \ddot{\varphi}(t) + m\, g\, \sin(\varphi(t)) = 0$$

und somit

$$\ddot{\varphi} + \frac{g}{\ell} \sin(\varphi) = 0.$$

Im Startzeitpunkt gilt $\varphi(0) = \varphi_0$ (das Pendel ist ausgelenkt) und in Ruhe $\dot{\varphi}(0) = 0$. Wir betrachten daher das Anfangswertproblem

$$\ddot{\varphi}(t) + \frac{g}{\ell} \sin(\varphi(t)) = 0$$
$$\varphi(0) = \varphi_0$$
$$\dot{\varphi}(0) = 0.$$

Das System erster Ordnung ist in dem Fall gegeben durch

$$\dot{\varphi}_1 = \varphi_2$$
$$\dot{\varphi}_2 = -\frac{g}{\ell} \sin(\varphi_1)$$
$$\varphi_1(0) = \varphi_0 \quad \varphi_2(0) = 0.$$

Es gilt daher

$$\mathbf{y} = \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} \varphi_2 \\ -\frac{g}{\ell} \sin(\varphi_1) \end{pmatrix} \quad \text{und} \quad \mathbf{y}_0 = \begin{pmatrix} \varphi_0 \\ 0 \end{pmatrix}.$$

Es ist zu beachten, dass r_j bzw. k_j in dem Fall vektorwertige Grössen sind. In der Abbildung 3.11 ist die Lösung für $g \approx 9.81\text{m/s}^2, \ell = 1\text{m}$, berechnet mit dem klassischen Runge-Kutta-Verfahren (3.22), zu sehen.

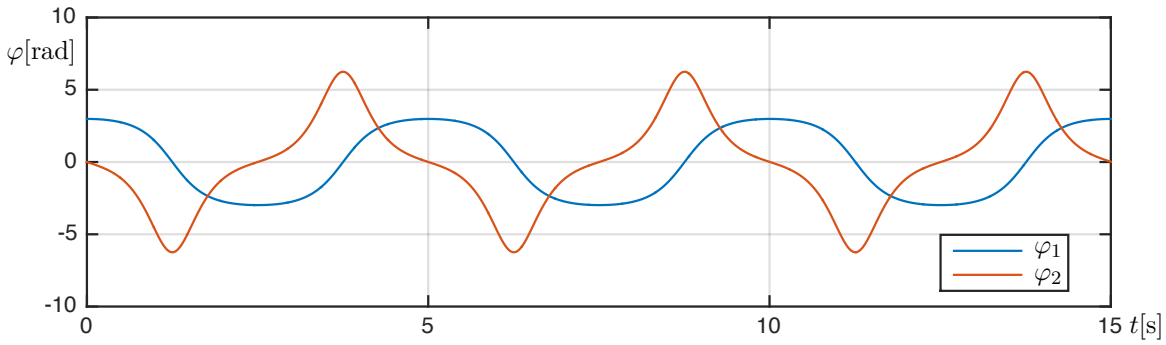


Abbildung 3.11: Lösung für $\varphi_0 = \pi - 0.16$, $\Delta t = 0.01\text{s}$

3.4.2 Konservative Methoden

Das autonome Anfangswertproblem

$$\begin{aligned} \dot{x}(t) &= y(t) \\ \dot{y}(t) &= -x(t) \end{aligned} \tag{3.31a}$$

und

$$(x(0), y(0)) = (0, 1) \tag{3.31b}$$

besitzt die analytische Lösung

$$(x(t), y(t)) = (\sin(t), \cos(t)).$$

In der Abbildung 3.12 sind die Lösungen zweier numerischer Methoden zu sehen. Es wurde wiederum das explizite Runge-Kutta Verfahren und das implizite Verfahren nach Hammer und Hollingsworth angewendet. Beide Verfahren sind von der Konvergenz Ordnung $p = 4$. Wird jeder numerische Fehler als Störung des stabilen Systems verstanden, dann wird letztendlich durch den numerischen Algorithmus nicht die Kreisbahn approximiert sondern eine benachbarte Spiralbahn berechnet, die sich - je nach Schrittweite - langsamer oder schneller asymptotisch dem Nullpunkt nähert.

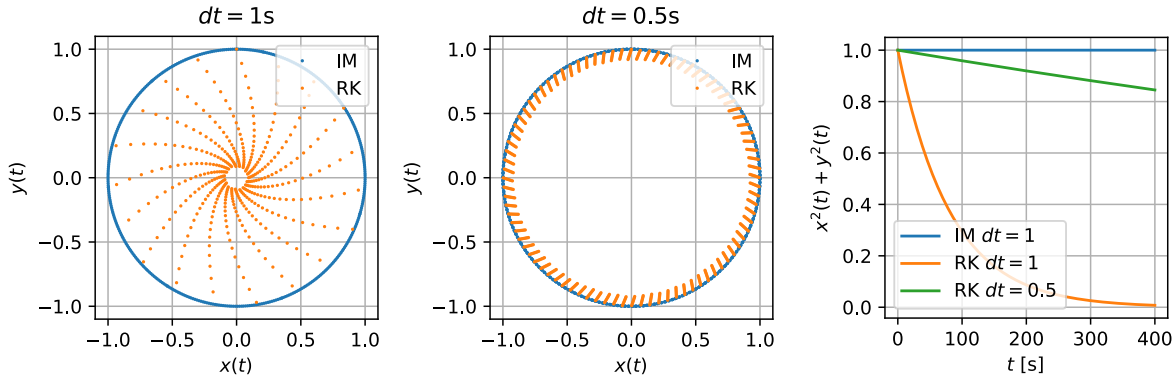


Abbildung 3.12: Konservative Methoden [Jupyter-Notebook]

Nach Ljapunoff kann dieses Verhalten als stetiger Energieverlust verstanden werden. Zur Berechnung dieser Kreisbahn sollte deswegen ein Verfahren benutzt werden, welches *konservativ* ist, dh. welches das Ljapunoff Potential

$$E(x, y) = \frac{1}{2}(x^2 + y^2)$$

konstant lässt. Zeitdiskret bedeutet dies

$$\frac{1}{2}(x_k^2 + y_k^2) \equiv C \quad \forall\, k \in \mathbb{N}$$

unabhängig von der Schrittweite h.

Satz 3.4 (Jupyter-Notebook). *Die implizite Trapezmethode, implizite Mittelpunkt-Regel und das Verfahren nach Hammer und Hollingsworth sind konservativ.*

Beweis. (Beweisskizze) Zur Verifikation betrachten wir das Anfangswertproblem (3.31). Da das Problem linear ist, kann die implizite Formulierung in eine explizite umgeschrieben werden. Für die implizite Trapezmethode (3.12) folgt

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} -\frac{h^2 x_k - 4h y_k - 4x_k}{h^2+4} \\ -\frac{h^2 y_k + 4h x_k - 4y_k}{h^2+4} \end{pmatrix}$$

Durch Quadrieren und Addieren der Komponenten (Skalarprodukt der linken und rechten Seite) folgt

$$x_{k+1}^2 + y_{k+1}^2 = x_k^2 + y_k^2.$$

Potential E ist daher unabhängig von h und somit das Verfahren konservativ. Beweis für die implizite Mittelpunkt-Regel als Übung.

Das Vorgehen beim Verfahren nach Hammer und Hollingsworth (3.24) ist identisch. □

Bemerkung 3.6. Für das klassische Runge-Kutta Verfahren (3.22) folgt

$$x_{k+1}^2 + y_{k+1}^2 = \underbrace{\frac{576 - 8h^6 + h^8}{576}}_{=:\kappa(h)} \cdot (x_k^2 + y_k^2).$$

In der Abbildung 3.13 ist $1-\kappa(h)$ graphisch dargestellt. Für kleine Schrittweiten h ist das klassische Runge-Kutta Verfahren „nahezu“ konservativ.

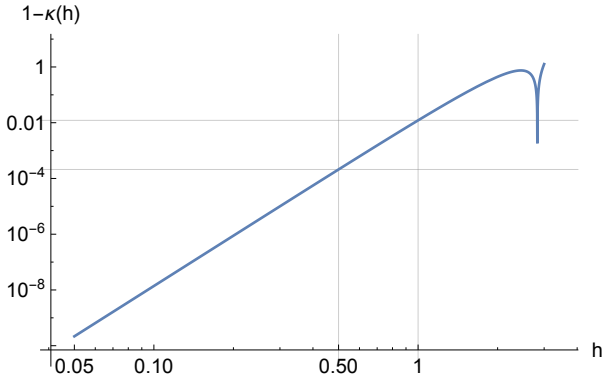


Abbildung 3.13: Die Schrittweite $h = 0.5, 1$ in Abbildung 3.12 verletzen die Energieerhaltung.

3.4.3 Stabilitätsbetrachtung

Steife Gleichungen sind Probleme, für die explizite Methoden nicht bzw. schlecht funktionieren. Wir starten die Betrachtung mit einem eindimensionalen Beispiel anhand dessen Curtiss & Hirschfelder (1952) die Steifigkeit illustrierten.

Beispiel 3.7 (Jupyter-Notebook). Betrachte das Anfangswertproblem

$$y'(x) = -50\,(y(x) - \cos(x)) \quad \text{und} \quad y(0) = 0. \tag{3.32}$$

In der Abbildung 3.14 sind die Lösungskurven für das implizite Euler Verfahren und für das explizite Euler Verfahren dargestellt (graue Linien in der Abbildung 3.14).

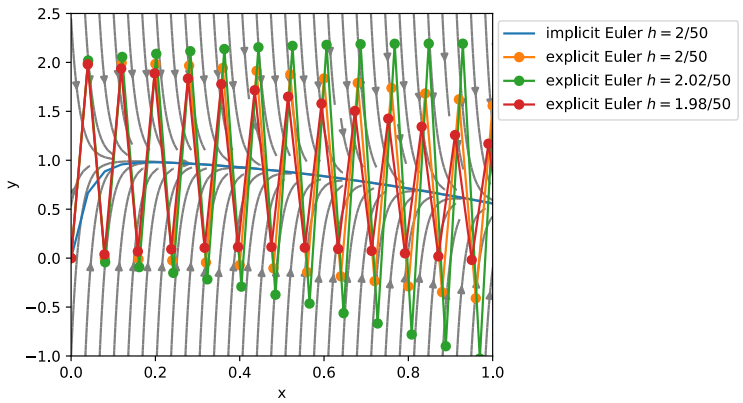


Abbildung 3.14: Lösungskurven des Anfangswertproblems (3.32).

Es gibt offenbar eine glatte Lösung in der Nähe von $y \approx \cos x$. Alle anderen Lösungen erreichen diese nach einer schnellen Einschwingphase. Solche Einschwingvorgänge sind typisch für steife Gleichungen, sind aber weder ausreichend noch notwendig.

In der Anwendung des Euler Verfahrens auf das Modellproblem $y' = \lambda y,\ y(0) = 1$ haben wir für das explizite Verfahren die Bedingung $|1 + \lambda h| < 1$ erhalten. Auf unser

Problem (3.32) angewandt folgt

$$|1 - 50h| < 1.$$

Da $h > 0$ folgt

$$1 - 50h > -1$$

und damit

$$0 < h < \frac{2}{50}.$$

In der Abbildung 3.14 sind für das explizite Euler Verfahren Lösungen für die Schrittweiten $h \in \{1.98/50, 2/50, 2.02/50\}$ dargestellt. Im Fall $h = 2/50$ oszilliert die numerische Lösung mit konstanter Amplitude. Für $h > 2/50$ nimmt die Amplitude mit jedem Schritt zu und für $h < 2/50$ entsprechend ab. ⌟

Stabilitätsanalyse für das explizite Euler Verfahren

Sei $\boldsymbol{\varphi}(x)$ eine Lösung von $\mathbf{y}' = \mathbf{f}(x, \mathbf{y})$. Das im Punkt $\boldsymbol{\varphi}(x)$ linearisierte Problem ist gegeben durch

$$\mathbf{y}'(x) = \mathbf{f}(x, \boldsymbol{\varphi}(x)) + \partial_{\mathbf{y}}\mathbf{f}(x, \boldsymbol{\varphi}(x)) \cdot (\mathbf{y}(x) - \boldsymbol{\varphi}(x)) + \dots,$$

wobei mit $\partial_{\mathbf{y}}\mathbf{f}$ die Jacobi-Matrix von $\mathbf{f}$ bezüglich $\mathbf{y}$ bezeichnet sei. Mit der Notation

$$\bar{\mathbf{y}}(x) = \mathbf{y}(x) - \boldsymbol{\varphi}(x)$$

folgt für $\bar{\mathbf{y}}(x)$ die Differentialgleichung

$$\bar{\mathbf{y}}'(x) = \partial_{\mathbf{y}}\mathbf{f}(x, \boldsymbol{\varphi}(x)) \cdot \bar{\mathbf{y}}(x) + \dots = J(x) \cdot \bar{\mathbf{y}}(x) + \dots.$$

Als erste Approximation betrachten wir die Jacobi-Matrix $J(x)$ als konstant und vernachlässigen die Terme höherer Ordnung. Wir erhalten somit

$$\mathbf{y}' = J \cdot \mathbf{y}, \tag{3.33}$$

wobei wir den Strich über y wieder weglassen. Wendet man das explizite Euler Verfahren an, so folgt

$$\mathbf{y}_{k+1} = R(h\,J) \cdot \mathbf{y}_k \tag{3.34}$$

mit

$$R(z) = 1 + z.$$

Um das Verhalten der Gleichung (3.34) studieren zu können, nehmen wir an, dass die Jacobi-Matrix J diagonalisierbar ist. Mit den Eigenvektoren $\mathbf{v}_j$ und Eigenwerten λ_j folgt

$$\mathbf{y}_k = \sum_{j=1}^{n} R(h\lambda_j)^k \alpha_j \mathbf{v}_j,$$

wobei α_j die Koeffizienten der Linearkombination

$$\mathbf{y}_0 = \sum_{j=1}^{n} \alpha_j \mathbf{v}_j$$

bezeichnen. Es folgt sofort, dass $\mathbf{y}_k$ für $k \to \infty$ beschränkt bleibt, wenn für alle Eigenwerte λ_j die komplexe Zahl $z = h\,\lambda_j$ in der Menge

$$S = \{z \in \mathbb{C} |\ |R(z)| \leq 1\}$$

liegt. Für das explizite Euler Verfahren mit $R(z) = 1 + z$ folgt

$$S = \{z \in \mathbb{C} |\ |1 + z| \leq 1\}.$$

Die Menge in der Eulerschen Zahlenebene dargestellt, ist ein Einheitskreis im Punkt $(-1, 0)$.

Definition 3.6 (Stabilitätsanalyse). Wir bezeichnen ein Verfahren im Punkt $\boldsymbol{\varphi}(x)$ *stabil*, falls

$$|R(z)| \leq 1 \quad \text{mit } z = h\,\lambda_j \text{ für alle } j = 1, \ldots, n \tag{3.35}$$

gilt, wobei $\lambda_j,\ j = 1, \ldots, n$ die Eigenwerte der Jacobi-Matrix $J = \partial_{\mathbf{y}}\mathbf{f}(x, \boldsymbol{\varphi}(x))$ im Arbeitspunkt $\boldsymbol{\varphi}(x)$ bezeichnen.

Es wird daher im Weiteren darum gehen, die Stabilitätsfunktion $R(z)$ für weitere Verfahren zu untersuchen.

Stabilitätsanalyse für explizite Runge-Kutta Verfahren

Wir betrachten nun explizite Runge-Kutta Verfahren im Allgemeinen. In dem Fall gilt gemäss (3.17)

$$k_j = y_k + h \sum_{\ell=1}^{j-1} a_{j,\ell} f(x_k + c_\ell h, k_\ell) \quad j = 1, \ldots, s \tag{3.36a}$$

$$y_{k+1} = y_k + h \sum_{j=1}^{s} b_j\, f(x_k + c_j\, h, k_j). \tag{3.36b}$$

Der entscheidende Unterschied zum allgmeinen RK-Verfahren liegt in der Summe für k_j, für den Index ℓ gilt $\ell = 1, \ldots, j-1$. Das heisst, die Summe (3.36b) für y_{k+1} kann explizit berechnet werden. Für die linearisierte Differentialgleichung (3.33) folgt mittels sukzessivem Einsetzen das Polynom

$$R(z) = 1 + z \sum_j b_j + z^2 \sum_{j,\ell} b_j a_{j,\ell} + z^3 \sum_{j,\ell,k} b_j a_{j,\ell} a_{\ell,k} + \ldots.$$

mit $\text{grad}(R(z)) \leq s$.

Definition 3.7 (Stabilitätsfunktion). Die Funktion $R(z)$ wird *Stabilitätsfunktion* der numerischen Methode genannt. Sie kann als numerische Lösung nach einem Schritt der *Dahlquist Testgleichung*

$$y' = \lambda\, y, \quad y(0) = 1, \quad z = h\,\lambda \tag{3.37}$$

interpretiert werden. Die Menge

$$S = \left\{z \in \mathbb{C} \mid |R(z)| \ \leq 1\right\} \tag{3.38}$$

wird *Stabilitätsbereich* genannt.

Satz 3.5. *Ist die Runge-Kutta Methode von der Ordnung p, dann gilt*

$$R(z) = 1 + z + \frac{z^2}{2!} + \ldots + \frac{z^p}{p!} + \mathcal{O}(z^{p+1}).$$

Beweis. Die exakte Lösung der Dahlquist Gleichung (3.37) ist e^z. Daher gilt für die numerische Lösung nach einem Schritt

$$y_1 = R(z).$$

Aufgrund der vorausgesetzten Konvergenzordnung gilt

$$e^z - R(z) = \mathcal{O}(h^{p+1}) = \mathcal{O}(z^{p+1}).$$

□

Bemerkung 3.7. Als direkte Konsequenz aus dem Satz 3.5 folgt, dass die Stabilitätsfunktion aller expliziten Runge-Kutta Verfahren mit der Ordnung $p = s$ gegeben ist durch

$$R(z) = 1 + z + \frac{z^2}{2!} + \ldots + \frac{z^s}{s!}.$$

In der Abbildung 3.15b sind die Stabilitätsbereiche für $s = 1, \ldots, 4$ dargestellt. ⌟

Stabilitätsanalyse für implizite Runge-Kutta Verfahren

Wir starten wieder mit dem impliziten Euler Verfahren. Das Verfahren lautet $y_1 = y_0 + h\, f(x_1, y_1)$, angewandt auf die Dahlquist Gleichung (3.37) liefert $y_1 = y_0 + h\,\lambda\, y_1$. Mit $z = \lambda\, h$ folgt

$$R(z) = \frac{1}{1 - z}.$$

In dem Fall erhalten wir als Stabilitätsbereich das Äussere des Einheitskreises im Punkt $(1, 0)$. Der Stabilitätsbereich umfasst damit die negative Halbebene und grosse Bereiche der positiven Halbebene. Das implizite Euler Verfahren ist daher *sehr* stabil.

Für das s-stufige Runge-Kutta Verfahren gilt der Satz:

Satz 3.6. *Das s-stufige Runge-Kutta Verfahren (3.17) angewandt auf die Dahlquist Testgleichung (3.37) liefert* $y_1 = R(h\,\lambda) y_0$ *mit*

$$R(z) = 1 + z\, b^T \cdot (\mathbb{1} - z\, A)^{-1} \cdot \mathbf{1}, \tag{3.39}$$

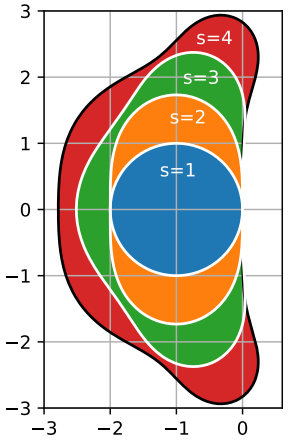
wobei $b \in \mathbb{R}^s, A \in \mathbb{R}^{s \times s}$ *durch das zugehörige Butcher-Schema (3.16) gegeben sind und* $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^s$.

Beweis. Aufgrund der Linearität der Dahlquist Testgleichung folgt, dass das Gleichungssystem (3.17a) für $k_j, j = 1, \ldots, s$ ebenfalls linear ist und somit $\boldsymbol{k} = (\mathbb{1} - z\, A)^{-1} \cdot \mathbf{1}$. Aus dem Integrationsschritt (3.17b) folgt die Behauptung. □

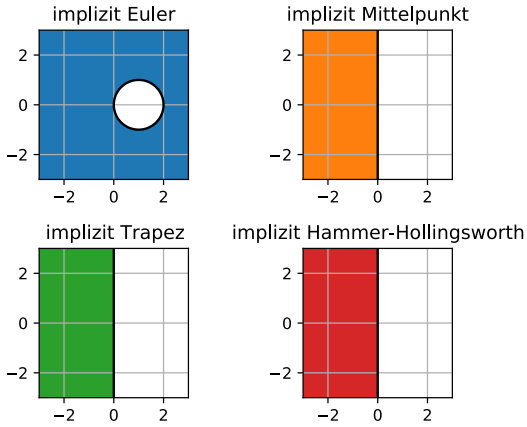
Direkt aus der Cramer'schen Regel folgt der Korrolar

Korrolar 3.2. *Für die Stabilitätsfunktion (3.39) gilt*

$$R(z) = \frac{\det(\mathbb{1} - zA + z\mathbf{1} \cdot b^T)}{\det(\mathbb{1} - zA)}.$$



(a) Explizite Runge-Kutta Methoden der Ordnung s.



(b) Implizite Runge-Kutta Methoden.

Abbildung 3.15: Stabilitätsbereiche [Jupyter-Notebook]

Kapitel 4

Einführung partielle Differentialgleichungen

Bis jetzt haben wir uns mit numerischen Methoden für gewöhnliche Differentialgleichungen betrachtet. In der Technik und Naturwissenschaft haben wir es oft mit partiellen Differentialgleichungen (PDE) zu tun.

4.1 Was ist eine partielle Differentialgleichung

Es sei D ein Gebiet in $\mathbb{R}^n$ und $\mathbf{x} = (x_1, ..., x_n) \in D$. Unter einer partiellen Differentialgleichung der Ordnung k für eine Funktion $u(\mathbf{x})$ in D versteht man eine Gleichung der Form

$$F\left(\mathbf{x}, u(\mathbf{x}), \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \ldots, \frac{\partial^k u}{\partial x_n^k}\right) = 0. \tag{4.1}$$

In u treten also mehrere unabhängige Veränderliche, nämlich $x_1, \ldots, x_n$ auf, und in (4.1) neben $\mathbf{x}$ und u partielle Ableitungen von u bis zur Ordnung k. Im Spezialfall $n = 1$ liegt mit (4.1) eine gewöhnliche Differentialgleichung vor. Die partielle Differentialgleichung heisst *linear*, wenn die Funktion $u(\mathbf{x})$ und ihre Ableitungen in Form einer Linearkombination in der Gleichung (4.1) auftreten.

Um zu eindeutig bestimmten Lösungen von (4.1) zu gelangen, sind zusätzliche Bedingungen zu stellen, etwa *Rand- und/oder Anfangsbedingungen* oder Abklingbedingungen im Unendlichen.

Wir betrachten hier ein paar ausgewählte Beispiele von (in der Physik auftretenden) partiellen Differentialgleichungen. Im Folgenden bezeichnet Δ bzw. ∇ wie üblich den Laplace- und Nablaoperator im $\mathbb{R}^n$ (vgl. Analysis 3)

$$\Delta u(\mathbf{x}) = \frac{\partial^2 u(\mathbf{x})}{\partial x_1^2} + \ldots + \frac{\partial^2 u(\mathbf{x})}{\partial x_n^2}, \quad \nabla u(\mathbf{x}) = \left(\frac{\partial u(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial u(\mathbf{x})}{\partial x_n}\right)^T.$$

Transport 1. Ordnung, linear

$$u_x + u_y = 0 \tag{4.2a}$$

Transport 1. Ordnung, linear

$$u_x + y u_y = 0 \tag{4.2b}$$

Stosswelle 1. Ordnung, nichtlinear

$$u_x + uu_y = 0 \tag{4.2c}$$

Laplace-Gleichung oder Potentialgleichung, 2. Ordnung, linear

$$u_{xx} + u_{yy} = 0 \quad \text{oder kompakt } \Delta u(\mathbf{x}) = 0 \tag{4.2d}$$

Poisson-Gleichung 2. Ordnung, linear

$$-u_{xx} - u_{yy} = f(x, y) \quad \text{oder kompakt } -\Delta u(\mathbf{x}) = f(\mathbf{x}) \tag{4.2e}$$

Wärmeleitungsgleichung 2. Ordnung, linear, eindimensional im Ort $D \subset \mathbb{R}$

$$u_t - u_{xx} = q(x) \tag{4.2f}$$

oder mehrdimensional im Ort $D \subset \mathbb{R}^n$

$$u_t - \Delta u(\mathbf{x}) = q(\mathbf{x}) \tag{4.2g}$$

Wellengleichung 2. Ordnung, linear, eindimensional im Ort $D \subset \mathbb{R}$

$$u_{tt} - u_{xx} = 0 \tag{4.2h}$$

oder mehrdimensional im Ort $D \subset \mathbb{R}^n$

$$u_{tt} - \Delta u(\mathbf{x}) = 0 \tag{4.2i}$$

Helmholtzsche Schwingungsgleichung 2. Ordnung, linear

$$u_{xx} + k^2 u(x) = 0, \quad k \in \mathbb{C} \tag{4.2j}$$

oder mehrdimensional im Ort $D \subset \mathbb{R}^n$

$$\Delta u(\mathbf{x}) + k^2 u(\mathbf{x}) = 0, \quad k \in \mathbb{C} \tag{4.2k}$$

Schwingender Stab 4. Ordnung, linear, eindimensional im Ort $D \subset \mathbb{R}$

$$u_{tt} - u_{xxxx} = 0 \tag{4.2l}$$

oder mehrdimensional im Ort $D \subset \mathbb{R}^n$

$$u_{tt} - \Delta^2 u(\mathbf{x}) = 0(\mathbf{x}) \tag{4.2m}$$

$\Delta^2 u(\mathbf{x}) = \Delta\Delta u(\mathbf{x})$ nennt man den Biharmonischen Operator.

Maxwell Gleichungen System 1. Ordnung $D \subset \mathbb{R}^3$

$$\begin{aligned}
\tfrac{\partial \mathbf{B}}{\partial t} + \operatorname{rot} \mathbf{E} &= 0 && \text{Faraday's Gesetz} \\
\tfrac{\partial \mathbf{D}}{\partial t} + \mathbf{J} &= \operatorname{rot} \mathbf{H} && \text{Maxwell-Ampère Gesetz} \\
\operatorname{div} \mathbf{D} &= \rho && \text{Gauss'sches Gesetz} \\
\operatorname{div} \mathbf{B} &= 0 && \text{magnetisches Gauss'sches Gesetz} \\
\mathbf{B} = \mu \mathbf{H},\ \mathbf{D} &= \epsilon \mathbf{E} && \text{Material Gesetz}
\end{aligned}$$

Navier-Stokes Gleichungen System 2. Ordnung $D \subset \mathbb{R}^3$

$$-\Delta \mathbf{u} + Re\,(\mathbf{u}\,\nabla)\mathbf{u} - \operatorname{grad} p = \mathbf{f}$$
$$\operatorname{div} \mathbf{u} = 0$$

Die Liste ist natürlich bei weitem nicht vollständig! Es zeigt sich, dass die Theorie der partiellen Differentialgleichungen ein recht umfangreiches mathematisches Gebiet darstellt, bei dem sehr unterschiedliche Verfahren und Methoden Verwendung finden und das auch unter Gesichtspunkten der mathematischen Forschung Aktualität besitzt. Dies macht die Entwicklung in den letzten Jahrzehnten deutlich.

Im Folgenden gehen wir auf zwei Beispiele etwas vertieft ein, wobei wir uns auf die Betrachtung eindimensionaler Gebiete beschränken.

4.2 Eindimensionale Wärmeleitung

Für die meisten Probleme in der Praxis kann man keine analytische Lösungsformel der PDE berechnen oder sie ist so kompliziert, dass man es vorzieht eine typische Lösung als Graphen zu veranschaulichen. Benutzt man Numerische Methoden, so müssen diese sorgfältig gewählt werden. Es ist möglich dass die Methode die (wahre) Lösung - falls sie dann auch existiert(!) - nicht überall annähert. Die andere Gefahr besteht darin, dass die Berechnung mehr Rechenzeit und Ressourcen (Memory, Disk Space) beansprucht, als in der Praxis zur Verfügung gestellt werden kann.

Ziel dieses Kapitel ist eine kurze Einführung und Illustration der oben erwähnten Punkte zu geben.

4.2.1 Finite Differenzenmethode

$$u_j \approx u(j\,h).$$

Die drei üblichen Approximationen für die *erste Ableitung* können wir wie folgt definieren:

Definition 4.1 (Approximation der ersten Ableitung).

$$\text{Rückwärtsdifferenz:} \quad \frac{\partial u}{\partial x}(j\,h) \approx \frac{u_j - u_{j-1}}{h} \tag{4.3a}$$
$$\text{Vorwärtsdifferenz:} \quad \frac{\partial u}{\partial x}(j\,h) \approx \frac{u_{j+1} - u_j}{h} \tag{4.3b}$$
$$\text{zentrale Differenz:} \quad \frac{\partial u}{\partial x}(j\,h) \approx \frac{u_{j+1} - u_{j-1}}{2h} \tag{4.3c}$$

Jede von ihnen ist eine korrekte Approximation, denn nach der Taylorentwicklung ist

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + O(h^4).$$

(Sie ist gültig, wenn $u(x)$ eine vier mal stetig differenzierbare Funktion, sprich: C^4-Funktion ist.) Ersetzt man h mit $-h$, so folgt

$$u(x-h) = u(x) - u'(x)h + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 + O(h^4).$$

Aus beiden Ausdrücken folgt

$$\begin{aligned} u'(x) &= \frac{u(x) - u(x-h)}{h} + O(h) \\ u'(x) &= \frac{u(x+h) - u(x)}{h} + O(h) \\ u'(x) &= \frac{u(x+h) - u(x-h)}{2h} + O(h^2). \end{aligned} \tag{4.4}$$

Unter $O(h)$ versteht man ein Ausdruck, der durch eine mit h multiplizierte Konstante beschränkt ist. Ersetzt man x durch $j\,h$, so folgt, dass (4.3a) und (4.3b) korrekte Approximationen erster Ordnung $O(h)$ und (4.3c) eine Approximation zweiter Ordnung $O(h^2)$ ist.

Für die *zweite Ableitung* ist die einfachste Approximation die zentrale zweite Differenz

Definition 4.2 (Approximation der zweiten Ableitung).

$$\text{zentrale zweite Differenz} \quad \frac{\partial^2 u}{\partial x^2}(j\,h) \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2}. \tag{4.5}$$

Siehe auch (2.12). Diese Approximation wird dadurch gerechtfertigt, dass dieselben beiden Taylorreihen wie oben nach Addition die Gleichung liefern

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2).$$

Das heisst, (4.5) gilt mit einem Fehler zweiter Ordnung $O(h^2)$.

Für Funktionen zweier Variablen $u(t, x)$ wählen wir eine Schrittweite in beiden Variablen. Wir schreiben

$$u(n\ \tau, j\ h) \approx u_{n,j}.$$

Dann können wir beispielsweise $\partial u/\partial t$ durch die Vorwärtsdifferenz

$$\frac{\partial u}{\partial t}(n\ \tau, j\ h) \approx \frac{u_{n+1,j} - u_{n,j}}{\tau}$$

approximieren.

Beispiel 4.1. Wir lösen die Diffusionsgleichung

$$\begin{aligned} u_t &= u_{xx} \quad \text{für } x \in (0, \pi) \\ u(0, x) &= \phi(x) \quad \text{und} \\ u(t, 0) &= u(t, \pi) = 0 \end{aligned} \tag{4.6}$$

(vereinfachte Gleichung der Wärmeleitung) mit Hilfe endlicher Differenzen. Für u_t verwenden wir die vorwärts genommene Differenz, für u_{xx} die zentrale zweite. Die *Differenzengleichung* lautet dann

$$\frac{u_{n+1,j} - u_{n,j}}{\tau} = \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2} \quad \text{für alle } j = 1, \ldots, N-1, \tag{4.7}$$

wobei N durch die Bedingung $\pi = N\,h$ gegeben ist. Es existiert daher für alle inneren Punkte eine Differenzengleichung. Sie enthält Abschneidefehler der Grössenordnung $O(\tau)$ auf der linken und $O(h^2)$ auf der rechten. Daher wählen wir einen kleinen Wert für h und setzen $\tau = h^2$. Die Differenzengleichung (4.7) vereinfacht sich zu

$$u_{n+1,j} = u_{n,j+1} - u_{n,j} + u_{n,j-1}. \tag{4.8}$$

Die Anfangsbedingung $\phi(x)$ soll eine ganz einfache Rechteckfunktion sein

$$(\phi(j\ h))_{j=1\ldots N} = (0,\ldots,0,\underset{\uparrow}{1},0,\ldots,0)$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \tfrac{N}{2}$$

für N ungerade. Damit erhalten wir ein entsetzliches Ergebnis, siehe Tabelle 4.1.

	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$
n = 0	0	0	0	0	0	1	0	0	0	0	0
n = 1	0	0	0	0	1	-1	1	0	0	0	0
n = 2	0	0	0	1	-2	3	-2	1	0	0	0
n = 3	0	0	1	-3	6	-7	6	-3	1	0	0
n = 4	0	1	-4	10	-16	19	-16	10	-4	1	0

Tabelle 4.1: Lösung der expliziten Differenzengleichung (4.8)

Bemerkung 4.1 (Dirichlet-Randbedingungen). Für alle inneren Punkte existiert eine Differenzengleichung (4.7). Für die Randpunkte ist keine Gleichung gegeben. Die Information muss daher durch Randbedingungen vorgegeben werden. Mit *Dirichlet-Randbedingungen* bezeichnet man die Ausgangslage, dass die Funktionswerte vorgegeben sind. Allgemeine Dirichlet-Randbedingungen für das Problem (4.6) sind daher gegeben durch

$$u(t,0) = g(t) \quad \text{und} \quad u(t,\pi) = h(t).$$

Der Spezialfall im Beispiel $g(t) = h(t) \equiv 0$ bezeichnet man auch als *homogene* Dirichlet-Randbedingungen.

4.2.2 Approximation der Diffusionsgleichung

Analysieren wir, was im Beispiel 4.1 für die Diffusionsgleichung $u_t = u_{xx}$ schief läuft. Im Verfahren ist nichts offensichtlich falsch, denn jede Ableitung wurde mit kleinem Abschneidefehler angemessen approximiert. Die kleinen Fehler haben sich aufgeschaukelt. Das Problem liegt in der Wahl des Verhältnisses der Schrittweite h (bezüglich dem Ort) und τ (bezüglich der Zeit) zueinander. Betrachten wir daher das Verhältnis

$$s = \frac{\tau}{h^2}.$$

Analog wie im Beispiel folgt die explizite Differenzengleichung

(**F**orward **T**ime **C**entered **S**pace Scheme)

$$u_{n+1,j} = s(u_{n,j+1} + u_{n,j-1}) + (1-2s)u_{n,j} \quad \forall\, j = 1, \dots N-1. \tag{4.9}$$

Das Verfahren nennt man explizit, da sich jeder Wert des Zeitschrittes $(n+1)$ explizit aus den alten berechnen lässt.

Beispiel 4.2. Um konkret zu sein, betrachten wir das selbe Problem (4.6), jedoch mit

$$\phi(x) = \begin{cases} x & \text{für } 0 < x < \frac{\pi}{2} \\ \pi - x & \text{für } \frac{\pi}{2} \le x < \pi. \end{cases}$$

Für diese Anfangsbedingung lautet die exakte Lösung

$$u(t,x) = \frac{4}{\pi} \sum_{k=1}^{\infty} b_k \sin(kx)\, e^{-k^2 t}$$

mit

$$b_k = \begin{cases} \frac{(-1)^{(k-1)/2}}{k^2} & \text{für } k \text{ ungerade} \\ 0 & \text{für } k \text{ gerade.} \end{cases}$$

Wir approximieren dieses Problem durch das Verfahren (4.9) für $j = 1, \dots, J-1$ und $n = 0, 1, 2, \dots$ zusammen mit den diskretisierten Anfangs- und Randbedingungen

$$u(0, j\,h) = \phi(j\,h) \quad \text{und } u(n\,\tau, 0) = u(n\,\tau, \pi) = 0.$$

In der Abbildung 4.1 ist das Resultat zu sehen (vgl. [Jupyter-Notebook]). Für $J = 20, h = \pi/20$ und $s = 5/11$ sieht das Resultat wie gewünscht aus. Wenn wir jedoch $J = 20, h = \pi/20$ und $s = 5/9$ wiederholen, sieht das Ergebnis wild oszillierend aus. Bei der Wahl $s = 5/11$ ist das Verhalten stabil, bei $s = 5/9$ offenbar instabil.

⌟

4.2.3 Stabilitätskriterium

Der Unterschied der beiden Lösungen im Beispiel 4.2 liegt darin, dass im ersten Fall $s < 1/2$ und im zweiten $s > 1/2$ gilt. Ein Hinweis auf diesen Unterschied ist direkt in der Gleichung (4.9) sichtbar: nur für $s < 1/2$ sind die Koeffizitenten positiv. Den Effekt untersuchen wir hier jedoch etwas präziser. Dazu suchen wir nach einer Lösung der Gleichung (4.9) in der Form

$$u_{n,j} = X_j\, T_n. \tag{4.10}$$

Damit folgt

$$\frac{T_{n+1}}{T_n} = 1 - 2s + s\frac{X_{j+1} + X_{j-1}}{X_j}.$$

Beide Seiten von (4.10) müssen mit einer von j und n unabhängigen Konstanten ξ übereinstimmen. Damit folgt

$$T_n = \xi^n T_0$$

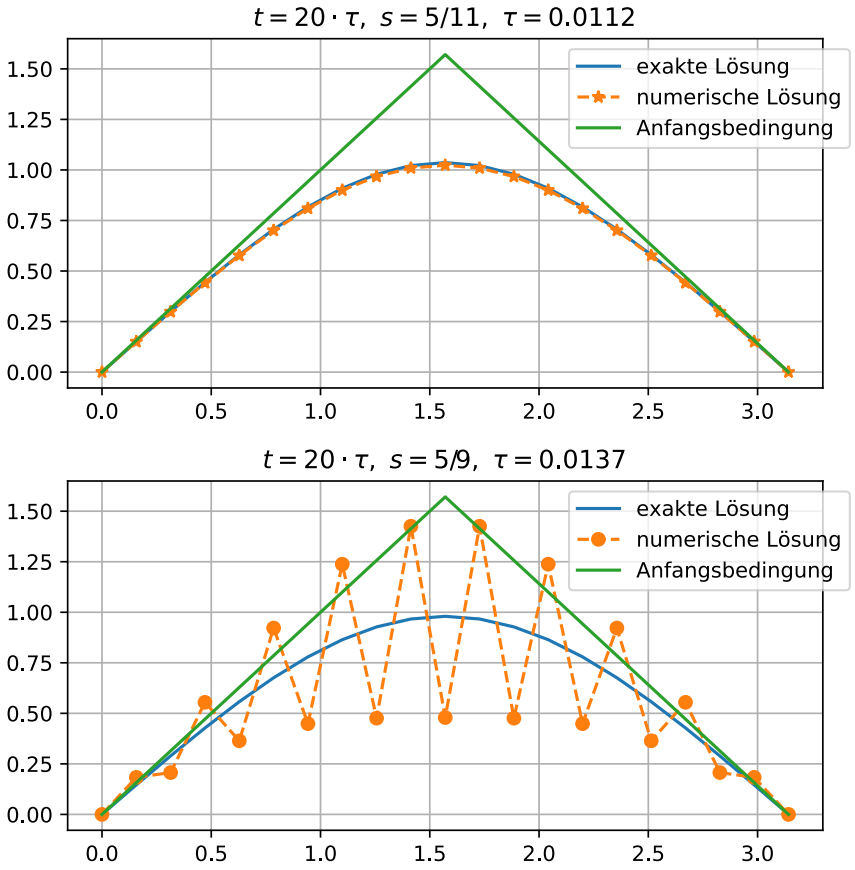


Abbildung 4.1: Stabilität des FTCS-Scheme

sowie

$$\xi = 1 - 2s + s\frac{X_{j+1} + X_{j-1}}{X_j}. \tag{4.11}$$

Für den Ansatz

$$X_j = A\cos(j\ \theta) + B\sin(j\ \theta)$$

folgt aus der linken Randbedingung $X_0 = 0$ sofort $A = 0$, Für B setzen wir $B = 1$. Aus der rechten Randbedingung $X_J = 0$ folgt $\sin(J\,\theta) = 0$, somit ist $J\,\theta = k\,\pi$ mit k ganzzahlig. Die Diskretisierung zerlegt das Intervall $(0, \pi)$ in J gleichgrosse Teilintervalle der Länge h. Das bedeutet $J = \pi/h$ und somit $\theta = k\,h$. Für X_j folgt

$$X_j = \sin(j\,k\,h).$$

Für (4.11) erhalten wir

$$s\ \frac{\sin((j+1)\,k\,h) - \sin((j-1)\,k\,h)}{\sin(j\,k\,h)} + 1 - 2\,s = \xi.$$

Für h klein interpretieren wir den Quotient

$$2\,\frac{\sin((j+1)\,k\,h) - \sin((j-1)\,k\,h)}{2\sin(j\,k\,h)} \approx 2\ \cos(k\,h)$$

als die Ableitung von sin an der Stelle $k\,h$. Da die Differenz zwischen $(j+1)$ und $(j-1)$ gemacht wird, kommt ein Faktor 2 dazu. Wir erhalten damit für ξ

$$\xi = \xi(k) = 1 - 2s(1 - \cos(k\,h)). \tag{4.12}$$

Da für T_n das zeitliche Wachstum $t = n\,\tau$ durch die Potenz $\xi(k)^n$ bestimmt wird, folgt sofort

$$|\xi(k)| \leq 1 \quad \text{für alle } k$$

als Stabilitätsbedingung und daraus

$$\frac{\tau}{h^2} = s \leq \frac{1}{2}. \tag{4.13}$$

Bemerkung 4.2. Der Ansatz $X_j = A\cos(j\theta) + A\sin(j\theta)$ kann auch einfach durch den Ansatz

$$X_j = (e^{i\,k\,h})^j$$

ersetzt werden. Für ξ folgt in dem Fall

$$\xi = 1 - 2s\left(1 - \frac{e^{i\,k\,h} + e^{-i\,k\,h}}{2}\right)$$

analog zu (4.12) und damit dasselbe Resultat. ⌟

4.2.4 Neumannsche Randbedingung

Wir nehmen an, die Diffusionsgleichung ist im Intervall $0 \leq x \leq \ell$ definiert und die Randbedingungen sind

$$\frac{\partial u}{\partial x}(t,0) = g_0(t) \quad \text{und} \quad \frac{\partial u}{\partial x}(t,\ell) = g_1(t).$$

Die einfachste Approximation ist

$$\frac{u_{n,1} - u_{n,0}}{h} = g_0(t_n) \quad \text{und} \quad \frac{u_{n,J} - u_{n,J-1}}{h} = g_1(t_n).$$

Da der Abschneidefehler für den Differenzenquotienten für die erste Ableitung eine lineare Ordnung hat, würden wir die quadratische Konvergenzordnung der zweiten Ableitung mit der zentralen zweiten Differenz zerstören (vgl. (4.4)). Der Ausweg besteht in der Einführung sogenannter *Geisterpunkte* $u_{n,-1}$ und $u_{n,J+1}$. Die diskrete Randbedingungen lauten dann

$$\frac{u_{n,1} - u_{n,-1}}{2h} = g_0(t_n) \quad \text{und} \quad \frac{u_{n,J+1} - u_{n,J-1}}{2h} = g_1(t_n). \tag{4.14}$$

Im n-ten Zeitschritt werden die Werte $u_{n,0}, \ldots, u_{n,J}$ berechnet und anschliessend die Werte der Geisterpunkte mit (4.14) bestimmt. Die Abbildung 4.2 zeigt die Lösung für das Beispiel 4.2 jedoch mit homogenen ($g_0(t) = g_1(t) \equiv 0$) Neumann Randbedingungen.

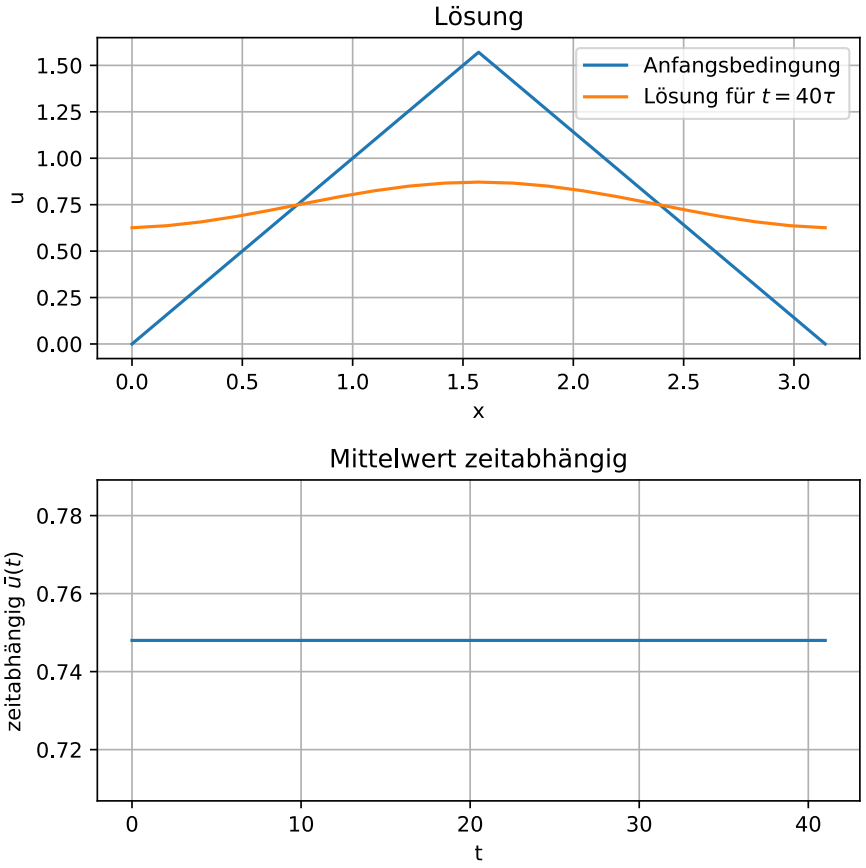


Abbildung 4.2: Lösung für das Beispiel 4.2 mit Neumann Randbedingungen.

4.2.5 Crank-Nicolson-Verfahren

Praktisch bedeutet die Stabilitätsbedingung (4.13), dass die Zeitschritte sehr klein gehalten werden müssen. Trifft man beispielsweise $h = 10^{-2}$, so muss $\tau \leq \frac{1}{2} \cdot 10^{-4}$ sein. Die Lösung zum Zeitpunkt $t = 1$ zu bestimmen, würde damit 20'000 Zeitschritte erfordern. Das ist für Praxis oft nicht brauch bar. Gesucht ist daher ein Verfahren, das keine Stabilitätsbedingung mit sich bringt. Es gibt davon eine ganze Klasse die unabhängig vom Wert s stabil sind.

Für die Herleitung definieren wir den Operator $(\delta^2 u)_{n,j}$

$$(\delta^2 u)_{n,j} = \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2}$$

für die zentrale zweite Differenz ein und betrachten das Differenzenschema

θ-Schema

$$\frac{u_{n+1,j} - u_{n,j}}{\tau} = (1-\theta)\,(\delta^2 u)_{n,j} + \theta(\delta^2 u)_{n+1,j}, \quad \forall\, j = 1, \ldots N-1 \tag{4.15}$$

mit $\theta \in [0, 1]$.

Für $\theta = 0$ folgt direkt das erwähnte Schema (4.9), für $\theta > 0$ heisst das Schema *implizit*, da der neue Zeitschritt $u_{n+1,j}$ auf beiden Seiten der Gleichung auftritt. Auch hier lässt

sich die Stabilität analysieren, man macht einen separierten Ansatz für die Lösung

$$u_{n,j} = (e^{ikh})^j \xi(k)^n$$

wie zuvor. In dem Fall folgt

$$\xi(k) = \frac{1 - 2(1-\theta)s(1-\cos(kh))}{1 + 2\theta s(1-\cos(kh))},$$

mit $s = \tau/h^2$. Wir erhalten für $|\xi(k)| \leq 1$ die

Stabilitätsbedingung

$$\frac{1}{2} \leq \theta < 1$$

unabhängig von s.

Ein solches Verfahren nennt man *uneingeschränkt stabil.* Für $\theta = \frac{1}{2}$ heisst das Verfahren

Definition 4.3 (Crank-Nicolson-Verfahren).

$$\frac{u_{n+1,j} - u_{n,j}}{\tau} = \frac{1}{2}\left((\delta^2 u)_{n,j} + (\delta^2 u)_{n+1,j}\right) \quad \forall\, j = 1, \ldots N-1 \tag{4.16}$$

mit

$$(\delta^2 u)_{n,j} = \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2}. \tag{4.17}$$

Bemerkung 4.3. Das Crank-Nicolson-Verfahren ein Verfahren zweiter Ordnung analog zur impliziten Trapezmethode 3.12. Das Verfahren kann auch mit anderer räumlicher Diskretisierung (4.17) kombiniert werden. ⌟

Bemerkung 4.4. Ist $\theta < \frac{1}{2}$, so ist $s \leq (2-4\theta)^{-1}$ eine notwendige Bedingung für die Stabilität. Man erwartet also, dass (4.15) stabil ist, wenn

$$\frac{\tau}{h^2} = s < \frac{1}{2-4\theta}$$

gilt. ⌟

Beispiel 4.3 (Zweidimensionale Poisson Gleichung). Als abschliessendes Beispiel und einen Vorgeschmack für das Modul *Höhere Analysis und Numerik* sei hier auf ein Jupyter-Notebook für die zweidimensionale Poisson Gleichung verwiesen. ⌟

Kapitel 5

Splines

5.1 Splinefunktionen

Die Polynominterpolation liefert bei vielen äquidistanten Stützstellen in der Regel keine befriedigende Lösung im Sinne einer guten Approximation der interpolierten Funktion, weil Polynome hohen Grades zu starken Oszillationen neigen. Splinefunktionen, welche in diesem Kapitel eingeführt werden, bilden ein sehr viel geeigneteres Hilfsmittel als Polynome, um grössere Datenmengen an beliebigen Stützstellen zu interpolieren.

Das Wort „Spline“ bedeutet so viel wie „dünne Holzlatte“, früher wurde im Englischen ein biegsames Lineal damit benannt, das zum Zeichnen glatter Kurven verwendet wurde. Dabei sollte gerade die Minimierung der Biegeenergie Oszillationen unterdrücken.

5.1.1 Beispiel einer Splineinterpolation

Sei $\tau = \{x_0, \ldots, x_n\}$ eine Stützstellenmenge mit

$$a = x_0 < x_1 < \ldots < x_n = b$$

und

$$f_j, \quad j = 0, 1, \ldots, n,$$

die entsprechenden Daten an diesen Stützstellen.

Definition 5.1. Der Raum der *kubischen Splines* sei gegeben durch

$$\mathbb{P}_{3,\tau} := \{g \in C^2([a,b]) \mid g|_{[x_i,x_{i+1}]} \in \Pi_3,\ i = 0, 1, \ldots, n-1\}.$$

Eine Funktion in $\mathbb{P}_{3,\tau}$ ist in jedem Teilintervall ein Polynom vom Grad 3 und ist an jeder Stützstelle zweimal stetig differenzierbar. Solche Splines sind also stückweise polynominale Funktionen, die an den Stützstellen noch gewisse Glattheitseigenschaften haben.

Die Aufgabe besteht darin, zu den Daten $f_0, f_1, \ldots, f_n$ eine Funktion $S \in \mathbb{P}_{3,\tau}$ zu finden so, dass

$$S(x_j) = f_j, \quad , j = 0, 1, \ldots, n \tag{5.1a}$$

erfüllt ist. Damit die Aufgabe eine eindeutige lösung hat, muss jedoch eine weitere Bedingung gestellt werden: z.B. die natürlichen Endbedingungen

$$S''(a) = S''(b) = 0. \tag{5.1b}$$

Die Lösung $S \in \mathbb{P}_{3,\tau}$ der Gleichungen (5.1) hat folgende interessante Extremaleigenschaften

Korrolar 5.1. *Sei g eine beliebige Funktion aus $C^2([a,b])$ mit $g(x_j) = f_j$, $j = 0,1,\ldots,n$ und $g''(a) = g''(b) = 0$. Für die eindeutige Lösung S von (5.1) gilt*

$$\int_a^b S''(x)^2 dx \leq \int_a^b g''(x) dx.$$

Diese Eigenschaft bedeutet, dass der kubische Spline S unter allen Funktionen $g \in C^2([a,b])$, die dieselben Interpolationsforderungen erfüllen, näherungsweise die mittlere quadratische Krümmung minimiert.

Die Krümmung ist durch

$$\kappa(x) = \frac{g''(x)}{(1+g'(x)^2)^{3/2}}$$

definiert. Dies liefert die mittlere quadratische Krümmung

$$\|\kappa\|_2^2 = \int_a^b \kappa(x)^2 dx \approx \int_a^b g''(x) dx$$

unter der Annahme, dass $|g'(x)| \ll 1$ für alle $x \in [a,b]$.

Bemerkung 5.1. Anstelle der *natürlichen* Bedingung (5.1b) wird auch die *periodische* Bedingung

$$S(a) = S(b),\ S'(a) = S'(b),\ S''(a) = S''(b)$$

benutzt. ⌟

Berechnung der Spline

Der Einfachheit halber betrachten wir den Fall mit äquidistanten Stützstellen, dh. $x_{j+1} - x_j = h$ $j = 0,\ldots,n-1$. Man führt die Bezeichnung

$$m_j := S''(x_j), \quad j = 0,1,\ldots n,$$

und

$$I_j := [x_j, x_{j+1}], \quad j = 0,1,\ldots,n-1$$

ein. Wegen $S|_{I_j} \in \Pi_3$ folgt, dass $S''|_{I_j}$ linear ist und dass

$$S''|_{I_j}(x) = \frac{(x_{j+1}-x)\,m_j + (x-x_j)\,m_{j+1}}{h}$$

gilt. Zweifache Integration zusammen mit den Forderungen

$$S(x_j) = f_j, \quad S(x_{j+1}) = f_{j+1}$$

liefert

$$S|_{I_j}(x) = \frac{(x_{j+1}-x)^3 m_j + (x-x_j)^3 m_{j+1}}{6h} + \frac{(x_{j+1}-x)f_j + (x-x_j)f_{j+1}}{h} - \frac{1}{6}h((x_{j+1}-x)m_j + (x-x_j)m_{j+1}). \tag{5.2}$$

Das stückweise Polynom S aus (5.2) hat die gewünschten Eigenschaften $S|_{I_j} \in \Pi_3$, $S \in C([a,b])$, $S''|_{I_j}(x_{j+1}) = m_{j+1} = S''|_{I_{j+1}}(x_{j+1})$.

Man muss nun die noch unbekannten Grössen m_j so wählen, dass die erste Ableitung von S in den Stützstellen x_j stetig ist. Es soll also

$$S'|_{I_{j-1}}(x_j) = S'|_{I_j}(x_j), \quad j = 1, \ldots, n-1$$

gelten. Mit Hilfe der Darstellung (5.2) erhält man daraus die Bedingungen

$$m_{j-1} + 4m_j + m_{j+1} = \frac{6}{h^2}(f_{j-1} - 2f_j + f_{j+1}), \quad j = 1, 2, \ldots n-1.$$

Wegen (5.1b) gilt

$$m_0 = m_n = 0.$$

Es ergibt sich das System

$$\begin{pmatrix} 4 & 1 & & & 0 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ 0 & & & 1 & 4 \end{pmatrix} \begin{pmatrix} m_1 \\ \\ \vdots \\ \\ m_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} f_0 - 2f_1 + f_2 \\ \\ \vdots \\ \\ f_{n-2} - 2f_{n-1} + f_n \end{pmatrix}. \tag{5.3}$$

Die gesuchte Lösung $S \in \mathbb{P}_{3,\tau}$ hat die Darstellung (5.2), wobei $m_0 = m_n = 0$ und $(m_1, \ldots, m_{n-1})$ die Lösung des Gleichungssystems (5.3) ist.

Beispiel 5.1. In der Abbildung 5.1 sind die Splineinterpolationen für $n = 7, 9, 15$ für

$$f(x) = \frac{1}{1+x^2}$$

zu sehen. Im Gegensatz zur Polynominterpolation (nicht in diesem Modul behandelt) wird die Splineinterpolation mit wachsender Anzahl Stützstellen besser. ⌟

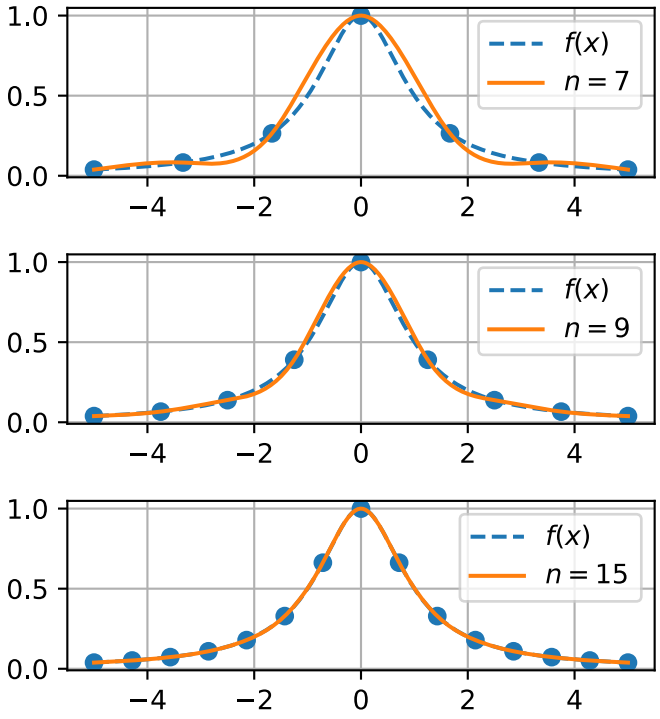


Abbildung 5.1: Splineinterpolationen für $f(x) = \frac{1}{1+x^2}$

5.2 B-Splines (optional)

5.2.1 Einführung

Bei der Behandlung von Splines ist es bequemer, statt mit dem Grad von Polynomen, mit der Ordnung $k := \text{Grad} + 1$ (Anzahl Freiheitsgrade) zu arbeiten. Die Bruchstellen zwischen den polynomialen Abschnitten werden häufig als Knoten bezeichnet. Für eine Knotenmenge $\tau = \{\tau_0, \ldots, \tau_{\ell+1}\}$ mit

$$a = \tau_0 < \tau_1 < \ldots < \tau_\ell < \tau_{\ell+1} = b$$

und $k \geq 1$ definieren wir den Splineraum der Splines der Ordnung k durch

Definition 5.2. Splineraum der Ordnung k

$$\mathbb{P}_{1,\tau} = \{f : [a,b) \to \mathbb{R} \mid f|_{[\tau_i,\tau_{i+1})} \in \Pi_0,\ 0 \leq i \leq \ell\}$$
$$\mathbb{P}_{k,\tau} = \{f \in C^{k-2}([a,b]) \mid f|_{[\tau_i,\tau_{i+1})} \in \Pi_{k-1},\ 0 \leq i \leq \ell\},\ k \geq 2.$$

Eine geeignete Basis für $\mathbb{P}_{k,\tau}$ bilden die sogenannten B-Splines. Der Raum $\mathbb{P}_{1,\tau}$ besteht aus den stückweise Konstanten. Die charakteristischen Funktionen bilden eine Basis

$$N_{j,1}(x) = \chi_{[\tau_j,\tau_{j+1})}(x) := \begin{cases} 1 & \text{für } x \in [\tau_j, \tau_{j+1}) \\ 0 & \text{sonst,} \end{cases} \qquad j = 0, \ldots, \ell,$$

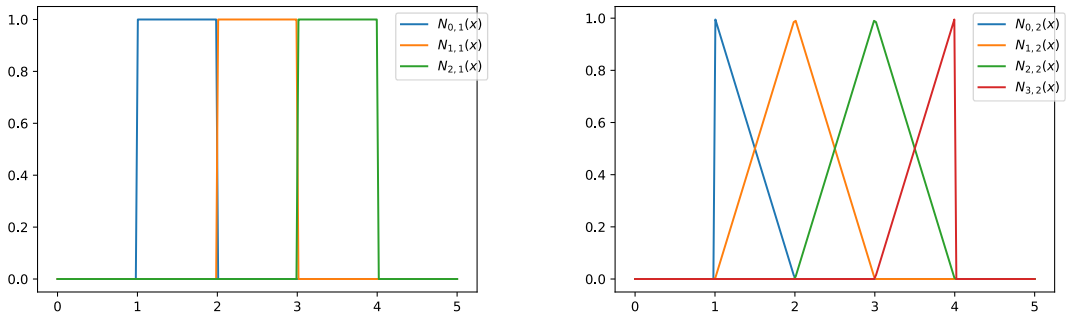
für $\mathbb{P}_{1,\tau}$. Jede stückweise konstante Funktion aus $\mathbb{P}_{1,\tau}$ lässt sich als Linearkombination der $N_{j,1}$ schreiben

$$S(x) = \sum_{j=0}^{\ell} c_j N_{j,1}(x).$$

Für den Fall $k = 2$, der stückweise linearen Funktionen (sogenannte Polygonzüge), werden zwei Hilfsknoten $\tau_{-1} < \tau_0 = a$, $\tau_{\ell+2} > \tau_{\ell+1} = b$ und zugehörige Hilfsfunktionen $N_{-1,1}$, $N_{\ell+1}$ eingeführt. Diese Hilfsfunktioenn sind die charakteristischen Funktionen $N_{-1,1} = \chi_{[\tau_{-1},\tau_0)}$ und $N_{\ell+1} = \chi_{[\tau_{\ell+1},\ell+2)}$. Damit können wir die Funktionen

$$N_{j,2}(x) = \frac{x - \tau_j}{\tau_{j+1} - \tau_j} N_{j,1}(x) + \frac{\tau_{j+2} - x}{\tau_{j+2} - \tau_{j+1}} N_{j+1,1}(x), \quad j = -1, \ldots, \ell$$

definieren, welche eine Basis für $\mathbb{P}_{2,\tau}$, den stückweise linearen Funktionen bilden (vgl. Abb. 5.2).



(a) Basis für $\mathbb{P}_{1,\tau}$, $\tau = (1, 2, 3, 4)$ (b) Basis für $\mathbb{P}_{2,\tau}$, $\tau = (1, 2, 3, 4)$

Abbildung 5.2: Beispiel für Basisfunktionen von $\mathbb{P}_{1,\tau}$, $\mathbb{P}_{2,\tau}$. Siehe auch Jupyter-Notebook

Für den allgemeinen Fall werden die Basisfunktionen rekursiv definiert:

Definition 5.3 (B-Splines). Sei $t_1 < t_2 < \ldots < t_n$ eine beliebige Folge von paarweise verschiedenen Knoten. Dann werden die B-Splines $N_{j,k}$ der Ordnung k ($1 \leq k \leq n$) rekursiv definiert durch

$$\begin{aligned} N_{j,1}(x) :=& \chi_{[t_j,t_{j+1})}(x) \quad \text{für } j = 1, \ldots, n-1, \\ N_{j,k}(x) :=& \frac{x - t_j}{t_{j+k-1} - t_j} N_{j,k-1}(x) + \frac{t_{j+k} - x}{t_{j+k} - t_{j+1}} N_{j+1,k-1}(x), \\ & \text{für } k = 2, \ldots, n-1, \quad \text{und} \quad j = 1, \ldots, n-k. \end{aligned} \tag{5.4}$$

Beispiel 5.2 (Jupyter-Notebook). Für das Beispiel aus der Abbildung 5.2 mit dem Knotenset $\tau = (1, 2, 3, 4)$ folgt aus der rekursiven Definition, dass die B-Spline $N_{2,3}(x)$ mit Ordnung 3 sich über das gesamte Intervall $[1, 4)$ erstreckt. Die Ordnung 3 ist damit die maximale Ordnung.

Für das Verständnis betrachten wir die B-Spline Basis $N_{2,3}(x)$ und überlegen uns welche Basisfunktionen in der Definition beteiligt sind:

$$N_{2,3}(x) = \frac{x-t_2}{t_4-t_2}\underbrace{N_{2,2}(x)}_{\frac{x-t_2}{t_3-t_2}\underbrace{N_{2,1}(x)}_{\chi_{[t_2,t_3)}(x)} + \frac{t_4-x}{t_4-t_3}\underbrace{N_{3,1}(x)}_{\chi_{[t_3,t_4)}(x)}} + \frac{t_5-x}{t_5-t_3}\underbrace{N_{3,2}(x)}_{\frac{x-t_3}{t_4-t_3}\underbrace{N_{3,1}(x)}_{\chi_{[t_3,t_4)}(x)} + \frac{t_5-x}{t_5-t_4}\underbrace{N_{4,1}(x)}_{\chi_{[t_4,t_5)}(x)}}$$

In der Abbildung 5.3 sind die Basisfunktionen graphisch dargestellt. Um eine B-Spline mit Ordnung 3, welche nur auf dem ersten Teilintervall $[1, 2)$ ungleich null ist, berechnen zu können, muss das Knotenset erweitert werden. Das selbe gilt für das letzte Teilintervall $[3, 4)$.

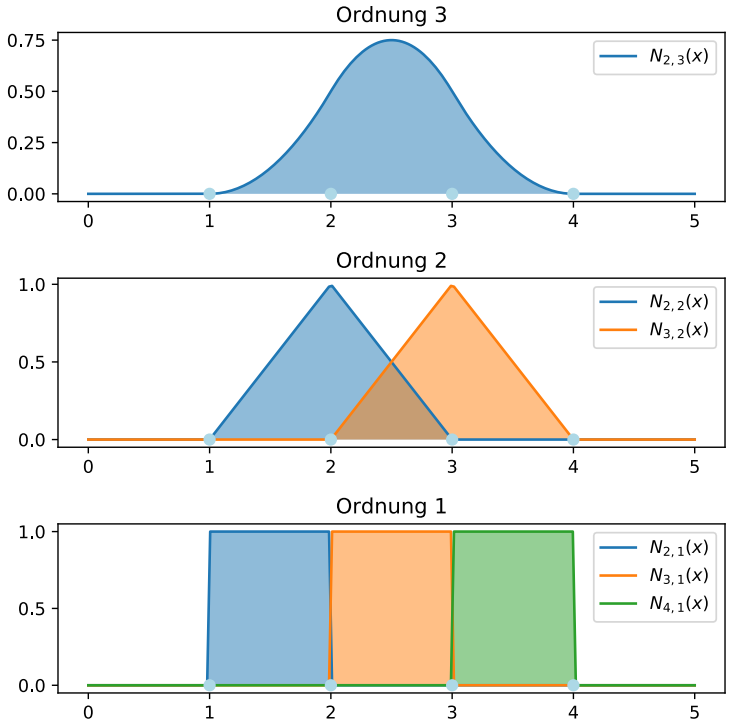


Abbildung 5.3: Rekursive Definition der B-Spline Basis $N_{2,3}(x)$ für das Knotenset $\tau = (1, 2, 3, 4)$

Bemerkung 5.2. Man kann zeigen, dass die Ableitung $N'_{j,k}(x)$ gegeben ist durch

$$N'_{j,k}(x) = (k-1)\left(\frac{N_{j,k-1}(x)}{t_{j+k-1}-t_j} - \frac{N_{j+1,k-1}(x)}{t_{j+k}-t_{j+1}}\right)$$

Die Knotenmenge

$$T = \{t_1, \ldots, t_n\} \quad \text{mit } n = 2k + \ell$$

ist die *Erweiterung* der Knotenmenge $\tau = \{\tau_0, \ldots, \tau_{\ell+1}\}$. Es gilt

$$\begin{aligned} t_1 < \ldots < t_k &= \tau_0, \\ t_{k+j} &= \tau_j, \text{ für } j = 1, \ldots, \ell \\ \tau_{\ell+1} &= t_{k+\ell+1} < \ldots < t_{2k+\ell}. \end{aligned} \tag{5.5}$$

Zu dieser erweiterten Knotenmenge T werden die B-Splines

$$N_{j,k}, \quad 1 \le j \le n - k = k + \ell$$

wie in (5.4) definiert. Jeden Spline $S \in \mathbb{P}_{k,\tau}$ kann als Linearkombination von B-Splines in der Form

$$S(x) = \sum_{j=1}^{k+\ell} c_j N_{j,k}(x) \quad x \in [a, b]$$

schreiben.

Satz 5.1. *Sei $T = \{t_j\}_{j=1}^n$ wie in (5.5). Seien $x_1, \ldots, x_{k+\ell} \in [a, b]$ Stützstellen und die zugehörigen Daten $f_1, \ldots, f_{k+\ell}$ gegeben. Das Problem der Bestimmung eines $S \in \mathbb{P}_{k,T}$ so, dass*

$$S(x_j) = f_j \quad j = 1, \ldots, k + \ell \tag{5.6}$$

gilt, hat genau dann eine eindeutige Lösung, wenn

$$x_j \in (t_j, t_{j+1}), \quad j = 1, \ldots, k + \ell,$$

daher, wenn in den Träger jedes B-Splines mindestens eine Stützstelle fällt.

Bemerkung 5.3 (Jupyter-Notebook). Unter den Voraussetzungen von Satz 5.1 ist die Matrix

$$A = (N_{j,k}(x_i))_{i,j=1}^{k+\ell}$$

regulär. $S(x) = \sum_{j=1}^{k+\ell} c_j N_{j,k}(x)$ löst (5.6) genau dann, wenn für $c = (c_1, \ldots, c_{k+\ell})^T$ und $f = (f_1, \ldots, f_{k+\ell})^T$

$$A\,c = f$$

gilt. ⌟

Bemerkung 5.4. Die Knoten t_j müssen nicht zwingend mit den Stützstellen x_j zusammenfallen. ⌟

Im Folgenden beschränken wir uns auf den Spezialfall der kubischen Splineinterpolation ($k = 4$) und dass Stützstellen und Knoten übereinstimmen

$$x_j := t_{j+3} = \tau_{j-1} \quad \text{für } j = 1, \ldots, \ell + 2.$$

Erhält man aus der Bedingung (5.6) nur $\ell + 2$ Bedingungen. Man benötigt, wie einleitend auch schon gezeigt, zwei weitere Bedingungen. Es gibt dazu unter anderem folgende Möglichkeiten

1. Vollständige kubische Splineinterpolation: Als zusätzliche Bedingung wird gefordert, dass an den Intervallenden jeweils noch die erste Ableitungen interpoliert werden, so dass der vollständige Satz von Interpolationsbedingungen durch

$$S(x_j) = f(x_j), \quad j = 1, \ldots, \ell + 2, \quad S'(a) = f'(a) \quad S'(b) = f'(b). \tag{5.7}$$

   gegeben ist.

2. Natürliche kubische Splineinterpolation:

$$S(x_j) = f(x_j) \quad j = 1, \ldots, \ell + 2, \quad S''(a) = S''(b) = 0. \tag{5.8}$$

Berechnung der vollständigen kubischen Splineinterpolation

Sei $I_4 f$ die vollständigen Splineinterpolation einer Funktion f, dann gilt

$$(I_4 f)(x) = \sum_{j=1}^{\ell+4} c_j N_{j,4}(x).$$

Ohne auf die Details einzugehen halten wir fest (vgl. [1], Seite 340ff), dass mit

$$\begin{aligned} c_1 &= f(t_4), \\ c_2 &= \frac{f'(a) - f(a)N'_{1,4}(a)}{N'_{2,4}(a)}, \\ c_{\ell+3} &= \frac{f'(b) - f(b)N'_{\ell+4,4}(b)}{N'_{\ell+3,4}(b)}, \\ c_{\ell+4} &= f(t_{\ell+5}) \end{aligned}$$

nur noch die Koeffizienten $(c_3, \ldots, c_{\ell+2})^T$ zu bestimmen sind. Wegen (5.7) muss c das Gleichungssystem

$$A_T c = f$$

erfüllen, wobei

$$f = (f_3, \ldots, f_{\ell+2})^T$$

mit

$$\begin{aligned} f_3 &= f(t_5) - c_2 N_{2,4}(t_5), \\ f_{\ell+2} &= f(t_{\ell+4}) - c_{\ell+3} N_{\ell+3,4}(t_{\ell+4}), \\ f_j &= f(t_{j+2}), \quad j = 4, \ldots, \ell + 1 \end{aligned}$$

und

$$A_T = \begin{pmatrix} N_{3,4}(t_5) & N_{4,4}(t_5) & & & & & \\ N_{3,4}(t_6) & N_{4,4}(t_6) & N_{5,4}(t_6) & & & & \\ & N_{4,4}(t_7) & N_{5,4}(t_7) & N_{6,4}(t_7) & & & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & & & N_{\ell,4}(t_{\ell+3}) & N_{\ell+1,4}(t_{\ell+3}) & N_{\ell+2,4}(t_{\ell+3}) & \\ & & & & N_{\ell+1,4}(t_{\ell+4}) & N_{\ell+2,4}(t_{\ell+4}) & \end{pmatrix}.$$

$A_T \in \mathbb{R}^{\ell \times \ell}$ ist eine Tridiagonal-Matrix, welche eine LU-Zerlegung (Gauss-Elimination ohne Pivotisierung) besitzt, so dass sich das System $A_T \cdot c = f$ sehr einfach und effizient lösen lässt (vgl. Algorithmus 2.6).

Beispiel 5.3. Wir betrachten ein Beispiel mit äquidistanten Stützstellen,

$$\tau_j = j \cdot 1/n, \quad j = 0, \ldots, n$$

und Daten $f(\tau_j)$ mit

$$f(x) = \begin{cases} 1 & x < 0.5 \\ 0 & \text{sonst.} \end{cases}$$

Die entsprechende vollständige kubische-Splineinterpolation ist in Abbildung 5.4 dargestellt. Man stellt fest, dass wegen des Sprunges in den Daten Oszillationen auftreten.

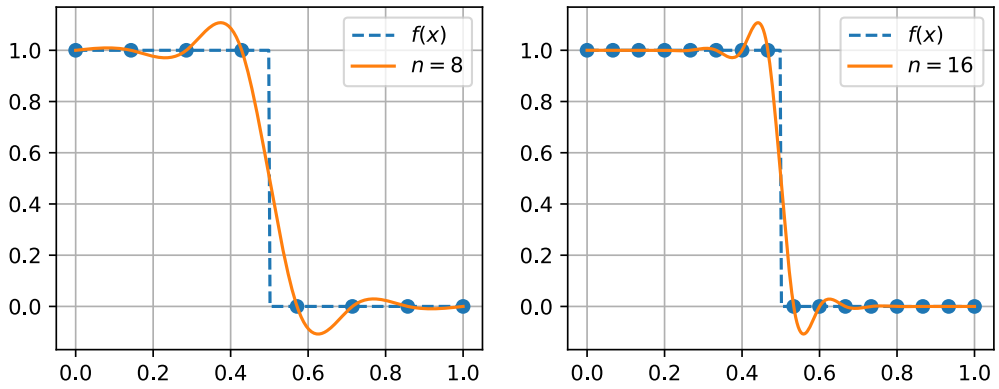


Abbildung 5.4: Spline-Interpolation für eine Sprungfunktion. Das Phänomen des Überschwingers ist mit dem der Fourierinterpolation verwandt, bekannt unter dem Begriff des Gibbs'schen Phänomens.

5.2.2 Datenfit - Smoothing Splines

In vielen Anwendungen ist eine unbekannte Funktion aus einer grossen Anzahl Messungen zu ermitteln. Aufgrund der zu erwartenden Messfehler oder auch wegen des Umfangs der Datenmengen ist Interpolation dann häufig nicht mehr sinnvoll.

Gegeben seien Messungen f_j, $j = 1, \ldots m$, die gewissen Abszissen x_j, $j = 1, \ldots, m$ in einem Intervall $[a, b]$ zugeordnet werden. Die dahintersteckende Funktion f wird mit Hilfe einer B-Spline für die Knoten $\tau = \{\tau_0, \ldots, \tau_{\ell+1}\}$ bzw. der erweiterten Knotenfolge $T = \{t_1, \ldots, t_n\}$ $n := 2k + \ell$ approximiert, wobei das Funktional

$$\sum_{j=1}^{m} (S(x_j) - f_j)^2 = \sum_{j=1}^{m} \left(\sum_{i=1}^{\ell+k} c_i N_{i,k}(x_j) - f_j \right)^2 \xrightarrow{c \in \mathbb{R}^{\ell+k}} \min \tag{5.9}$$

minimiert wird. Hier ist $m \gg k + \ell$ und die x_j müssen natürlich nicht mit den Knoten t_j übereinstimmen. Die Aufgabe (5.9) ist ein lineares Ausgleichsproblem der Form

$$\|A_T \cdot c - f\|_2 \to \min$$

wobei

$$A_T = (N_{j,k}(x_i))_{i=1,j=1}^{m,k+\ell} \in \mathbb{R}^{m\times(k+\ell)}$$
$$f = (f_1, \ldots, f_m)^T \in \mathbb{R}^m.$$

Bei der Wahl der Knoten t_j ist darauf zu achten, dass in jedem Träger der B-Splines $N_{j,k}$ *mindestens* ein x_i fällt. Die Matrix A hat dann vollen Rang, so dass die Methoden aus dem Kapitel 2.2 angewandt werden können.

Beispiel 5.4 (Jupyter-Notebook). Die Abbildung 5.5 zeigt als Beispiel einen Datenfit einer Messung. ⌟

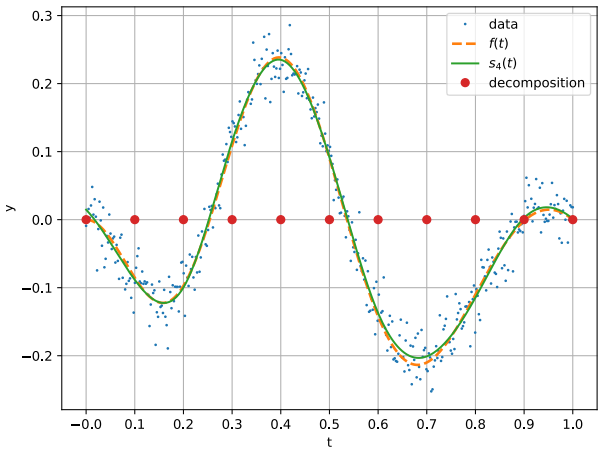


Abbildung 5.5: Spline Approximation.

Stark fehlerbehaftete Datensätze und starke Datenausreisser können zu einem Überfitten mit entsprechenden Oszillationen führen. Das Konzept des *Smoothing-Splines* schaft Abhilfe. Ein Ansatz besteht darin, die Krümmung global zu minimieren:

$$\sum_{j=1}^{m}(S(x_j) - f_j)^2 + \theta\|S''\|_2^2 = \sum_{j=1}^{m}(S(x_j) - f_j)^2 + \theta\int_a^b |S''(x)|^2 dx \xrightarrow{c\in\mathbb{R}^{\ell+k}} \min \qquad (5.10)$$

Mehr Hintergrund ist in [1] zu finden. In der Abbildung 5.6 ist die Anwendung auf das obige Beispiel zusehen, wobei anstelle einer Zerlegung mit 10 Intervalle 100 benutzt wurden. Ohne Glättung (smoothing) schwingt die kubische B-Spline Approximation und folgt dem Rauschen. Mit zusätzlicher Minimierung der globalen Krümmung kann dies gedämpft werden.

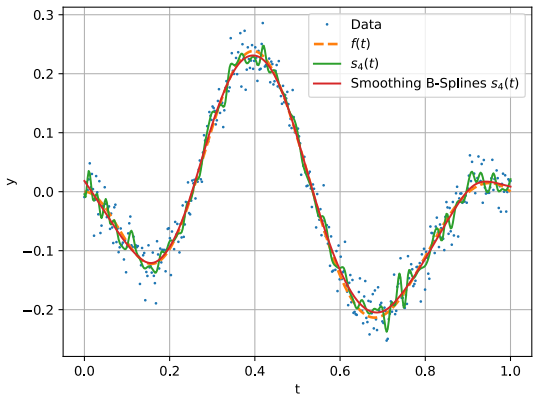


Abbildung 5.6: Smoothing Spline Approximation (vgl. Jupyter-Notebook).

Literaturverzeichnis

[1] Dahmen Wolfgang, Reusken Arnold, *Numerik für Ingenieure und Naturwissenschaftler*, Springer-Lehrbuch, Springer-Verlag Berlin Heidelberg, Deutschland, 2008. ISBN 978-3-540-76492-2, ISBN 978-3-540-76493-9 (eBook), DOI 10.1007/978-3-540-76493-9.

[2] Quarteroni Alfio, Sacco Riccardo, Saleri Fausto, *Numerical Mathematics*, Springer, Springer-Verlag Berlin Heidelberg, Deutschland, 2007. ISBN 978-3-540-34658-6.

[3] Burg Klemens, Haf Herbert, Wille Friedrich, Meister Andreas, *Höhere Mathematik für Ingenieure, Band III: Gewöhnliche Differentialgleichnungen, Distributionen, Integraltransformationen*, Springer Vieweg, Springer Fachmedien Wiesbaden, Deutschland, 2013. ISBN 978-3-8348-1943-7, ISBN 978-3-8348-2334-2 (eBook), DOI 10.1007/978-3-8348-2334-2.

[4] Stingelin Simon, *Arbeitsunterlagen zum Kurs Analysis 1 & 2 für Studierende des Studiengangs Elektrotechnik und Systemtechnik*, ZHAW, Technikumstr. 9, 8400 Winterthur, 2015.

[5] ———, *Arbeitsunterlagen zum Kurs Numerik und Differentialgleichungen 1 & 2 für Studierende des Studiengangs Elektrotechnik*, ZHAW, Technikumstr. 9, 8400 Winterthur, 2015.

[6] Gander Walter, *Algorithms for the QR-Decomposition*, Research Report No. 80-02, ETH Zürich, Seminar für Angewandte Mathematik (1980).

[7] Schwarz Hans Rudolf, Köckler Norbert, *Numerische Mathematik*, Vieweg+Teubner Verlag, Springer Fachmedien Wiesbaden, Deutschland, 2011. ISBN 978-3-8348-1551-4.

[8] Strauss, Walter A., *Partielle Differentialgleichungen : eine Einführung* (Salzmann, Helmut, ed.), Vieweg, Wiesbaden, 1995. aus dem Amerianischen übersetzt.

[9] Hairer Ernst, Nørsett Syvert P., Wanner Gerhard, *Solving Ordinary Differential Equations I, Non-stiff Equations*, Springer-Verlag, Berlin Heidelberg, 1993.

[10] ———, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin Heidelberg, 1996.