Numerische Verfahren

für die Studiengänge
Data Science, Ingenieurinformatik, Materials Science and Engineering,
Materialwissenschaften, Physik und Wirtschaftsingenieurwesen

Vorlesungsmitschrift

Version: 10. Juli 2026

Lisa Beck
Sommersemester 2026

Universität
Augsburg
University

Dieses Skript entstand aus den Mitschriften der Vorlesung “Numerische Verfahren” an der Universität Augsburg und ist für eine gründliche Nachbereitung des Stoffes gedacht. Zu diesem Zweck gibt es auch einige Ergänzungen, die zwar wissenswert oder aufschlussreich, aber für das Verständnis der Vorlesung nicht essentiell sind. Dieses zusätzliche Material dient der Vertiefung für besonders interessierte Student:innen und ist jeweils mit ★ gekennzeichnet.

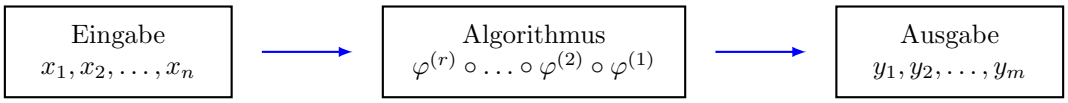
Hinweise zu Fehlern, Verbesserungsvorschläge oder sonstige Kommentare sind jederzeit willkommen und ich bitte darum, mir diese per E-Mail an lisa.beck@math.uni-augsburg.de zukommen zu lassen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Numerische Mathematik beschäftigt sich mit der Entwicklung, Analyse und Implementierung von Algorithmen, mit denen die Lösung mathematischer Problemstellungen approximativ (und möglichst effektiv) berechnet werden kann. Solche *Algorithmen* stellen eine eindeutig festgelegt Abfolge von endlich vielen "elementaren" Operationen dar, mit denen man aus Eingabedaten (etwa Anfangsbedingungen, Materialeigenschaften, Messwerten etc.) eine approximative Lösung des gegebenen Problems bestimmt.



Viele Anwendungen stammen aus der Simulation oder der mathematischen Modellierung komplexer Vorgänge aus Natur-, Ingenieur-, Lebens- und Wirtschaftswissenschaften. Um die Aussagekraft des Ergebnisses einschätzen zu können, muss man die verschiedenen dabei auftretenden Fehler analysieren. Zu unterscheiden sind dabei grundsätzlich:

(1) **Modellfehler** (unabhängig von der numerischen Mathematik)

   (a) *Idealisierungsfehler*: mathematische Modelle sind nur Näherungen, bei denen zudem oft Vereinfachungen wie Linearisierung etc. vorgenommen werden.

   (b) *Datenfehler*: Parameter in den mathematischen Modellen sind fehlerbehaftet, beispielsweise werden die Materialeigenschaften durch Messungen nur näherungsweise bestimmt.

(2) **Numerische Fehler**

   (a) *Diskretisierungsfehler*: bei den numerischen Behandlung werden kontinuierliche Prozesse und Größen durch diskrete ersetzt, beispielsweise werden Integrale über endliche Summen und Ableitungen über Differenzenquotienten ausgedrückt.

   (b) *Abbruch- und Approximationsfehler*: unendliche Rechenvorschriften werden nach endlich vielen Schritten abgebrochen.

   (c) *Rundungsfehler*: durch das Rechnen mit Maschinenzahlen treten immer Fehler auf.

Wir beschäftigen uns in dieser Vorlesung mit der numerischen Behandlung mathematischer Probleme. Vorab wollen wir aber zumindest ein Beispiel für mathematische Modellierung geben, nämlich für Diffusionsprozesse.

Mathematische Modellierung von Diffusion. Wir wollen beschreiben, wie die Änderung einer physikalischen Größe u (wie etwa einer Teilchenkonzentration oder einer Temperaturverteilung) in Abhängigkeit von der Zeitvariable t und der Ortsvariable $x \in \Omega \subset \mathbb{R}^d$ (mit typischerweise $d = 1, 2, 3$ in Anwendungen) aussieht. Das Prinzip der *Massenerhaltung* besagt zunächst, dass die zeitliche Änderung

der Größe u zwischen zwei Zeitpunkten $t_1 < t_2$ in einem beliebigen, glatt berandeten Testvolumen $V \Subset \Omega$ einerseits durch den Fluss durch den Rand ∂V und andererseits durch die Effekte von Quellen und Senken im Inneren von V begründet sind. Bezeichnen wir die *Flussdichte* von u (d.h. die Konzentration, die in einer Zeiteinheit durch eine senkrecht zum Diffusionsstrom stehende Flächeneinheit strömt) mit F und die Dichte von Quellen und Senken mit f, so gilt also

$$\int_V u(x,t_2)\,dx - \int_V u(x,t_1)\,dx = -\int_{t_1}^{t_2}\int_{\partial V} F(x,t)\cdot\nu(x)\,dS^{d-1}(x)\,dt + \int_{t_1}^{t_2}\int_V f(x,t)\,dx\,dt$$

mit dem $(d-1)$-dimensionalen Oberflächenmaß S^{d-1} und dem äußeren Einheitsnormalenfeld ν. Indem wir nun zu Differenzenquotienten in der Zeit übergehen und dann den Satz von Gauß anwenden, folgt

$$\begin{aligned}\frac{d}{dt}\int_V u(x,t)\,dx &= -\int_{\partial V} F(x,t)\cdot\nu(x)\,dS^{d-1}(x) + \int_V f(x,t)\,dx\\ &= -\int_V \operatorname{div} F(x,t)\,dx + \int_V f(x,t)\,dx.\end{aligned}$$

Da dieser Zusammenhang für alle glatt berandeten Teilmengen $V \Subset \Omega$ gilt, können wir durch einen Grenzübergang im Testvolumen auf die partielle Differentialgleichung

$$\partial_t u = -\operatorname{div} F + f \quad \text{in } \Omega\times[0,\infty)$$

schließen. In vielen Situationen ist es nun physikalisch vernünftig, dass die Bewegung der Größe u von Regionen höherer Konzentration zu Regionen niedrigerer Konzentration geht und sich die Flussdichte F dabei proportional zum (negativen) Gradienten Du verhält, also

$$F = -a\nabla u \quad \text{für eine Konstante } a > 0$$

gilt. Im Fall von Diffusion von Teilchen (für die u eine chemische Konzentration repräsentiert) ist dies genau die Aussage des ersten Fick'schen Gesetzes, und im Fall von Wärmeleitung (für die u eine Temperaturverteilung repräsentiert) ist dies die Aussage des Fourierschen Gesetzes. Setzen wir die Annahme $F = -a\nabla u$ nun in die partielle Differentialgleichung oben ein, erhalten wir (mit der Notation $\Delta u = \operatorname{div}\nabla u$ des Laplace-Operators) eine inhomogene Wärmeleitungsgleichung der Form

$$\partial_t u = \operatorname{div}(aDu) + f = a\Delta u + f \qquad \text{in } \Omega\times[0,\infty),$$

mit Anfangsbedingung

$$u(x,0) = u_0(x) \quad \text{in } \Omega.$$

Um Eindeutigkeit der Lösung erwarten zu können, wird typischerweise noch eine Randbedingung vorgeschrieben, beispielsweise eine Dirichlet-Randbedingung

$$u(x,t) = g(x) \quad \text{auf } \partial\Omega\times(0,\infty).$$

Befindet sich die Größe u im Gleichgewicht, gilt also $\partial_t u = 0$, und wird $a = 1$ angenommen, so vereinfacht sich die partielle Differentialgleichung zur *Poisson-Gleichung*

$$-\Delta u = f \quad \text{in } \Omega.$$

Aspekte der numerischen Behandlung von Diffusionsgleichungen. Zur Vereinfachung nehmen wir als Gebiet $\Omega = (0,1)^2$ den offenen zwei-dimensionalen Einheitswürfel an. Nun zerlegen wir das Gebiet und ein Zeitintervall $[0,T]$ äquidistant, d.h. für natürliche Zahlen $n, N \in \mathbb{N}$ setzen wir $h := \frac{1}{N}$ und $\tau = \frac{T}{n}$ (also die *Gitterweite* in Raum und Zeit), definieren

$$\begin{aligned}\Omega_h &:= \big\{x_{j,\ell} := (jh,\ell h)\colon j,\ell\in\{1,\dots,N-1\}\big\},\\ \Gamma_h &:= \big\{x_{j,\ell} := (jh,\ell h)\colon j \text{ oder } \ell\in\{0,N\}\big\},\\ [0,T]_\tau &:= \big\{t_k := k\tau\colon k\in\{0,1,\dots,n\}\big\},\end{aligned}$$

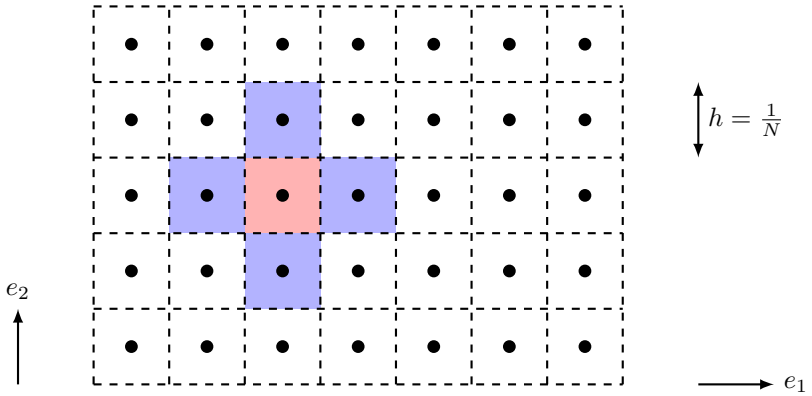
und versuchen nun die kontinuierliche Größe u durch eine Funktion u_h^k auf den Gitterpunkten aus $(\Omega_h \cup \Gamma_h)$ und für jeden Zeitschritt $t_k \in [0,T]_\tau$ zu approximieren. Das einfachstes Vorgehen ist es nun, die auftretenden partiellen Ableitungen durch Differenzenquotienten (mit Abstand der Gitterweiten in Raum und Zeit) zu approximieren:

$$\begin{aligned}
\partial_t u(x,t) &\approx \frac{u(x,t) - u(x,t-\tau)}{\tau} \\
\partial_{x_1} u(x,t) &\approx \frac{u(x+he_1,t) - u(x,t)}{h} \\
\partial_{x_1}^2 u(x,t) &\approx \frac{\partial_{x_1} u(x,t) - \partial_{x_1} u(x-he_1,t)}{h} = \frac{u(x+he_1,t) - 2u(x,t) + u(x-he_1,t)}{h^2} \\
\partial_{x_2}^2 u(x,t) &\approx \frac{u(x+he_2,t) - 2u(x,t) + u(x-he_2,t)}{h^2}.
\end{aligned}$$

Insbesondere können wir so nun einen *diskreten Laplace-Operator* Δ_h einer Funktion h auf Ω_h definieren über

$$\Delta_h h(jh,\ell h) \coloneqq \frac{1}{h^2}\big(-4h(jh,\ell h) + h((j+1)h,\ell h) + h((j-1)h,\ell h) + h(jh,(\ell+1)h) + h(jh,(\ell-1)h)\big).$$

Diese Diskretisierung bezeichnet man auch als 5-*Punkt-Stern*, da neben dem Punkt $(jh,\ell h)$ in der Mitte nur die vier Nachbarpunkte rechts, links, oben und unten vorkommen.



Diskretisieren wir auch noch die Funktion f, die Anfangswerte u_0 sowie die Randwerte g in den Gitterpunkten aus Ω_h bzw. Γ_h über

$$f_h^k(jh,\ell h) \coloneqq f(jh,\ell h,t_k) \quad \text{und} \quad u_h^0(jh,\ell h) \coloneqq u_0(jh,\ell h) \quad \text{für } (jh,\ell h) \in \Omega_h$$

und

$$g_h(jh,\ell h) \coloneqq g(jh,\ell h) \quad \text{für } (jh,\ell h) \in \Gamma_h,$$

so ergibt sich für jeden Zeitschritt t_{k+1} mit $k \in \{1,\dots,n-1\}$ das (lineare) Gleichungssystem

$$\begin{cases} u^{k+1}(jh,\ell h) = u^k(jh,\ell h) + \tau\big(a\Delta_h u^{k+1}(jh,\ell h) + f_h^{k+1}(jh,\ell h)\big) & \text{für } (jh,\ell h) \in \Omega_h \\ u^{k+1}(jh,\ell h) = g_h(jh,\ell h) & \text{für } (jh,\ell h) \in \Gamma_h \end{cases}$$

mit $(n{+}1)^2$ Gleichungen pro Zeitschritt in den $(N{+}1)^2$ Unbekannten $u^{k+1}(jh,\ell h)$ für $j,\ell \in \{0,1,\dots,N\}$. Einige zentrale Fragestellungen im Zusammenhang mit der Diskretisierung partieller Differentialgleichungen sind nun die folgenden:

(1) Existiert immer eine Lösung?

und falls ja, dann:

(2) Konvergiert (in einem geeigneten Sinn) das Verfahren gegen die kontinuierliche Lösung und können wir den Fehler quantifizieren?

(3) Ist die Approximation stabil unter Störungen (wie etwa durch Fehler in den Eingangsdaten oder durch das Rechnen mit Maschinenzahlen)?

Im Rahmen dieser Vorlesung behandeln wir solche and andere Fragen in Zusammenhang mit numerischen Verfahren. Dazu betrachten wir auch wichtige Teilaspekte separat und gehen insbesondere auf die folgenden Themen ein:

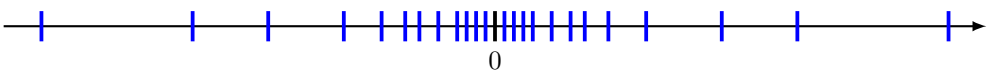
- *lineare Gleichungssysteme*: finde zu einer gegebenen regulären Matrix $A \in \mathbb{R}^{n\times n}$ und einem Vektor $b \in \mathbb{R}^n$ einen Vektor $x \in \mathbb{R}^n$ mit $Ax = b$.
- *Datenklassifizierung über neuronale Netze*: bestimme die Parameter einer neuronalen Netzes so, dass die Klassifikation eines gegebenen Datensatzes möglich gut reproduziert wird.
- *nichtlineare Gleichungssysteme*: finde zu einer gegebenen Funktion $F\colon D \subset \mathbb{R}^n \to \mathbb{R}^n$ einen Vektor $x \in \mathbb{R}^n$ mit $F(x) = 0$ (Nullstellenproblem).
- *Interpolation*: finde zu gegebenen Mess- oder Stützpunkten $(x_0, y_0), \dots, (x_n, y_n)$ eine Funktion in einer Klasse $\mathcal{F}_n$ (z.B. der Klasse aller Polynome vom Grad höchstens n) mit $f(x_j) = y_j$ für alle $j \in \{0, 1, \dots, n\}$.
- *numerische Integration*: approximiere ein Integral $\int_a^b f\,dx$ durch eine gewichtete Summe von Funktionswerten von f (mit einer geschickten Wahl von Gewichten und Stützpunkten), insbesondere in der Situation, in der keine Stammfunktion bekannt ist.
- *gewöhnliche Differentialgleichungen*: bestimme eine Lösung der gewöhnlichen Differentialgleichung $u' = f(t, u)$ mit Anfangsbedingung $u(t_0) = u_0$. Schreiben wir dies äquivalent in der integrierten Form
$$u(t) = u_0 + \int_{t_0}^{t} f(s, u(s))\,ds,$$
so entspricht dies der Integration einer Funktion, die von der Lösung selbst abhängt und demzufolge nur implizit gegeben ist.
- *partielle Differentialgleichungen*: bestimme eine Lösung für eine partielle Differentialgleichung mit geeigneten Anfangs- und Randbedingungen wie bei der oben behandelten Diffusionsgleichung.

Zur Bewertung der Aussagekraft numerischer Lösungen

Wir gehen nun noch etwas genauer darauf ein, welche Probleme grundsätzlich bei der numerischen Lösung mathematischer Probleme auftreten.

Rundungsfehler und Gleitpunktarithmetik. Ausgangsdaten, Zwischen- und Endergebnisse werden in *Maschinenzahlen* $\mathbb{M}$ angegeben. Dies ist eine endliche Zahlenmenge, auf die $\mathbb{R}$ mithilfe einer Rundungsfunktion Rd abgebildet wird (bzw. "overflow", falls sie zu groß vom Betrag ist). Die Verteilung der Maschinenzahlen ist *nicht äquidistant*, sondern (außer für ein Intervall um die Null) so, dass der relative Fehler bei der Rundung immer etwa gleich ist, also

$$\mathrm{Rd}(x) = x(1 + \varepsilon(x)) \quad \text{für ein } \varepsilon(x) \text{ mit } |\varepsilon(x)| \le \mathrm{eps} = \textit{relative Maschinengenauigkeit}.$$



Schema von exakt darstellbaren Maschinenzahlen

Beim Rechnen mit Maschinenzahlen muss man entsprechende Maschinenoperationen, sogenannte *Gleitpunktoperationen*, einführen (etwa $\oplus, \ominus, \otimes, \oslash$ anstelle von $+, -, \times, \div$, mit $x \circledast y = \mathrm{Rd}(x \star y)$ für Maschinenzahlen $x, y \in \mathbb{M}$ und $\star \in \{+, -, \times, \div\}$), da die Menge der Maschinenzahlen unter diesen Operationen nicht abgeschlossen ist. Insbesondere gilt für die Addition und Multiplikation im Allgemeinen weder das Assoziativgesetz noch das Distributivgesetz. Die Reihenfolge der Ausführung einzelner Operationen kann damit essentiell sein. Bei der Subtraktion ähnlich großer Zahlen kann es zu *Auslöschung führender Stellen* und damit zu einem *großen relativen Fehler* kommen.

Beispiel 1.1. Wir runden über eine Funktion Rd_3 auf drei Nachkommastellen. Angewandt auf die beiden Zahlen

$$x := 0,7356 \qquad \mathrm{Rd}_3(x) = 0,736$$
$$y := 0,7344 \qquad \mathrm{Rd}_3(y) = 0,734$$

ergibt sich dann als relative Fehler

$$\left|\frac{\mathrm{Rd}_3(x) - x}{x}\right|, \left|\frac{\mathrm{Rd}_3(y) - y}{y}\right| < 0,0006 = 0,06\%.$$

Bei Bildung der Differenz der exakten bzw. der auf drei Nachkommastellen gerundeten Zahlen erhalten wir

$$x - y = 0,0012 \quad \text{bzw.} \quad \mathrm{Rd}_3(x) - \mathrm{Rd}_3(y) = 0,002$$

mit einem relativen Fehler

$$\left|\frac{\mathrm{Rd}_3(x) - \mathrm{Rd}_3(y) - (x - y)}{x - y}\right| = 0,\overline{6} > 66\%.$$

Rundungsfehler und das Rechnen mit Maschinenzahlen führt also zwangsläufig zu Fehlern, die große Auswirkungen haben und im schlimmsten Fall sogar fatal enden können:

(1) *Börsencrash Vancouver 1982*
An der Vancouver Stock Exchange wurde im Januar 1982 ein Startwert des Aktienindex von 1000 Punkten genommen. Bei jeder Neuberechnung (was täglich tausende von Malen geschah) wurde der Aktienindex auf vier Nachkommastellen genau berechnet, dann die vierte Nachkommastelle abgeschnitten, also konsequent abgerundet, und mit diesem Wert dann weitergerechnet. Diese minimalen Fehler führten dazu, dass der Aktienindex bis Ende November 1983 auf ca. 524,8 Punkt fiel, obwohl der tatsächliche Indexwert ca. 1098,9 betrug. Nachdem Analysten diesen Fehler entdeckt hatten, wurde die korrekte Rundung in die Software eingefügt und der Wert des Aktienindex zum nächsten Börsentag korrigiert (was dann zu einer scheinbaren Verdopplung des Indexwertes führte).

(2) *Patriot-Raketen-Fehler 1991 während des zweiten Golfkriegs*
Am 25. Februar 1991 versagte ein US-Patriot-Raketensystem beim Abfangen einer irakischen Scud-Rakete, die daraufhin im US-Stützpunkt Dhahran in Saudi-Arabien einschlug. Dabei kamen 28 US-Soldaten ums Leben und viele weitere wurden verletzt. Das Problem lag in der Zeitmessung der Patriot-Rakete. Um die Zeit seit dem Neustart des Raketensystems zu messen, wird eine interne Uhr verwendet, die auf Zehntelsekunden basiert, was aber als Gleitkommazahl in dem 24-Bit-System nicht exakt darstellbar ist. Dies führte pro Zehntelsekunde zu einem Rundungsfehler von $9,5 \cdot 10^{-8}$, was sich nach 100 Stunden Dauerbetrieb auf einem Gesamtfehler von 0,34 Sekunden aufsummierte. Da die Geschwindigkeit einer Scud-Rakete ungefähr 6000 Kilometer pro Stunde beträgt (und sie damit in 0,34 Sekunden ca. 570 Meter zurücklegt), führte die fehlerhafte Systemzeituhr im Patriot-Raketensystem dazu, dass die Scud-Rakete zwar erkannt, aber die Zielerfassung durch das Radarsystem fehlschlug und daher auch keine Abwehrrakete abgefeuert wurde.

(3) *Explosion der Ariane 5-Rakete 1996*
Am 4. Juni 1996 explodierte die Ariane 5 ca. 40 Sekunden nach ihrem Start in einer ungefähren Höhe von 4000 Metern, was zu einem Verlust von etwa 370 Mio. US$ (für die Rakete, die vier geladenen Satelliten und die Entwicklungskosten) führte und damit als der teuerste Computerfehler der Geschichte zählt. Die Ursache war eine Fehlfunktion des Bordcomputers 36,7 Sekunden nach dem Start: bei der Umwandlung der horizontalen Geschwindigkeit von einer 64-Bit-Gleitkomma-Zahl in eine vorzeichenbehaftete 16-Bit-Ganzzahl kam es zu einem overflow, wobei dieser Teil der Software noch aus dem Ariane 4 Programm stammte, bei der der entsprechende Zahlenbereich aufgrund geringerer Geschwindigkeit noch nicht relevant war. Der overflow wurde als gültiges Ergebnis und damit als große Abweichung vom geplanten Kurs interpretiert. Als Folge wurde eine Kurskorrektur eingeleitet, die wiederum zu höheren aerodynamischen Kräfte führte, die die Feststoffbooster abreißen ließ und eine automatische Selbstzerstörung einleitete.

Da bei numerischen Berechnungen immer Fehler auftreten können, müssen wir uns also auch damit beschäftigen, wie sich Fehler auf die Lösung eines mathematischen Problems auswirken und wie ein numerischer Algorithmus mit Fehlern umgeht.

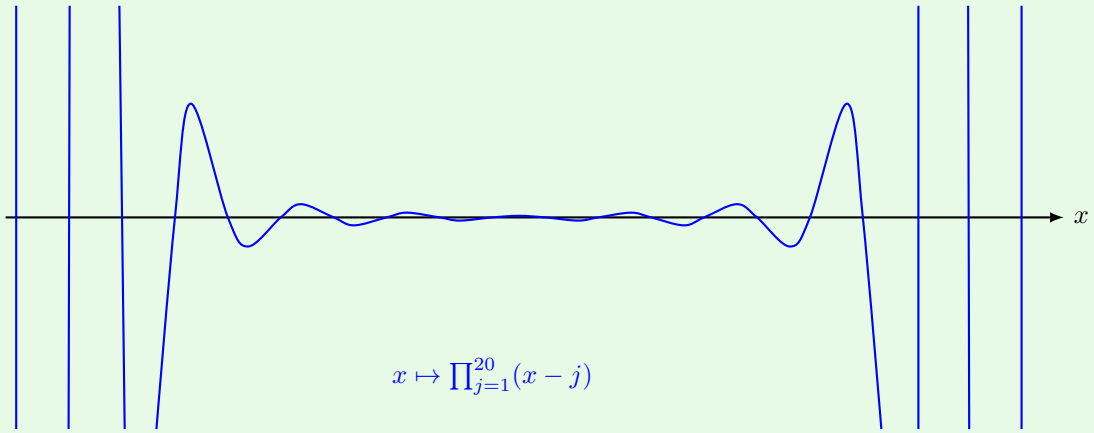
Kondition eines mathematischen Problems. Über den Begriff der Kondition quantifiziert man die Empfindlichkeit bei einem mathematischen Problem, wie stark sich Störungen der Eingabewerte auf das Ergebnis auswirken. Hierbei unterscheidet man zwischen *absoluter* und *relativer Kondition*, also der Verstärkung des absoluten und des relativen Fehlers für "kleine" Störungen (im Sinne einer linearisierten Fehlertheorie).

Beispiele 1.2.

(i) Die Subtraktion zweier annähernd gleich großer Zahlen verstärkt den relativen Fehler stark (siehe Beispiel 1.1) und ist daher schlecht konditioniert.

(ii) Die Bestimmung von Nullstellen einer Funktion kann sehr schlecht konditioniert sein. Dies sieht man gut anhand des *Polynoms von Wilkinson*[1](1963)

$$p_W(x) := \prod_{j=1}^{20}(x-j) = x^{20} - 210x^{19} \pm \ldots + 20! = \sum_{\ell=0}^{20} a_\ell x^\ell.$$



Stört man hier den Koeffizienten a_{19} minimal

$$a_{19} = -210 \quad \rightarrow \quad \tilde{a}_{19} = -210,0000001992$$

[1]Wilkinson selbst beschrieb diese Entdeckung Jahre später folgendermaßen: "Speaking for myself I regard it as the most traumatic experience in my career as a numerical analyst."

(was in einer üblichen Größenordnung beim Runden ist), so sind die Nullstellen des gestörten Polynoms ganz andere: die größte Nullstelle ist anstelle von 20 (wo die Ableitung von p_W den Wert $-20^{19}/(19!)$ hat und damit sehr groß ist) dann $\sim 20,84691$, und es gibt sogar 10 komplexe Nullstellen.

Stabilität numerischer Algorithmen. Über den Begriff der Stabilität beschreibt man die Robustheit des numerischen Verfahrens gegenüber Störungen in den Eingabedaten (z.B. durch Rundungen). Statt eine vorgegebene Abbildung f bei einem x auszuwerten, wertet man stattdessen eine gestörte Abbildung $\tilde{f}$ in einem gestörten Punkt $\tilde{x}$ aus und schaut, ob das Ergebnis – im Vergleich zu dem unvermeidbaren Fehler $|f(x) - f(\tilde{x})|$ durch Rundungen – akzeptabel ist.

Kapitel 2

Lineare Gleichungssysteme

In diesem Kapitel beschäftigen wir uns im Wesentlichen mit der Lösung linearer Gleichungssysteme. Dazu seien eine quadratische Matrix $A \in \mathbb{R}^{n\times n}$ und ein Vektor $b \in \mathbb{R}^n$ gegeben, und wir suchen einen Vektor $x \in \mathbb{R}^n$ mit

$$Ax = b, \tag{2.1}$$

das sich in Komponentenschreibweise darstellen lässt als

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

In dieser Situation gibt es nur die Möglichkeiten, dass es keine einzige, genau eine, oder unendlich viele Lösungen gibt. Wir beschränken uns hier auf den Fall, dass A eine reguläre Matrix ist, also zu A eine Inverse A^{-1} existiert (was äquivalent zur Bedingung $\det(A) \neq 0$ ist), und damit unabhängig von der Wahl der rechten Seite b immer eine eindeutige Lösung existiert, die bei exakter Rechnung als $x = A^{-1}b$ gegeben ist.

Satz 2.1. *Es sei $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix und $b \in \mathbb{R}^n$ ein beliebiger Vektor. Dann existiert ein eindeutiger Vektor $x \in \mathbb{R}^n$ mit $Ax = b$.*

Bemerkung 2.2. *Sucht man die Lösungen $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^n$ des Gleichungssystems für die kanonischen Einheitsvektoren $e^{(1)}, \ldots, e^{(n)}$ als rechter Seite b, so hat man insbesondere die inverse Matrix zu A als $A^{-1} = (x^{(1)}, \ldots, x^{(n)}) \in \mathbb{R}^{n\times n}$ bestimmt.*

In vielen Anwendungen in der Praxis treten lineare Gleichungssysteme mit hoher Dimension n auf (z.B. bei der Diskretisierung von Randwertproblemen partieller Differentialgleichungen), weswegen wir uns nicht nur für die numerische Implementierung konstruktiver Verfahren zur Lösung linearer Gleichungssysteme, sondern auch für den Rechenaufwand (in Abhängigkeit von der Dimension n) interessieren. In diesem Kapitel beschäftigen wir uns mit *direkten Verfahren* und *iterativen Verfahren*. Im Fall von direkten Verfahren wird die Lösung x (bei exakter Rechnung) mittels einer endlichen Anzahl an elementaren Rechenoperationen (Addition, Subtraktion, Multiplikation und Division) bestimmt. Als Beispiele hierfür behandeln wir den *Gauß-Algorithmus* für allgemeine reguläre Matrizen A sowie kurz auch das *Cholesky-Verfahren* für symmetrische, positiv definite Matrizen A. Bei iterativen Verfahren dagegen bestimmt man, ausgehend von einem Startvektor $x^{(0)} \in \mathbb{R}^n$, eine Folge $\{x^{(k)}\}_{k\in\mathbb{N}}$ von Vektoren in $\mathbb{R}^n$, die (bei exakter Rechnung) gegen die Lösung x konvergiert. Hier werden wir das *Verfahren der konjugierten Gradienten* für symmetrische, positiv definite Matrizen A diskutieren.

2.1 Wiederholung: Normen und Skalarprodukte

Allgemeine Normen. Um bei einem mathematischen Problem die Abweichung der Näherungslösung eines numerischen Verfahrens von der exakten Lösung abschätzen zu können, nutzt man Normen.

Definition 2.3 (Norm und normierter Raum)**.** *Es sei V ein reeller (oder komplexer) Vektorraum. Eine Abbildung $\|\cdot\|\colon V \to \mathbb{R}_0^+$ heißt eine* Norm*, falls die folgenden Eigenschaften gelten:*

(N1) Dreiecksungleichung: $\|x+y\| \le \|x\| + \|y\|$ *für alle* $x, y \in V$*,*

(N2) Homogenität: $\|\lambda x\| = |\lambda|\|x\|$ *für alle* $x \in V$ *und* $\lambda \in \mathbb{R}$ *(oder* $\lambda \in \mathbb{C}$*),*

(N3) Positivität: $\|x\| = 0$ *genau dann wenn* $x = 0$*.*

Das Paar $(V, \|\cdot\|)$ eines Vektorraums V und einer Norm $\|\cdot\|$ auf V heißt normierter Raum*.*

Normen auf $\mathbb{R}^n$. Meist arbeiten wir mit den endlich-dimensionalen Vektorräumen $\mathbb{R}^n$ (oder auch $\mathbb{C}^n \simeq \mathbb{R}^{2n}$) und daher geben wir nun einige wichtige Beispiele für Normen auf dem $\mathbb{R}^n$ an.

Beispiel 2.4. Für jedes $p \in [1, \infty]$ ist die Abbildung $\|\cdot\|_p\colon \mathbb{R}^n \to \mathbb{R}_0^+$, definiert durch

$$\|x\|_p := \begin{cases} \left(\sum_{j=1}^n |x_j|^p\right)^{\frac{1}{p}} & \text{für } 1 \le p < \infty \\ \max\{|x_j| \colon j \in \{1,\dots,n\}\} & \text{für } p = \infty \end{cases}$$

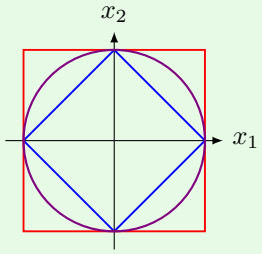
für $x \in \mathbb{R}^n$ (wobei $|\cdot|$ den üblichen Betrag einer reellen Zahl bedeutet), eine Norm auf $\mathbb{R}^n$. Speziell bezeichnet man diese für

(i) $p = 1$ als *Summennorm* oder 1*-Norm*,

(ii) $p = 2$ als *euklidische Norm* oder 2*-Norm*,

(iii) $p = \infty$ als *Maximumsnorm* oder ∞*-Norm*.

Die Einheitssphäre $\{x \in \mathbb{R}^2 \colon \|x\|_p = 1\}$ in der Ebene (bzgl. dieser Norm) ist rechts für diese drei speziellen Fälle gezeichnet.



Bemerkung 2.5. *Man kann zeigen, dass alle Normen auf dem $\mathbb{R}^n$ (und allgemeiner auf einem endlich-dimensionalen Vektorraum)* äquivalent *sind. Dies bedeutet, dass es zu zwei gegebenen Normen $\|\cdot\|$, $\|\cdot\|_*$ auf dem $\mathbb{R}^n$ zwei positive Konstanten C_1 und C_2 gibt mit*

$$C_1\|x\| \le \|x\|_* \le C_2\|x\| \qquad \text{für alle } x \in \mathbb{R}^n.$$

Diese Konstanten müssen jedoch in Abschätzungen berücksichtigt werden, so dass die Normen für die Numerik typischerweise nicht gleichwertig *sind.*

Allgemeine Skalarprodukte. Manchmal hat man auf einem Vektorraum noch mehr Struktur als bei einer Norm, nämlich bei einem Skalarprodukt.

Definition 2.6 (Skalarprodukt und Skalarprodukt-Raum)**.** *Es sei V ein reeller Vektorraum. Eine Abbildung $\langle\cdot,\cdot\rangle\colon V \times V \to \mathbb{R}$ heißt* Skalarprodukt*, falls die folgenden Eigenschaften gelten:*

(S1) Symmetrie: $\langle x, y\rangle = \langle y, x\rangle$ *für alle* $x, y \in V$*,*

(S2) Linearität im ersten Argument: *die Abbildung $x \mapsto \langle x, y\rangle$ ist linear für jedes $y \in V$, d.h. es gilt $\langle \lambda x + \mu\tilde{x}, y\rangle = \lambda\langle x, y\rangle + \mu\langle \tilde{x}, y\rangle$ für alle $x, \tilde{x} \in X$ und $\lambda, \mu \in \mathbb{R}$,*

(S3) positive Definitheit: $\langle x, x\rangle \ge 0$ *für alle $x \in V$, und $\langle x, x\rangle = 0$ gilt genau dann wenn $x = 0$.*

Das Paar $(V, \langle\cdot,\cdot\rangle)$ eines reellen Vektorraums V und eines Skalarprodukts $\langle\cdot,\cdot\rangle$ auf V heißt Skalarprodukt- *oder* Prä-Hilbert-Raum*.*

Bemerkung 2.7. *Über die Symmetrie* (S1) *des Skalarprodukts folgt aus der Linearität* (S2) *im ersten Argument sofort auch die Linearität im zweiten Argument.*

Beispiel 2.8. Auf dem $\mathbb{R}^n$ können wir mithilfe einer beliebigen symmetrischen, positiv definiten Matrix $A \in \mathbb{R}^{n\times n}$ (also mit $A = A^T$ sowie $x^T Ax > 0$ für alle $x \in \mathbb{R}^n \setminus \{0\}$) ein Skalarprodukt über

$$\langle x, y \rangle_A := x^T Ay \qquad \text{für alle } x, y \in \mathbb{R}^n$$

definieren (und man kann sogar zeigen, dass jedes Skalarprodukt auf $\mathbb{R}^n$ tatsächlich so darstellbar ist). Im speziellen Fall, dass A die Einheitsmatrix ist, erhalten wir das *euklidische Skalarprodukt*

$$\langle x, y \rangle_2 := x^T y := \sum_{j=1}^{n} x_j y_j \qquad \text{für alle } x, y \in \mathbb{R}^n. \tag{2.2}$$

Lemma 2.9. *Es sei $(V, \langle \cdot, \cdot \rangle)$ ein Skalarprodukt-Raum. Dann gelten die folgenden Aussagen:*

(i) *Es gilt die* Cauchy–Schwarz Ungleichung $|\langle x, y \rangle| \le \sqrt{\langle x, x \rangle}\sqrt{\langle y, y \rangle}$ *für alle $x, y, \in V$.*

(ii) *Die Vorschrift $\|x\| := \sqrt{\langle x, x \rangle}$ für $x \in V$* induziert eine Norm $\|\cdot\| \colon V \to \mathbb{R}_0^+$.

Bemerkungen 2.10.

(1) *Für die induzierte Norm gilt die Identität*

$$\|x\| = \sup\{\langle x, y \rangle \colon y \in V \text{ mit } \|y\| = 1\} \qquad \text{für alle } x \in V$$

(*die Ungleichung $\ge$ folgt dabei sofort aus der Cauchy–Schwarz Ungleichung, wohingegen die Ungleichung $\le$ für $x \neq 0$ mit der Wahl $y := x/\|x\|$ gilt und für $x = 0$ trivial ist*).

(2) *Die in Beispiel* 2.4 *eingeführte p-Norm $\|\cdot\|_p$ ist nur im Fall $p = 2$ von einem Skalarprodukt induziert, und zwar vom euklidischen Skalarprodukt* (2.2).

Matrixnormen. Wir betrachten nun den Vektorraum $\mathbb{R}^{m\times n}$ aller $(m \times n)$-Matrizen, mit m Zeilen und n Spalten. Da man $\mathbb{R}^{m\times n}$ mit dem Vektorraum $\mathbb{R}^{mn}$ aller Vektoren der Länge mn identifiziert kann, haben wir so bereits erste Beispiele für (Matrix-)Normen auf $\mathbb{R}^{m\times n}$.

Beispiele 2.11. Klassische Beispiele von Matrixnormen auf $\mathbb{R}^{m\times n}$, bei denen jede Matrix $A \in \mathbb{R}^{m\times n}$ (mit Eintrag a_{ij} in der i-ten Zeile und j-ten Spalte) als ein Vektor aufgefasst wird, sind:

(i) die *Frobenius-Norm*, definiert über

$$\|A\|_F := \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\right)^{\frac{1}{2}} = \big(\operatorname{Spur}(A^T A)\big)^{\frac{1}{2}},$$

(ii) die *Maximumsnorm*, definiert über

$$\|A\|_{\max} := \max\{|a_{ij}| \colon i \in \{1, \dots, m\},\, j \in \{1, \dots, n\}\}.$$

Da es sich bei jeder Matrix $A \in \mathbb{R}^{m\times n}$ auch um eine lineare Abbildung von $\mathbb{R}^n$ nach $\mathbb{R}^m$ handelt, sind wir speziell an Matrixnormen interessiert, die diese Struktur berücksichtigen. Dies ist etwa bei der *Operatornorm* der Fall. Hier gilt:

Satz 2.12 (über die von einer Vektornorm induzierte Matrixnorm)**.** *Es sei $\|\cdot\|_{\mathbb{R}^n}$ eine Vektornorm auf $\mathbb{R}^n$ und $\|\cdot\|_{\mathbb{R}^m}$ eine Vektornorm auf $\mathbb{R}^m$. Für die Abbildung $\|\cdot\| \colon \mathbb{R}^{m\times n} \to \mathbb{R}_0^+$, definiert durch*

$$\|A\| := \max\left\{\frac{\|Ax\|_{\mathbb{R}^m}}{\|x\|_{\mathbb{R}^n}} \colon x \in \mathbb{R}^n \setminus \{0\}\right\} = \max\{\|Ax\|_{\mathbb{R}^m} \colon x \in \mathbb{R}^n, \|x\|_{\mathbb{R}^n} = 1\} \quad \text{für } A \in \mathbb{R}^{m\times n},$$

gelten die folgenden Eigenschaften:

(i) $\|\cdot\|$ *ist eine* Norm *auf* $\mathbb{R}^{m\times n}$*,*

(ii) $\|\cdot\|$ *ist* verträglich *mit den Vektornormen* $\|\cdot\|_{\mathbb{R}^n}$ *und* $\|\cdot\|_{\mathbb{R}^m}$*, d.h. es gilt* $\|Ax\|_{\mathbb{R}^m} \leq \|A\|\|x\|_{\mathbb{R}^n}$ *für alle* $A \in \mathbb{R}^{m\times n}$ *und* $x \in \mathbb{R}^n$*.*

(iii) $\|\cdot\|$ *ist im Fall* $m = n$ submultiplikativ*, d.h. es gilt* $\|AB\| \leq \|A\|\|B\|$ *für alle* $A, B \in \mathbb{R}^{n\times n}$*.*

★ *Beweis.* Als Vorbemerkung stellen wir fest, dass die Suprema über beide Mengen in der Definition der induzierten Matrixnorm übereinstimmen, da die Matrixmultiplikation $x \mapsto Ax$ sowie die Vektornormen auf $\mathbb{R}^m$ bzw. $\mathbb{R}^n$ linear sind. Weil die Vektornorm auf $\mathbb{R}^m$ außerdem stetig und die Einheitssphäre $\{x \in \mathbb{R}^n \colon \|x\|_{\mathbb{R}^n} = 1\}$ im $\mathbb{R}^n$ abgeschlossen und beschränkt, also kompakt ist, wird das Supremum angenommen. Folglich ist die Abbildung $\|\cdot\|$ auf $\mathbb{R}^{m\times n}$ wohldefiniert.

(i) Die Normeigenschaften übertragen sich sofort von der Vektornorm auf $\mathbb{R}^m$.

(ii) Verträglichkeit der induzierten Norm $\|\cdot\|$ mit den Vektornormen auf $\mathbb{R}^m$ bzw. $\mathbb{R}^n$ folgt mit der folgenden Rechnung

$$\|Ax\|_{\mathbb{R}^m} = \frac{\|Ax\|_{\mathbb{R}^m}}{\|x\|_{\mathbb{R}^n}}\|x\|_{\mathbb{R}^n} \leq \max\left\{\frac{\|Ax\|_{\mathbb{R}^m}}{\|x\|_{\mathbb{R}^n}} \colon x \in \mathbb{R}^n \setminus \{0\}\right\}\|x\|_{\mathbb{R}^n} = \|A\|\|x\|_{\mathbb{R}^n}$$

für alle $A \in \mathbb{R}^{m\times n}$ und $x \in \mathbb{R}^n \setminus \{0\}$ (und ist für $x = 0$ offensichtlich).

(iii) Für alle $A, B \in \mathbb{R}^{n\times n}$ gilt wegen der Verträglichkeit der induzierten Norm (angewendet auf den Vektor $Bx \in \mathbb{R}^n$) in (ii) auch

$$\begin{aligned}\|AB\| &= \max\left\{\|ABx\|_{\mathbb{R}^n} \colon x \in \mathbb{R}^n, \|x\|_{\mathbb{R}^n} = 1\right\}\\ &\leq \max\left\{\|A\|\|Bx\|_{\mathbb{R}^n} \colon x \in \mathbb{R}^n, \|x\|_{\mathbb{R}^n} = 1\right\} = \|A\|\|B\|,\end{aligned}$$

und somit ist auch die Submultiplikativität bewiesen. □

Beispiele 2.13. Für die in Beispiel 2.4 vorgestellten Vektornormen (jeweils sowohl im Definitionsraum $\mathbb{R}^n$ als auch im Zielraum $\mathbb{R}^m$ verwendet) erhalten wir die folgenden induzierten Matrixnormen (mit $A \in \mathbb{R}^{m\times n}$):

(i) die *Summennorm* $\|\cdot\|_1$ induziert die *Spaltensummennorm*

$$\|A\|_1 = \max\left\{\sum_{i=1}^{m} |a_{ij}| \colon j \in \{1, \dots, n\}\right\},$$

(ii) die *euklidische Norm* $\|\cdot\|_2$ induziert die *Spektralnorm*

$$\|A\|_2 = \left(\lambda_{\max}(A^T A)\right)^{\frac{1}{2}},$$

wobei $\lambda_{\max}(A^T A)$ der größte Eigenwert der (symmetrischen, positiv semi-definiten) Matrix $A^T A$ ist,

(iii) die *Maximumsnorm* $\|\cdot\|_\infty$ induziert die *Zeilensummennorm*

$$\|A\|_\infty = \max\left\{\sum_{j=1}^{n} |a_{ij}| \colon i \in \{1, \dots, m\}\right\}.$$

Wir verwenden in Zukunft hauptsächlich diese induzierten Normen und notieren daher den Index wie in der zugrunde liegenden Vektornorm.

2.2 Konditionsanalyse

Da wir, wie wir gesehen haben, mit dem Computer nicht exakt rechnen können, überlegen wir uns zunächst, wie sich für das Lösen linearer Gleichungssysteme Störungen in den Eingabewerten auf das Ergebnis auswirken (bei sonst exakter Rechnung und damit auch unabhängig von der Wahl des verwendeten Algorithmus). Wir beschränken uns dabei auf Störungen der rechten Seite b für eine feste Matrix A und überlegen uns zunächst anhand eines einfachen Beispiels, dass selbst kleine Störungen zu einem großen Fehler im Ergebnis führen können.

Beispiel 2.14. Für $\varepsilon \neq 0$ betrachten wir die Matrix $A_\varepsilon \in \mathbb{R}^{2\times 2}$ definiert durch

$$A_\varepsilon = \begin{pmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{pmatrix},$$

die wegen $\det(A_\varepsilon) = \varepsilon \neq 0$ regulär ist. Als rechte Seite wählen wir außerdem die Vektoren $b_\varepsilon, \tilde{b}_\varepsilon \in \mathbb{R}^2$ definiert durch

$$b_\varepsilon = \begin{pmatrix} 4 \\ 4+\varepsilon \end{pmatrix} \qquad \text{und} \qquad \tilde{b}_\varepsilon = \begin{pmatrix} 4-\varepsilon \\ 4+2\varepsilon \end{pmatrix},$$

die sich wegen $\|b_\varepsilon - \tilde{b}_\varepsilon\|_\infty = |\varepsilon|$ für kleine $\varepsilon \neq 0$ im Supremumsabstand nur minimal unterscheiden. Als Lösungen $x, \tilde{x} \in \mathbb{R}^2$ für die beiden linearen Gleichungssysteme $A_\varepsilon x = b_\varepsilon$ und $A_\varepsilon \tilde{x} = \tilde{b}_\varepsilon$ erhalten wir

$$x = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \qquad \text{und} \qquad \tilde{x} = \begin{pmatrix} 1-\varepsilon \\ 3 \end{pmatrix},$$

mit $\|x - \tilde{x}\|_\infty = \max\{2, |2+\varepsilon|\}$. Relativ gesehen zur Störung im Eingabewert ist die Störung im Ergebnis für kleine Werte $\varepsilon \neq 0$ also ganz enorm.

Dass das Ergebnis so empfindlich auf (manche) Störungen reagiert, ist tatsächlich eine Eigenschaft, die man aus der Matrix ablesen kann. Hierzu hilft uns das folgende Lemma:

Lemma 2.15. *Es sei $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix. Sind $x, \tilde{x} \in \mathbb{R}^n$ Lösungen der linearen Gleichungssysteme $Ax = b$ und $A\tilde{x} = \tilde{b}$ für Vektoren $b, \tilde{b} \in \mathbb{R}^n$ mit $b \neq 0$, so gilt für jedes $p \in [1, \infty]$*

$$\frac{\|\tilde{x} - x\|_p}{\|x\|_p} \leq \|A\|_p \|A^{-1}\|_p \frac{\|\tilde{b} - b\|_p}{\|b\|_p}$$

mit der von der Vektornorm $\|\cdot\|_p$ auf $\mathbb{R}^n$ induzierte Matrixnorm $\|\cdot\|_p$ auf $\mathbb{R}^{n\times n}$.

Beweis. Da die Matrix A nach Voraussetzung invertierbar ist, haben wir als Lösungen der linearen Gleichungssysteme die Vektoren $x = A^{-1}b$ sowie $\tilde{x} = A^{-1}\tilde{b}$. Da die Matrixnorm $\|\cdot\|_p$ auf $\mathbb{R}^{n\times n}$ als von der Vektornorm $\|\cdot\|_p$ auf $\mathbb{R}^n$ induzierte Norm mit dieser Vektornorm wiederum verträglich ist (siehe Satz 2.12), gilt außerdem

$$\|Bv\|_p \leq \|B\|_p \|v\|_p \qquad \text{für alle } B \in \mathbb{R}^{n\times n} \text{ und } v \in \mathbb{R}^n,$$

so dass wir die Ungleichungen

$$\|\tilde{x} - x\|_p = \|A^{-1}\tilde{b} - A^{-1}b\|_p = \|A^{-1}(\tilde{b} - b)\|_p \leq \|A^{-1}\|_p \|\tilde{b} - b\|_p$$

und

$$\|b\|_p = \|Ax\|_p \leq \|A\|_p \|x\|_p$$

folgern. Zusammengesetzt ergibt sich damit gerade die Behauptung des Lemmas. □

Bemerkungen 2.16.

(1) *Die Größe* $\|\tilde{b} - b\|_p / \|b\|_p$ *auf der rechten Seite der Ungleichung stellt gerade den relativen Eingabefehler dar und die Größe* $\|\tilde{x} - x\|_p / \|x\|_p$ *auf der linken Seite der Ungleichung den relativen Fehler der Lösung. Die Zahl* $\|A\|_p \|A^{-1}\|_p$ *beschreibt daher den* maximalen Verstärkungsfaktor des relativen Fehlers*.*

(2) *Im Beispiel* 2.14 *oben haben wir*

$$A_\varepsilon^{-1} = \varepsilon^{-1} \begin{pmatrix} 1+\varepsilon & -1 \\ -1 & 1 \end{pmatrix},$$

und damit erhalten wir für $\varepsilon > 0$ *als maximalem Verstärkungsfaktor (mit* $p = \infty$ *und der Zeilensummennorm aus Beispiel* 2.13 (iii)) *gerade*

$$\|A_\varepsilon\|_\infty \|A_\varepsilon^{-1}\|_\infty = (2+\varepsilon)^2 \varepsilon^{-1},$$

was die beobachtete hohe Sensitivität der Lösung bezüglich Störungen in der rechten Seite b widerspiegelt.

Die Interpretation der Größe $\|A\|_p \|A^{-1}\|_p$ als maximalem Verstärkungsfaktor des relativen Fehlers motiviert nun zu folgender Definition:

Definition 2.17 (Konditionszahl einer Matrix)**.** *Ist* $A \in \mathbb{R}^{n\times n}$ *eine reguläre Matrix, so definieren wir für* $p \in [1, \infty]$ *die* Konditionszahl von A bzgl. der Norm $\|\cdot\|_p$ *als*

$$\kappa_p(A) := \|A\|_p \|A^{-1}\|_p.$$

Bemerkung 2.18. *Die Konditionszahl einer Matrix in Anwendungen auszurechnen ist oft nicht möglich oder sehr umständlich, da in der Definition die Inverse* A^{-1} *vorkommt, die man im Allgemeinen nicht explizit bestimmen will oder kann. Abschätzungen der Konditionszahl sind typischerweise ausreichend, basierend etwa auf einer (nicht schwer zu zeigenden) Interpretation der Konditionszahl als Verhältnis zwischen kleinster und größter Streckung von Einheitsnormalenvektoren unter Anwendung auf die Matrix, also*

$$\kappa_p(A) = \frac{\max\{\|Av\|_p \colon v \in \mathbb{R}^n \text{ mit } \|v\|_p = 1\}}{\min\{\|Av\|_p \colon v \in \mathbb{R}^n \text{ mit } \|v\|_p = 1\}} \geq 1.$$

2.3 Gauß-Algorithmus

Die Gaußsche Eliminationsmethode ist ein konstruktives Verfahren zur Lösung linearer Gleichungssysteme der Form (2.1). In diesem Abschnitt werden wir das grundsätzliche Vorgehen wiederholen und uns dabei auf den Blickwinkel fokussieren, dass es sich um ein zweistufiges Verfahren handelt, in dem man das gegebene lineare Gleichungssystem in einem ersten Schritt zunächst so transformiert, dass man ein gestaffeltes lineares Gleichungssystem erhält, und dieses dann im zweiten Schritt löst. Daher werden wir uns zunächst mit diesem Spezialfall eines gestaffelten Gleichungssystems beschäftigen, bevor wir den Gauß-Algorithmus dann in voller Allgemeinheit für reguläre Matrizen diskutieren.

2.3.1 Gestaffelte Gleichungssysteme

Wir schauen uns nun das Lösungsverfahren für lineare Gleichungssysteme $Ax = b$ an, falls A eine untere oder obere Dreiecksmatrix ist.

Definition 2.19 (Dreiecksmatrix)**.** *Eine Matrix* $L \in \mathbb{R}^{n\times n}$ *heißt eine* (linke) untere Dreiecksmatrix*, falls* $\ell_{ij} = 0$ *für* $i < j$ *gilt, und eine Matrix* $R \in \mathbb{R}^{n\times n}$ *heißt eine* (rechte) obere Dreiecksmatrix*, falls* $r_{ij} = 0$ *für* $i > j$ *gilt. In diesen Fällen verschwinden bei* L *bzw.* R *alle Matrixeinträge oberhalb*

bzw. unterhalb der Diagonale und L und R sind damit von der Form

$$L = \begin{pmatrix} \ell_{11} & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \cdots & \ell_{n,n-1} & \ell_{nn} \end{pmatrix} \quad \text{bzw.} \quad R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ 0 & \cdots & 0 & r_{nn} \end{pmatrix}.$$

Sind alle Diagonaleinträge gleich 1, so nennen wir eine untere oder obere Dreiecksmatrix normiert.

Bemerkungen 2.20.

(1) *Die Transponierte L^T einer unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ ist eine obere Dreiecksmatrix (und umgekehrt ist die Transponierte einer oberen Dreiecksmatrix eine untere Dreiecksmatrix).*

(2) *Für eine Dreiecksmatrix $A \in \mathbb{R}^{n\times n}$ gilt $\det(A) = \prod_{i=1}^{n} a_{ii}$. Damit ist A genau dann regulär, wenn $a_{ii} \neq 0$ für jedes $i \in \{1, \ldots, n\}$ gilt.*

(3) *Das Produkt aus zwei unteren bzw. oberen (normierten) Dreiecksmatrizen ist wieder eine untere bzw. obere (normierte) Dreiecksmatrix.*

Lösungsverfahren für untere Dreiecksmatrizen. Ist $L \in \mathbb{R}^{n\times n}$ eine untere Dreiecksmatrix, so liegt für $Lx = b$ ein gestaffeltes Gleichungssystem vor

$$\begin{array}{lllllllll} \ell_{11}x_1 & & & & & & & = & b_1 \\ \ell_{21}x_1 & + & \ell_{22}x_2 & & & & & = & b_2 \\ \vdots & & \vdots & & \ddots & & & & \vdots \\ \vdots & & \vdots & & & \ddots & & & \vdots \\ \ell_{n1}x_1 & + & \ldots & + & \ell_{n,n-1}x_{n-1} & + & \ell_{nn}x_n & = & b_n. \end{array}$$

Ist L regulär (und gilt damit $\ell_{ii} \neq 0$ für jedes $i \in \{1, \ldots, n\}$ nach Bemerkung 2.20 (2)), dann können wir dieses durch *Vorwärtssubstitution* lösen, d.h. durch das sukzessive Lösen der einzelnen Gleichungen (von oben nach unten) unter Ausnutzung der vorherigen:

$$\begin{aligned} x_1 &= b_1/\ell_{11} \\ x_2 &= (b_2 - \ell_{21}x_1)/\ell_{22} \\ &\vdots \\ x_n &= (b_n - \ell_{n1}x_1 - \ldots - \ell_{n,n-1}x_{n-1})/\ell_{nn}. \end{aligned}$$

Algorithmus 2.21: Vorwärtssubstitution

Eingabe: Reguläre untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$, rechte Seite $b \in \mathbb{R}^n$

```
for i = 1 : n do                          ▷ Schleife über die Zeilen von oben nach unten
    x_i = b_i
    for j = 1 : i − 1 do
        x_i := x_i − ℓ_ij x_j
    end for
    x_i = x_i/ℓ_ii
end for
```

Ausgabe: Lösung x von $Lx = b$

Lösungsverfahren für obere Dreiecksmatrizen. Bei einer regulären oberen Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$ geht man vollkommen analog vor. Hier liegt mit $Rx = b$ ebenfalls ein gestaffeltes Gleichungssystem vor

$$\begin{array}{ccccccccc} r_{11}x_1 & + & r_{12}x_2 & + & \dots & + & r_{1n}x_n & = & b_1 \\ & \ddots & & & \vdots & & \vdots & & \vdots \\ & & \ddots & & \vdots & & \vdots & & \vdots \\ & & & & r_{n-1,n-1}x_{n-1} & + & r_{n-1,n}x_n & = & b_{n-1} \\ & & & & & & r_{nn}x_n & = & b_n, \end{array}$$

das wir nun durch *Rückwärtssubstitution* (von unten nach oben) lösen können:

$$\begin{aligned} x_n &= b_n/r_{nn} \\ x_{n-1} &= (b_{n-1} - r_{n-1,n}x_n)/r_{n-1,n-1} \\ &\vdots \\ x_1 &= (b_1 - r_{12}x_2 - \dots - r_{1n}x_n)/r_{11}. \end{aligned}$$

Algorithmus 2.22: Rückwärtssubstitution

Eingabe: Reguläre obere Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$, rechte Seite $b \in \mathbb{R}^n$

```
1: for i = n : 1 do                          ▷ Schleife über die Zeilen von unten nach oben
2:     x_i = b_i
3:     for j = i + 1 : n do
4:         x_i := x_i − r_ij x_j
5:     end for
6:     x_i = x_i / r_ii
7: end for
```

Ausgabe: Lösung x von $Rx = b$

Komplexität. Zur Berechnung der Lösung sind bei Vorwärts- und Rückwärtssubstitution jeweils

$$\sum_{i=1}^{n}(2i-1) = n(n+1) - n = n^2$$

Gleitkommaoperationen der Addition, Subtraktion, Multiplikation und Division (auch abgekürzt als FLOP für die englische Bezeichnung floating-point operation) notwendig. Damit ist die Komplexität dieser Verfahren von der Ordnung $O(n^2)$. Dies ist optimal, da wir n Unbekannte und bis zu $(n-1)n/2$ nicht-verschwindende Einträge in der Matrix L bzw. R haben.

2.3.2 Gleichungssysteme mit LR-Zerlegung

Als nächstes diskutieren wir das Lösungsverfahren für lineare Gleichungssysteme $Ax = b$, falls A regulär ist und als Produkt aus einer unteren und einer oberen Dreiecksmatrix gegeben ist.

Definition 2.23 (LR-Zerlegung)**.** *Wir sagen, dass eine Matrix $A \in \mathbb{R}^{n\times n}$ eine LR-*Zerlegung *hat, falls eine normierte untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ sowie eine obere Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$ existieren, so dass*

$$A = LR \qquad \textit{gilt.}$$

Bemerkung 2.24 (Existenz und Eindeutigkeit der LR-Zerlegung)**.** *Für eine reguläre Matrix $A \in \mathbb{R}^{n\times n}$ gelten die folgenden Aussagen:*

(1) *Die LR-Zerlegung von A ist* eindeutig (*falls existent*).

(2) *Die LR-Zerlegung* existiert *genau dann, wenn alle Hauptminoren*

$$M_m(A) = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mm} \end{pmatrix},$$

die durch Einschränkung der Matrix A auf die ersten $m \in \{1, \dots, n\}$ Zeilen und Spalten entstehen, regulär sind, also $\det(M_m(A)) \neq 0$ *erfüllen. In diesem Fall lässt sich die LR-Zerlegung über den Gauß-Algorithmus* (*siehe nächstes Kapitel* 2.3.3) *bestimmen.*

Beispiele 2.25. Für (reguläre) Matrizen kleiner Dimension kann man eine solche Zerlegung (falls existent) sehr einfach bestimmen, etwa gilt:

(i) Die Matrix

$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} \quad \text{hat als } LR\text{-Zerlegung} \quad \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 0 & -1 \end{pmatrix}.$$

(ii) Die Matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{besitzt keine } LR\text{-Zerlegung}$$

(und hier gilt $\det(M_1(A)) = 0$).

Lösungsverfahren für Matrizen mit LR-Zerlegung. Der Vorteil einer Zerlegung $A = LR$ als Matrixprodukt aus einer unteren mit einer oberen Dreiecksmatrix ist es, dass man die Lösung des linearen Gleichungssystems $Ax = LRx = b$ in *zwei* Probleme zerlegen kann, die man dann mit dem oben beschriebenen Verfahren für Dreiecksmatrizen lösen kann:

- Bestimme die Lösung $y \in \mathbb{R}^n$ des linearen Gleichungssystems $Ly = b$ durch Vorwärtssubstitution,
- und bestimme danach die Lösung $x \in \mathbb{R}^n$ des linearen Gleichungssystems $Rx = y$ durch Rückwärtssubstitution.

Zu beachten ist hierbei, dass mit A auch L und R reguläre Matrizen sind, da wegen der Produkteigenschaft der Determinante $\det(A) = \det(L)\det(R)$ gilt. Mit x haben wir dann auch die Lösung des ursprünglichen Problems gefunden:

$$Ax = LRx = Ly = b.$$

Komplexität. Wir müssen eine Vorwärts- und eine Rückwärtssubstitution durchführen, so dass die Komplexität von der Ordnung $O(n^2)$ ist.

2.3.3 Der Gauß-Algorithmus

Das *Gaußsche Eliminationsverfahren* ist ein zweistufiges konstruktives Verfahren zum Lösen eines linearen Gleichungssystems der Form $Ax = b$ mit regulärer Matrix $A \in \mathbb{R}^{n \times n}$ und einem beliebigen Vektor $b \in \mathbb{R}^n$. Durch elementare Umformungen, nämlich

- Addition eines Vielfachen einer Zeile zu einer anderen Zeile oder
- Vertauschen zweier Zeilen,

bringt man das ursprüngliche Gleichungssystem dabei in der ersten Stufe sukzessive (Zeile für Zeile) auf ein äquivalentes lineares Gleichungssystem $A'x = b'$ in Zeilenstufenform, mit einem transformierten Vektor $b' \in \mathbb{R}^n$ sowie einer transformierten Matrix $A' \in \mathbb{R}^{n \times n}$, die als obere Dreiecksmatrix vorliegt. Dieses Gleichungssystem kann man nun in der zweiten Stufe mit Rückwärtssubstitution lösen. Das Standardverfahren ist dabei wie folgt:

Verfahren 2.26 (Gaußsches Eliminationsverfahren)**.**

(1) *Sukzessive Vorwärtselimination*: Im Schritt $k \in \{1, 2, \dots, n-1\}$ startet man von einem linearen Gleichungssystem $A^{(k)}x = b^{(k)}$ mit regulärer Matrix $A^{(k)} \in \mathbb{R}^{n\times n}$ mit $a_{ij}^{(k)} = 0$ für $j \in \{1, \dots, k\}$ und $i \in \{j+1, \dots, n\}$ (so dass die ersten k Spalten keine nicht-verschwindende Einträge unterhalb der Diagonalen haben) und einem Vektor $b^{(k)} \in \mathbb{R}^n$ (mit $A^{(1)} := A$ und $b^{(1)} := b$) und geht dann zu einem linearen Gleichungssystem $A^{(k+1)}x = b^{(k+1)}$ über, bei dem $A^{(k+1)} \in \mathbb{R}^{n\times n}$ eine reguläre Matrix mit $a_{ij}^{(k+1)} = 0$ für $j \in \{1, \dots, k+1\}$ und $i \in \{j+1, \dots, n\}$ ist und $b^{(k+1)} \in \mathbb{R}^n$ ein Vektor. Dies erreicht man folgendermaßen:

(a) Im Fall $a_{kk}^{(k)} \neq 0$ eliminiert man die Unbekannte x_k aus den Gleichungen $k+1, \dots, n$ mittels der k-ten Gleichung und definiert dazu als Einträge der neuen Matrix $A^{(k+1)}$ und der neuen rechten Seite $b^{(k+1)}$

$$a_{ij}^{(k+1)} := \begin{cases} a_{ij}^{(k)} & \text{für } i \leq k \text{ und } j \in \{1, \dots, n\} \\ a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} & \text{für } i > k \text{ und } j \in \{1, \dots, n\} \end{cases}$$

$$b_i^{(k+1)} := \begin{cases} b_i^{(k)} & \text{für } i \leq k \\ b_i^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} & \text{für } i > k. \end{cases}$$

Die Umformung des ursprünglichen linearen Gleichungssystems $A^{(k)}x = b^{(k)}$ zum neuen Gleichungssystem $A^{(k+1)}x = b^{(k+1)}$ kann man mithilfe der um die rechte Seite $b^{(k)}$ erweiterten Koeffizientenmatrix über

$$\left(\begin{array}{ccccccc|c} a_{11}^{(k)} & & \cdots & & \cdots & & a_{1n}^{(k)} & b_1^{(k)} \\ 0 & a_{22}^{(k)} & & & & & \vdots & \vdots \\ \vdots & & \ddots & & & & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & & a_{kn}^{(k)} & b_k^{(k)} \\ \vdots & \ddots & \vdots & \vdots & & & \vdots & \vdots \\ \vdots & & 0 & a_{ik}^{(k)} & \cdots & & a_{in}^{(k)} & b_i^{(k)} \\ \vdots & & \vdots & \vdots & & & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & & a_{nn}^{(k)} & b_n^{(k)} \end{array}\right) \begin{array}{l} \\ \\ \\ \\ \\ \leftarrow + \cdot\left(-\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\right) \\ \\ \leftarrow + \cdot\left(-\frac{a_{nk}^{(k)}}{a_{kk}^{(k)}}\right) \end{array}$$

darstellen.

(b) Im Fall $a_{kk}^{(k)} = 0$ vertauscht man zunächst die k-te Zeile mit der m-ten Zeile für ein $m \in \{k+1, \dots, n\}$ mit $a_{mk}^{(k)} \neq 0$ (dies ist immer möglich, sonst wäre die Matrix A singulär). Diese Zeilenvertauschung wird mithilfe der erweiterten Koeffizientenmatrix über

$$\left(\begin{array}{ccccccc|c} a_{11}^{(k)} & & \cdots & & \cdots & & a_{1n}^{(k)} & b_1^{(k)} \\ 0 & a_{22}^{(k)} & & & & & \vdots & \vdots \\ \vdots & & \ddots & & & & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk}^{(k)} = 0 & \cdots & & a_{kn}^{(k)} & b_k^{(k)} \\ \vdots & \ddots & \vdots & \vdots & & & \vdots & \vdots \\ \vdots & & 0 & a_{mk}^{(k)} \neq 0 & \cdots & & a_{mn}^{(k)} & b_m^{(k)} \\ \vdots & & \vdots & \vdots & & & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & & a_{nn}^{(k)} & b_n^{(k)} \end{array}\right) \begin{array}{l} \\ \\ \\ \leftarrow \\ \\ \leftarrow \\ \\ \\ \end{array}$$

dargestellt. Danach hat man ein lineares Gleichungssystem $\tilde{A}^{(k)}x = \tilde{b}^{(k)}$, für das man in der Situation ist, dass das k-te Diagonalelement der Matrix $\tilde{A}^{(k)}$ nicht verschwindet und man dann wie in Schritt (a) vorgehen kann.

Nach $(n-1)$ Schritten landet man damit beim linearen Gleichungssystem $A^{(n)}x = b^{(n)}$, wobei die Matrix $A^{(n)} \in \mathbb{R}^{n\times n}$ nach Konstruktion nun $a_{ij}^{(n)} = 0$ für alle $i, j \in \{1, \dots, n\}$ mit $i < j$ erfüllt, also in oberer Dreiecksgestalt ist.

(2) *Rückwärtssubstitution*: Das gestaffelte lineare Gleichungssystem $A^{(n)}x = b^{(n)}$ mit oberer Dreiecksmatrix $A^{(n)}$ löst man nun durch die in Kapitel 2.3.1 beschriebene Rückwärtssubstitution. Diese Lösung $x \in \mathbb{R}^n$ ist dann nach Konstruktion auch die (eindeutige) Lösung des ursprünglichen Gleichungssystems $Ax = b$.

Beispiel 2.27. Wir bestimmen mit dem Gaußschen Eliminationsverfahren 2.26 nun die Lösung $x \in \mathbb{R}^3$ des linearen Gleichungssystems

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{pmatrix} x = \begin{pmatrix} -1 \\ 3 \\ 5 \end{pmatrix}.$$

Wir schreiben für die Vorwärtselimination die Matrix sowie die rechte Seite in die erweiterte Koeffizientenmatrix und formen diese nach dem oben beschriebenen Verfahren um:

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & -1 \\ 2 & 0 & 1 & 3 \\ 3 & 2 & 1 & 5 \end{array}\right) \begin{array}{l} \cdot(-2) \\ + \\ \end{array} \quad \begin{array}{l} \cdot(-3) \\ \\ + \end{array} \longrightarrow \left(\begin{array}{ccc|c} 1 & 2 & 3 & -1 \\ 0 & -4 & -5 & 5 \\ 0 & -4 & -8 & 8 \end{array}\right) \begin{array}{l} \cdot(-1) \\ + \end{array} \longrightarrow \left(\begin{array}{ccc|c} 1 & 2 & 3 & -1 \\ 0 & -4 & -5 & 5 \\ 0 & 0 & -3 & 3 \end{array}\right)$$

(dafür mussten wir keine Zeilenvertauschungen vornehmen, so dass wir immer im Fall (a) waren). Damit können wir äquivalent zum ursprünglichen Problem auch das lineare Gleichungssystem

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

lösen. Aus der letzten Zeile lesen wir zunächst $x_3 = -1$ ab und schließen dann über Rückwärtssubstitution sukzessive auf

$$\begin{aligned} -4x_2 - 5x_3 = 5 \quad &\Rightarrow \quad x_2 = -\frac{1}{4}\big(5 + 5 \cdot (-1)\big) = 0, \\ x_1 + 2x_2 + 3x_3 = -1 \quad &\Rightarrow \quad x_1 = -1 - 2 \cdot 0 - 3 \cdot (-1) = 2. \end{aligned}$$

Damit haben wir die Lösung $x \in \mathbb{R}^3$ des gegebenen Gleichungssystems bestimmt als $x = (2, 0, -1)^T$.

Darstellung der Vorwärtselimination über Matrixmultiplikation. Tritt im k-ten Schritt bei der Vorwärtselimination des Gaußschen Eliminationsverfahren 2.26 der Fall (a) ein, so ergibt sich $A^{(k+1)}$ aus $A^{(k)}$ und $b^{(k+1)}$ aus $b^{(k)}$ durch Linksmultiplikation mit einer Frobeniusmatrix, nämlich

$$A^{(k+1)} = M^{(k)}A^{(k)} \quad \text{und} \quad b^{(k+1)} = M^{(k)}b^{(k)}$$

mit

$$M^{(k)} = \mathbb{1}_{n\times n} - \ell^{(k)} e^{(k)T} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & & 1 & 0 & & & \vdots \\ \vdots & & -\ell_{k+1,k} & 1 & \ddots & & \vdots \\ \vdots & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \ddots & 1 & 0 \\ 0 & & -\ell_{n,k} & 0 & \cdots & 0 & 1 \end{pmatrix} \quad \text{für } \ell^{(k)} \coloneqq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \ell_{k+1,k} \\ \vdots \\ \ell_{n,k} \end{pmatrix},$$

wobei wir $\ell_{ik} \coloneqq \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$ für $i,k \in \{1,\dots,n\}$ mit $i > k$ definiert haben. Für $x \in \mathbb{R}^n$ gilt dabei

$$A^{(k)}x = b^{(k)} \quad \Leftrightarrow \quad M^{(k)}A^{(k)}x = M^{(k)}b^{(k)} \quad \Leftrightarrow \quad A^{(k+1)}x = b^{(k+1)},$$

da $M^{(k)}$ eine reguläre Matrix ist. Damit sind die beiden linearen Gleichungssysteme $A^{(k)}x = b^{(k)}$ und $A^{(k+1)}x = b^{(k+1)}$ also äquivalent. Tritt sogar in allen Schritten der Fall (a) ein, so haben wir

$$A^{(n)} = M^{(n-1)} \cdots M^{(1)} A^{(1)}$$

und wir haben damit über die Vorwärtselimination die LR-Zerlegung von A gefunden, als

$$LR = A \qquad \text{mit} \quad L \coloneqq (M^{(1)})^{-1} \cdots (M^{(n-1)})^{-1} \text{ und } R \coloneqq A^{(n)}.$$

Man kann sich leicht überlegen, dass bei der Invertierung der Frobeniusmatrizen $M^{(1)},\dots,M^{(n-1)}$ und der anschließenden Produktbildung nur die Einträge unterhalb der Diagonale mit dem Faktor (-1) multipliziert und diese Werte dann addiert werden. Damit sind die zuvor definierten Werte ℓ_{ij} für $i,j \in \{1,\dots,n\}$ mit $i > j$ also tatsächlich die Einträge der resultierenden unteren Dreiecksmatrix L.

Beispiel 2.28. Für das die Matrix aus dem Gleichungssystem in Beispiel 2.27 haben wir

$$A^{(2)} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -4 & -5 \\ 0 & -4 & -8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{pmatrix} = M^{(1)}A$$

sowie

$$A^{(3)} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -4 & -5 \\ 0 & 0 & -3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -4 & -5 \\ 0 & -4 & -8 \end{pmatrix} = M^{(2)}A^{(2)},$$

so dass wir mit $L = (M^{(1)})^{-1}(M^{(2)})^{-1}$ und $R = A^{(3)}$ die folgende LR-Zerlegung erhalten:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 0 & 1 \\ 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} A^{(3)} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 0 & -4 & -5 \\ 0 & 0 & -3 \end{pmatrix}.$$

Für allgemeine reguläre Matrizen können für die Durchführung des Gaußschen Eliminationsverfahren auch Zeilenvertauschungen aus dem Fall (b) benötigt werden. Dies entspricht gerade der Multiplikation von links mit einer sogenannten Permutationsmatrix.

Definition 2.29 (Permutationsmatrix)**.** *Es sei $n \in \mathbb{N}$ und π eine beliebige Permutation der Menge $\{1,\dots,n\}$. Dann heißt die Matrix $P_\pi \in \mathbb{R}^{n\times n}$ definiert über*

$$P_\pi = (e^{(\pi(1))},\dots,e^{(\pi(n))})$$

(*d.h. mit* $P_\pi e^{(j)} = e^{(\pi(j))}$ *für alle* $j \in \{1, \dots, n\}$) *die* Permutationsmatrix *zur Permutation* π.

Bemerkungen 2.30.

(1) *Ist* π *eine beliebige Permutation der Menge* $\{1, \dots, n\}$ *und* π^{-1} *ihr Inverses* (*d.h. mit* $\pi(\pi^{-1}(j)) = j = \pi^{-1}(\pi(j))$ *für alle* $j \in \{1, \dots, n\}$), *so gilt*

$$(P_\pi)^{-1} = P_{\pi^{-1}} = (P_\pi)^T.$$

(2) *Sind* π_1, π_2 *Permutationen der Menge* $\{1, \dots, n\}$, *so ist* $\pi = \pi_1 \circ \pi_2$ *wieder eine Permutation und es gilt* $P_\pi = P_{\pi_1} P_{\pi_2}$.

Im Gaußschen Eliminationsverfahren 2.26 muss man im k-ten Schritt gegebenenfalls eine Transposition π_k durchführen, die die k-te Zeile mit der m-ten Zeile für ein $m \in \{k+1, \dots, n\}$ vertauscht. Wenn man in $\pi = \pi_{n-1} \circ \dots \circ \pi_1$ alle diesen Transpositionen als eine Permutation ausdrückt, so kann man für die Matrix $P_\pi A$ die LR-Zerlegung nach obigem Schema (nun ohne Zeilenvertauschungen) durchführen und hat damit:

Satz 2.31. *Ist* $A \in \mathbb{R}^{n \times n}$ *eine reguläre Matrix, so gibt es eine Permutationsmatrix* $P \in \mathbb{R}^{n \times n}$, *so dass* PA *eine LR-Zerlegung besitzt.*

Komplexität. Zur Berechnung der LR-Zerlegung einer (permutierten) regulären Matrix aus $\mathbb{R}^{n \times n}$ müssen wir im k-ten Schritt $n-k$ Divisionen für die Aufstellung von $M^{(k)}$ und $(n-k)^2$ Multiplikationen für die Aufstellung von $A^{(k+1)}$ durchführen. Insgesamt haben wir also (per Induktion)

$$\sum_{k=1}^{n-1} \left[(n-k) + (n-k)^2 \right] = \frac{n^3}{3} - \frac{n}{3}$$

Multiplikationen und Divisionen durchzuführen. Damit ist die Komplexität der Berechnung der LR-Zerlegung von der Ordnung $O(n^3)$.

Bemerkungen 2.32.

(1) *Da der Aufwand der Rückwärtssubstitution von der Ordnung* $O(n^2)$ *war, ist dieser für die Lösung eines* $(n \times n)$*-Gleichungssystems* $Ax = b$ *gegenüber des Aufwands der Berechnung der LR-Zerlegung vernachlässigbar und wir haben eine Komplexität der Ordnung* $O(n^3)$ *wie für die Aufstellung der LR-Zerlegung.*

(2) *Der Vorteil der LR-Zerlegung ist, dass man diese für eine gegebene Matrix nur einmal berechnen muss und dann für die Lösung des linearen Gleichungssystems zu verschiedenen rechten Seiten verwenden kann. Lösen wir es etwa zu den Einheitsvektoren* $e^{(1)}, \dots, e^{(n)}$ *als rechter Seite, so erhalten wir die inverse Matrix* A^{-1}, *und das ebenfalls mit einem Aufwand der Ordnung* $O(n^3)$.

Zusammenfassung. Ist eine reguläre Matrix $A \in \mathbb{R}^{n \times n}$ sowie ein Vektor $b \in \mathbb{R}^n$ gegeben, so kann man die Lösung des linearen Gleichungssystems $Ax = b$ in *drei* Probleme zerlegen:

- Finde zuerst über die Vorwärtselimination eine Permutationsmatrix $P \in \mathbb{R}^{n \times n}$, eine untere normierte Dreiecksmatrix $L \in \mathbb{R}^{n \times n}$ und eine obere Dreiecksmatrix $R \in \mathbb{R}^{n \times n}$ mit $PA = LR$,
- bestimme die Lösung $y \in \mathbb{R}^n$ des linearen Gleichungssystems $Ly = Pb$ durch Vorwärtssubstitution,
- und bestimme danach die Lösung $x \in \mathbb{R}^n$ des linearen Gleichungssystems $Rx = y$ durch Rückwärtssubstitution.

Für $x \in \mathbb{R}^n$ gilt dann

$$PAx = LRx = Ly = Pb,$$

so dass wegen der Regularität der Matrix P die Lösung des ursprünglichen Problems gefunden wurde.

2.3.4 Ausgewählte numerische Aspekte

Wir diskutieren in diesem Abschnitt kurz die Pivotsuche für eine verbesserte Genauigkeit der Näherungslösung für lineare Gleichungssysteme sowie ein effizientes Speicherschema der Zwischenergebnisse beim Gaußschen Eliminationsverfahren 2.26.

Pivotisierung. Bei exakter Rechnung führen Zeilenvertauschungen in linearen Gleichungssystemen zur gleichen Lösung, beim Rechnen mit Maschinenzahlen können geschickte Permutationen jedoch zu einem genaueren Ergebnis führen. Dieses Phänomen veranschaulichen wir zunächst anhand eines einfachen Beispiels.

Beispiel 2.33. Zu einem sehr kleinen Parameter $\varepsilon \in (0, \frac{1}{3})$ betrachten wir das lineare Gleichungssystem

$$A^{(1)}x = b^{(1)} \quad \text{mit } A^{(1)} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \text{ und } b^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Wir berechnen zunächst die Kondition der Matrix $A^{(1)}$ bezüglich der Zeilensummennorm

$$(A^{(1)})^{-1} = \frac{1}{\varepsilon - 1}\begin{pmatrix} 1 & -1 \\ -1 & \varepsilon \end{pmatrix} \quad \Rightarrow \quad \kappa_\infty(A^{(1)}) = \|A^{(1)}\|_\infty \|(A^{(1)})^{-1}\|_\infty = 2 \cdot \frac{2}{1-\varepsilon} = \frac{4}{1-\varepsilon}.$$

Damit ist das mathematische Problem prinzipiell gut konditioniert und wir erwarten, dass kleine relative Fehler in den Daten nur zu kleinen relativen Fehlern im Ergebnis führen.

- Wir berechnen nun bei exakter Rechnung zunächst die LR-Zerlegung von $A^{(1)}$ und führen dann die Vorwärtssubstitution bzw. Rückwärtssubstitution durch, um die Lösung von $L^{(1)}y = b$ bzw. von $R^{(1)}x = y$ zu bestimmen. Als LR-Zerlegung von $A^{(1)}$ erhalten wir dabei

  $$A^{(1)} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{pmatrix} \begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - \varepsilon^{-1} \end{pmatrix} = L^{(1)}R^{(1)}.$$

  Wir beobachten, dass bei $L^{(1)}$ und $R^{(1)}$ die Zahl ε^{-1} in den Einträgen auftritt, und man kann leicht $\kappa_\infty(L^{(1)}) = (1 + \varepsilon^{-1})^2$, $\kappa_\infty(R^{(1)}) = \varepsilon^{-2}$ nachrechnen. Folglich werden die Konditionszahlen von $L^{(1)}$ und $R^{(1)}$ jeweils sehr groß (im Vergleich zur Konditionszahl von $A^{(1)}$). Als Lösung finden wir

  $$L^{(1)}y = b^{(1)} \quad \Rightarrow \quad y = \begin{pmatrix} 1 \\ -\varepsilon^{-1} \end{pmatrix} \quad \text{und dann} \quad R^{(1)}x = y \quad \Rightarrow \quad x = \frac{1}{1-\varepsilon}\begin{pmatrix} -1 \\ 1 \end{pmatrix} \approx \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

- Wir wiederholen das Vorgehen von eben nun, rechnen dabei aber mit Maschinenzahlen. Sind $1, \varepsilon, \varepsilon^{-1}$ Maschinenzahlen und wird $1 - \varepsilon^{-1}$ auf $-\varepsilon^{-1}$ gerundet (wie das bei $\varepsilon = 10^{-4}$ und Genauigkeit auf drei Stellen der Fall ist), so ergibt sich als Zerlegung

  $$A^{(1)} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{pmatrix} \begin{pmatrix} \varepsilon & 1 \\ 0 & -\varepsilon^{-1} \end{pmatrix} = \tilde{L}^{(1)}\tilde{R}^{(1)}$$

  (also mit $\tilde{L}^{(1)} = L^{(1)}$ und $\tilde{R}^{(1)} \approx R^{(1)}$). Als Lösung finden wir dann

  $$\tilde{L}^{(1)}\tilde{y} = b^{(1)} \quad \Rightarrow \quad \tilde{y} = \begin{pmatrix} 1 \\ -\varepsilon^{-1} \end{pmatrix} \quad \text{und dann} \quad \tilde{R}^{(1)}\tilde{x} = \tilde{y} \quad \Rightarrow \quad \tilde{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

  Damit liegt ein großer relativer Fehler bei der Lösung $\tilde{x}$ im Vergleich zur exakten Lösung x vor. Dieser ist nicht durch die Kondition des mathematischen Problems bedingt (die ja gut war), sondern wurde durch die schlechte Kondition der Gleichungssysteme, die für die LR-Zerlegung gelöst werden mussten, herbeigeführt.

Wir vertauschen nun die beiden Zeilen im ursprünglichen linearen Gleichungssystem (wodurch die Kondition des Problems nicht verändert wird). Wir betrachten also das lineare Gleichungssystem

$$PA^{(1)}x = Pb^{(1)} \quad \text{für } P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ und folglich mit } PA^{(1)} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} \text{ und } Pb^{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Wir verwenden im Folgenden die Notationen $A^{(2)} := PA^{(1)}$ und $b^{(2)} := Pb^{(1)}$.

- Wir berechnen wieder die LR-Zerlegung von $A^{(2)}$ bei exakter Rechnung

  $$A^{(2)} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{pmatrix} = L^{(2)}R^{(2)}.$$

  Hier kann man leicht $\kappa_\infty(L^{(2)}) = (1+\varepsilon)^2$, $\kappa_\infty(R^{(2)}) = 2(2-\varepsilon)/(1-\varepsilon)$ nachrechnen. Die Konditionszahlen von $L^{(2)}$ und $R^{(2)}$ sind damit ähnlich zur Konditionszahl $\kappa_\infty(A^{(2)}) = \kappa_\infty(A^{(1)})$. Die exakte Lösung ergibt sich über

  $$L^{(2)}y = b^{(2)} \quad \Rightarrow \quad y = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{und dann} \quad R^{(2)}x = y \quad \Rightarrow \quad x = \frac{1}{1-\varepsilon}\begin{pmatrix} -1 \\ 1 \end{pmatrix} \approx \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

  (was natürlich die Lösung des linearen Gleichungssystems $A^{(1)}x = b^{(1)}$ ist).

- Bei Rechnung mit Maschinenzahlen wie oben (wobei nun $1-\varepsilon$ auf 1 gerundet wird) erhalten wir in diesem Fall als Zerlegung

  $$A^{(2)} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & 0 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \tilde{L}^{(2)}\tilde{R}^{(2)}$$

  (also mit $\tilde{L}^{(2)} = L^{(2)}$ und $\tilde{R}^{(2)} \approx R^{(2)}$). Als Lösung finden wir dann

  $$\tilde{L}^{(2)}\tilde{y} = b^{(2)} \quad \Rightarrow \quad \tilde{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{und dann} \quad \tilde{R}^{(2)}\tilde{x} = \tilde{y} \quad \Rightarrow \quad \tilde{x} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

  Hier liegt nur ein kleiner relativer Fehler bei der Lösung $\tilde{x}$ im Vergleich zur exakten Lösung x vor.

In diesem Beispiel trat ein sehr kleines *Pivotelement*, also Diagonalelement, in der Matrix $A^{(1)}$ auf, das sich in einer sehr großen Kondition von $L^{(1)}$ und $R^{(2)}$ und dann in einem sehr großen relativen Fehler im Ergebnis niederschlug. Damit ist das Verfahren *numerisch instabil*. Um unnötig kleine Pivotelemente zu vermeiden (durch die man dann teilen muss), nimmt man in der Praxis eine (partielle) Pivotsuche vor. Bei der *Spaltenpivotsuche* sucht man im k-ten Schritt der Vorwärtselimination für jedes $k \in \{1, 2, \ldots, n-1\}$ in der Teilspalte $(a_{kk}^{(k)}, \ldots, a_{nk}^{(k)})^T$ das betragsgrößte Element und tauscht die k-te Zeile mit dessen Zeile. Damit macht man also Zeilenvertauschungen wie im Fall (b) des Verfahrens 2.26 auch bei nicht-verschwindenden Pivotelementen, um die Genauigkeit des Ergebnisses zu verbessern.

Speicherschema. Anstelle der Matrix $A \in \mathbb{R}^{n\times n}$ möchte man am Ende nur die beiden Dreiecksmatrizen $L, R \in \mathbb{R}^{n\times n}$ sowie die Permutation π, die die Permutationsmatrix P vollständig bestimmt, speichern. Für die Speicherung der Zwischenergebnisse der LR-Zerlegung der permutierten Matrix PA, also der Matrizen $A^{(k)}$ sowie der dafür benötigten Frobenius-Matrizen $M^{(1)}, \ldots, M^{(k-1)}$ für $k \in 2, \ldots, n$, nutzt man die Struktur dieser Matrizen: anstelle der in jedem Schritt erzeugten Nullen von $A^{(k)}$ kann man nämlich die nicht-trivialen Elemente von $(M^{(k-1)})^{-1}$ eintragen und bekommt so das Speicherschema:

$$A \longrightarrow \left(\begin{array}{c|c} \begin{matrix} \ell_{21} \\ \vdots \\ \ell_{n1} \end{matrix} & A^{(2)} \end{array}\right) \longrightarrow \quad \ldots \quad \longrightarrow \left(\begin{array}{cc|c} \ell_{21} & & \\ \vdots & \ddots & A^{(n)} \\ \ell_{n1} & \cdots & \ell_{n,n-1} \end{array}\right).$$

Algorithmus 2.34: Gauß-Elimination mit Spaltenpivotsuche

Eingabe: Matrix $A \in \mathbb{R}^{n\times n}$, rechte Seite $b \in \mathbb{R}^n$, Permutation $\pi = (1,\dots,n)^T \in \mathbb{R}^n$

for $k = 1 : n-1$ **do** ▷ Schleife über die Zeilen
  Bestimme p mit $|a_{pk}| = \max\{|a_{ik}| : i \in \{k,\dots,n\}\}$ ▷ Spaltenpivotsuche
  Vertausche die Zeilen $a_{k:}$ mit $a_{p:}$
  Vertausche b_k und b_p
  Vertausche π_k und π_p
  if $a_{kk} = 0$ **then**
    Stop ▷ Abbruch, da A singulär ist
  end if
  for $i = k+1 : n$ **do** ▷ Schleife über verbliebene Zeilen
    $a_{ik} \coloneqq \frac{a_{ik}}{a_{kk}}$ ▷ Multiplikativer Faktor (Teil der Matrix L)
    for $j = k+1 : n$ **do** ▷ Schleife über verbliebene Spalten
      $a_{ij} \coloneqq a_{ij} - a_{ik}a_{kj}$ ▷ Update von A
    end for
    $b_i \coloneqq b_i - a_{ik}b_k$ ▷ Update von b
  end for
end for

Ausgabe: A, b, π

2.3.5 Gleichungssysteme mit symmetrischen, positiv definiten Matrizen

Als Ausblick wollen wir noch eine spezielle Klasse von Gleichungssystemen betrachten. Wir bezeichnen mit $\mathbb{R}^{n\times n}_{\text{spd}}$ den Raum aller symmetrischen, positiv definiten Matrizen aus $\mathbb{R}^{n\times n}$, also

$$\mathbb{R}^{n\times n}_{\text{spd}} \coloneqq \big\{A \in \mathbb{R}^{n\times n} : A = A^T \text{ und } x^TAx > 0 \text{ für alle } x \in \mathbb{R}^n \setminus \{0\}\big\}.$$

Solche Matrizen kommen etwa als Korrelationsmatrizen bei statistischen Berechnungen oder als Steifigkeitsmatrizen für die numerische Lösung von elliptischen Gleichungen mittels Finite-Elemente-Methoden vor. Wir sammeln kurz einige einfache Eigenschaften für symmetrische, positiv definite Matrizen.

Lemma 2.35. *Für eine Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ gelten die folgenden Aussagen:*

(i) *A ist regulär.*

(ii) *Die Diagonalelemente von A, also a_{ii} für $i \in \{1,\dots,n\}$, sind positiv.*

(iii) *Alle Eigenwerte von A sind positiv.*

Beweis.

(i) Wir gehen über Widerspruch vor und nehmen an, dass A nicht regulär ist. Dann gibt es einen Vektor $x \in \mathbb{R}^n \setminus \{0\}$ mit $Ax = 0$ und folglich auch mit $x^TAx = 0$, was im Widerspruch zur positiven Definitheit von A steht. Also muss A wie behauptet regulär sein.

(ii) Ist $e^{(i)}$ der i-te Einheitsvektor für ein $i \in \{1,\dots,n\}$ so gilt $a_{ii} = e^{(i)T}Ae^{(i)} > 0$ wegen der Voraussetzung der positiven Definitheit.

(iii) Ist $x \in \mathbb{R}^n$ mit $|x| = 1$ ein (normierter) Eigenvektor zum Eigenwert λ, so gilt $Ax = \lambda x$. Damit folgt aus der positiven Definitheit von A die Positivität von λ über $\lambda = \lambda|x|^2 = x^TAx > 0$. □

Bemerkung 2.36 (Kriterien für positive Definitheit)**.** *Es gibt zwei wichtige Kriterien für die positive Definitheit: eine symmetrische Matrix $A \in \mathbb{R}^{n\times n}$ ist genau dann positiv definit, wenn*

- *alle Eigenwerte von A positiv sind* (Eigenwertkriterium),

- *alle führenden Hauptminoren $M_1(A), \ldots, M_n(A)$ eine positive Determinante haben* (Hauptminorenkriterium von Hurwitz–Sylvester).

Für symmetrischen, positiv definiten Matrizen kann man nun auf eine Zerlegung schließen, die diese Struktur berücksichtigt.

Korollar 2.37 (Cholesky-Zerlegung)**.** *Zu jeder symmetrischen, positiv definiten Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ existiert eine eindeutige untere Dreiecksmatrix $G \in \mathbb{R}^{n\times n}$ mit*

$$A = GG^T \quad \text{mit } g_{ii} > 0 \text{ für alle } i \in \{1, \ldots, n\}.$$

Beispiel 2.38. Für die symmetrische Matrix

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 13 \end{pmatrix}$$

kann man sich leicht überlegen, dass sie positiv definit ist (entweder direkt über die Definition oder über eine der Beobachtungen, dass die Determinanten $M_1(A) = 1$ und $M_2(A) = 9$ der führenden Hauptminoren oder die beiden Eigenwerte $7 \pm 2\sqrt{10}$ positiv sind). Die LR-Zerlegung und die Cholesky-Zerlegung sind hier

$$\begin{aligned} A = \begin{pmatrix} 1 & 2 \\ 2 & 13 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & 9 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}. \end{aligned}$$

Die Cholesky-Zerlegung wird effizient berechnet, indem man mit dem Ansatz $A = GG^T$ startet, das Matrixprodukt berechnet und dann die Einträge der unteren Dreiecksmatrix G sukzessive über einen Koeffizientenvergleich ermittelt: dazu überlegt man sich, dass A durch seine untere Dreiecksmatrix bereits eindeutig bestimmt ist, und diese Einträge sind im Produkt GG^T gerade von der Form

$$a_{ij} \overset{!}{=} \sum_{k=1}^{j} g_{ik} g_{jk} \quad \text{für } i, j \in \{1, \ldots, n\} \text{ mit } j \le i.$$

Algorithmus 2.39: Cholesky-Zerlegung

Eingabe: Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$

1: **for** $j = 1 : n$ **do** ▷ Schleife über die Spalten
2: $\quad g_{jj} \coloneqq (a_{jj} - \sum_{k=1}^{j-1} g_{jk}^2)^{1/2}$ ▷ Bestimmung der Diagonaleinträge
3: $\quad$ **for** $i = j + 1 : n$ **do** ▷ Schleife über die Zeilen
4: $\quad\quad g_{ij} \coloneqq (a_{ij} - \sum_{k=1}^{j-1} g_{ik} g_{jk})/g_{jj}$
5: $\quad$ **end for**
6: **end for**

Ausgabe: untere Dreiecksmatrix G der Cholesky-Zerlegung

Der Aufwand für die Erzeugung der Cholesky-Zerlegung ist nur etwa halb so groß wie der Aufwand zur Erzeugung der LR-Zerlegung, und im Gegensatz zur allgemeinen LR-Zerlegung ist keine Pivotisierung erforderlich, was das Verfahren der Cholesky-Zerlegung besonders effizient und numerisch stabil macht. Ist die Cholesky-Zerlegung $A = GG^T$ bestimmt, so wird das Gleichungssystem $Ax = b$ dann wie bei der LR-Zerlegung durch eine Vorwärtssubstitution $Gy = b$ und eine Rückwärtssubstitution $G^T x = y$ gelöst.

2.4 Gradientenverfahren

Wir betrachten nun spezielle *iterative Verfahren* für die Lösung des linearen Gleichungssystems

$$Ax = b. \tag{2.3}$$

Iterative Verfahren spielen vor allem dann eine Rolle, wenn die Matrix A in einer besonderen Struktur vorliegt, die man ausnutzen möchte (etwa viele verschwindende Einträge), oder wenn die Dimension des Problems sehr groß ist und direkte Verfahren dann einen enorm hohen Rechen- und Speicheraufwand verursachen (Erinnerung: beim Gauß-Algorithmus wächst der Aufwand mit der dritten Potenz in der Dimension!). Ziel eines iterativen Verfahren ist es stets, ausgehend von einem Startwert $x^{(0)}$, eine Folge $\{x^{(k)}\}_{k\in\mathbb{N}}$ zu konstruieren, so dass

(1) die Folge der Iterierten möglichst schnell gegen die Lösung konvergiert,

(2) jede Iterierte $x^{(k+1)}$ möglichst einfach aus den Vorgängern $x^{(0)}, x^{(1)}, \dots, x^{(k)}$ berechnet werden kann.

Unter einer möglichst einfachen Berechnung jeder Iterierten in der zweiten Forderung versteht man typischerweise, dass der Aufwand nicht wesentlich mehr als eine Matrix-Vektor-Multiplikation sein sollte (die, falls die involvierte Matrix dünn besetzt ist, typischerweise nicht quadratisch, sondern nur linear in der Dimension wächst).

Im Folgenden konstruieren wir iterative Lösungsverfahren für lineare Gleichungsysteme (2.3) unter der Annahme, dass $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ *symmetrisch* und *positiv definit* ist, also $A = A^T$ und $x^T Ax > 0$ für alle $x \in \mathbb{R}^n \setminus \{0\}$ gelten (womit, wie wir uns bereits in Lemma 2.35 überlegt haben, A insbesondere eine reguläre Matrix ist). Ausgangspunkt der Konstruktion ist die Beobachtung, dass man die Problemstellung in dieser Situation äquivalent als ein Minimierungsproblem einer quadratischen Funktion formulieren kann:

Satz 2.40 (Äquivalente Formulierung als Minimierungsproblem)**.** *Es sei $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ und $b \in \mathbb{R}^n$. Für einen Vektor $x \in \mathbb{R}^n$ sind äquivalent:*

(i) *x ist die eindeutige Lösung von $Ax = b$;*

(ii) *definiert man eine quadratische Funktion $Q\colon \mathbb{R}^n \to \mathbb{R}$ mittels*

$$Q(y) \coloneqq \frac{1}{2} y^T Ay - b^T y \qquad \text{für } y \in \mathbb{R}^n, \tag{2.4}$$

so ist x das strikte Minimum von Q auf $\mathbb{R}^n$, d.h. es gilt

$$Q(x) < Q(y) \qquad \text{für alle } y \in \mathbb{R}^n \setminus \{x\}.$$

Beweis. Beweis über notwendige und hinreichende Kriterien für das Vorliegen einer Minimumstelle. Wir berechnen zunächst die erste und zweite Ableitung der Funktion Q. Wegen der Symmetrie von A haben wir für jedes $k \in \{1, \dots, n\}$ als partielle Ableitung

$$\begin{aligned}\frac{\partial}{\partial y_k} Q(y) &= \frac{1}{2}\frac{\partial}{\partial y_k}\left[\sum_{i,j=1}^n a_{ij} y_i y_j\right] - \frac{\partial}{\partial y_k}\left[\sum_{i=1}^n b_i y_i\right] \\ &= \frac{1}{2}\left[\sum_{j=1}^n a_{kj} y_j + \sum_{i=1}^n a_{ik} y_i\right] - b_k = \sum_{j=1}^n a_{kj} y_j - b_k = (Ay - b)_k,\end{aligned}$$

also erhalten wir als Gradienten von Q

$$\nabla Q(y) = Ay - b. \tag{2.5}$$

Für die zweite Ableitung gilt offensichtlich

$$D^2Q(y) = A.$$

Damit können wir nun einfach die Äquivalenz der Aussagen (i) und (ii) beweisen:

Implikation (ii) $\Rightarrow$ (i): Ist x ein Minimum von Q, so muss notwendigerweise gelten:

$$\nabla Q(x) = 0 \quad \Leftrightarrow \quad Ax - b = 0 \quad \Leftrightarrow \quad Ax = b.$$

Implikation (i) $\Rightarrow$ (ii): Erfüllt x umgekehrt die Bedingung $Ax = b$, so gilt nach obiger Äquivalenzumformung $\nabla Q(x) = 0$, und die strikte Minimalität folgt dann aus der Tatsache, dass $D^2Q(x) = A$ nach Voraussetzung positiv definit ist.

Alternativer direkter Beweis ohne mehrdimensionale Differentialrechnung. Wir betrachten im Folgenden Vektoren $x, z \in \mathbb{R}^n$. Zunächst nutzen wir die Symmetrie von A, um $x^TAz = x^TA^Tz = z^TAx$ einzusehen, und schreiben dann $Q(x+z)$ als

$$\begin{aligned} Q(x+z) &= \frac{1}{2}(x+z)^TA(x+z) - b^T(x+z) \\ &= \frac{1}{2}\big(x^TAx + x^TAz + z^TAx + z^TAz\big) - b^Tx - b^Tz \\ &= \frac{1}{2}x^TAx - b^Tx + \frac{1}{2}z^TAz + x^TAz - b^Tz \\ &= Q(x) + \frac{1}{2}z^TAz + (Ax-b)^Tz. \end{aligned}$$

Damit können wir für x nun die Mimierungseigenschaft für die Funktion Q in Zusammenhang zur Lösungseigenschaft $Ax = b$ bringen:

Implikation (ii) $\Rightarrow$ (i): Ist x ein Minimum von Q, so betrachten wir den Strahl $z = -t(Ax-b)$ für $t \in \mathbb{R}$ und wissen dann

$$\begin{aligned} Q(x - t(Ax-b)) - Q(x) &= \frac{1}{2}t^2(Ax-b)^TA(Ax-b) - t(Ax-b)^T(Ax-b) \\ &\leq \Big(\frac{1}{2}t^2\|A\|_2 - t\Big)\|Ax-b\|_2^2. \end{aligned}$$

Auf der rechten Seite dominiert der lineare Term in t den quadratischen Term in t für hinreichend kleine Werte $t > 0$. Wäre $Ax \neq b$, so wäre der gesamte Ausdruck auf der rechten Seite für solche Werte negativ, im Widerspruch zur Minimalität von x. Folglich muss $Ax = b$ gelten.

Implikation (i) $\Rightarrow$ (ii): Erfüllt x umgekehrt die Bedingung $Ax = b$, so erhalten wir über die positive Definitheit von A

$$Q(x+z) - Q(x) = \frac{1}{2}z^TAz > 0 \quad \text{für } z \neq 0.$$

Folglich ist x ein strikter Minimierer von Q. □

Die allgemeine Minimierung eines quadratischen Funktionals ist natürlich immer noch genauso schwierig wie die Lösung des zugehörigen linearen Gleichungssystems selbst zu finden, aber in gewisser Weise kann man bei der Minimierung sukzessive vorgehen, indem man ausnutzt, dass es sehr einfach ist, für einen festen Vektor $x \in \mathbb{R}^n$ und eine vorgegebene Richtung $d \in \mathbb{R}^n$ die quadratische Funktion

$$\mathbb{R} \ni \alpha \mapsto Q(x + \alpha d)$$

zu minimieren. Dadurch wollen wir eine Folge von Iterierten $\{x^{(k)}\}_{k\in\mathbb{N}}$ in $\mathbb{R}^n$ finden, entlang derer die Funktionswerte $\{Q(x^{(k)})\}_{k\in\mathbb{N}}$ eine monoton fallende Folge darstellen (und optimalerweise gegen das Minimum von Q auf $\mathbb{R}^n$ konvergiert).

Verfahren 2.41 (Allgemeines Abstiegsverfahren)**.** Für eine gegebene Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ und einen Vektor $b \in \mathbb{R}^n$ folgt man im *allgemeinen Abstiegsverfahren* der Vorschrift: zu einem Startvektor $x^{(0)}$ und $k \in \mathbb{N}_0$

(1) wählt man eine *Abstiegsrichtung* $d^{(k)} \in \mathbb{R}^n \setminus \{0\}$,

(2) bestimmt dann die *Schrittweite* $\alpha^{(k)}$ über das eindimensionale Minimierungsproblem

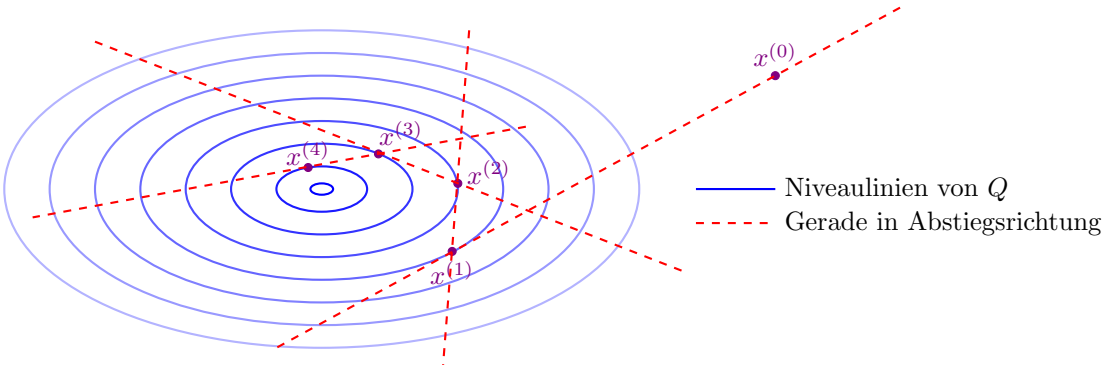
$$Q\big(x^{(k)} + \alpha^{(k)} d^{(k)}\big) = \min\big\{Q\big(x^{(k)} + \alpha d^{(k)}\big) : \alpha \in \mathbb{R}\big\},$$

(3) und setzt schließlich als *nächste Iterierte*

$$x^{(k+1)} := x^{(k)} + \alpha^{(k)} d^{(k)}.$$

Man bricht das Verfahren ab, sobald das *Residuum* $r^{(k)} := b - Ax^{(k)}$, also der Fehler im Ergebnis, unter einer vorgegebenen Genauigkeit liegt.

In zwei Dimensionen lässt sich dieses Verfahren graphisch gut veranschaulichen. Die Niveaulinie $N_c := \{x \in \mathbb{R}^2 : Q(x) = c\}$ der quadratischen Funktion Q bildet in diesem Fall für jedes Niveau $c > \min\{Q(x) : x \in \mathbb{R}^2\}$ eine Ellipse. Zu einer gegebenen Iterierten $x^{(k)}$ (oder dem Startwert $x^{(0)}$) und einer gegebenen Suchrichtung $d^{(k)}$ sucht man den kleinsten Wert von Q auf der durch die Suchrichtung und die Iterierte vorgegebene Gerade. Dieser kleinste Wert wird gerade in dem Berührpunkt der Gerade mit einer Ellipse angenommen. Dieser Berührpunkt stellt dann die neue Iterierte $x^{(k+1)}$ dar, von dem aus man dann eine neue Suchrichtung wählt und weitersucht.



Wir halten bereits einige nützliche Eigenschaften des allgemeinen Abstiegsverfahrens fest.

Lemma 2.42. *Für eine gegebene Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ und einen Vektor $b \in \mathbb{R}^n$ gilt beim allgemeinen Abstiegsverfahren* 2.41 *für jedes $k \in \mathbb{N}_0$:*

(i) *die Schrittweite $\alpha^{(k)}$ ist gegeben über die Formel*

$$\alpha^{(k)} = \frac{(b - Ax^{(k)})^T d^{(k)}}{d^{(k)T} A d^{(k)}} = \frac{r^{(k)T} d^{(k)}}{d^{(k)T} A d^{(k)}} = \frac{\langle\, r^{(k)}, d^{(k)} \,\rangle_2}{\langle\, d^{(k)}, d^{(k)} \,\rangle_A},$$

wobei $\langle\, \cdot, \cdot \,\rangle_2$ der euklidische und $\langle\, \cdot, \cdot \,\rangle_A$ das durch A induzierte Skalarprodukt bezeichnet (siehe Beispiel 2.8*),*

(ii) *es gilt die Orthogonalitätsbeziehung $r^{(k+1)} \perp d^{(k)}$, also*

$$\langle\, r^{(k+1)}, d^{(k)} \,\rangle_2 = 0,$$

(iii) *es gilt*

$$Q(x^{(k+1)}) \le Q(x^{(k)}).$$

Beweis. Übung. □

Bemerkung 2.43. *Bei diesem Verfahren ist der Rechenaufwand für die Berechnung der Iterierten wie gefordert relativ gering. Dagegen hängt die Konvergenz der Iterierten gegen die Lösung noch von der Wahl der Richtungen ab. Prinzipiell ist es möglich, die Richtungen so schlecht zu wählen, dass die Iterierten nicht gegen die Lösung konvergieren (beispielsweise ist das typischerweise der Fall, wenn man die Richtungen $d^{(k)} = d \in \mathbb{R}^n$ für alle $k \in \mathbb{N}^{(0)}$ festhält, so dass $x^{(k)} = x^{(1)}$ für alle $k \in \mathbb{N}$ gilt). Wir müssen uns also noch eine geschickte Wahl von Richtungen überlegen, so dass die Iterierten gegen die Lösung konvergieren, und dies auch noch möglichst schnell.*

Verfahren des steilsten Abstiegs. Eine naheliegende Wahl der Abstiegsrichtung im allgemeinen Abstiegsverfahren 2.41 ist die Richtung, in die die Funktion Q am stärksten abfällt. Wir erinnern daran, dass der Gradient gerade die Richtung und den Betrag des stärksten Anstiegs angibt. Somit wählt man zur aktuellen Iterierten $x^{(k)}$ als Abstiegsrichtung

$$d^{(k)} := -\nabla Q(x^{(k)}) = b - Ax^{(k)} = r^{(k)},$$

also gerade das aktuelle Residuum. Dies ist einfach zu berechnen und verschwindet nur dann, wenn $x^{(k)}$ bereits die gesuchte Lösung ist. Als Schrittlänge ergibt sich nach Lemma 2.42 (i)

$$\alpha^{(k)} = \frac{\langle r^{(k)}, d^{(k)} \rangle_2}{\langle d^{(k)}, d^{(k)} \rangle_A} = \frac{\langle r^{(k)}, r^{(k)} \rangle_2}{\langle r^{(k)}, r^{(k)} \rangle_A}.$$

Die Orthogonalitätsbeziehung aus Lemma 2.42 (i) ist nun $r^{(k+1)} \perp r^{(k)}$, was gerade bedeutet, dass *zwei aufeinanderfolgende* Abstiegsrichtungen senkrecht aufeinander stehen. Für die resultierende Folge $\{x^{(k)}\}_{k\in\mathbb{N}}$ gilt nun tatsächlich Konvergenz gegen die Lösung des linearen Gleichungssystems $Ax = b$, und zwar unabhängig vom gewählten Startwert $x^{(0)}$. Die Konvergenz kann jedoch sehr langsam sein (und zwar dann, wenn die Konditionszahl der Matrix sehr groß ist).

Algorithmus 2.44: Verfahren des steilsten Abstiegs

Eingabe: Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$, rechte Seite $b \in \mathbb{R}^n$, maximale Anzahl N an Iterationsschritten, Startvektor $x^{(0)} \in \mathbb{R}^n$

1: $r^{(0)} := b - Ax^{(0)}$ ▷ Initiales Residuum
2: **for** $k = 0 : N-1$ **do** ▷ Iterationsschleife
3: $\quad \alpha := \frac{r^{(k)T} r^{(k)}}{r^{(k)T} A d^{(k)}}$ ▷ Optimale Schrittweite
4: $\quad x^{(k+1)} := x^{(k)} + \alpha r^{(k)}$ ▷ Neue Iterierte
5: $\quad r^{k+1} = r^{(k)} - \alpha A r^{(k)} (= b - Ax^{(k+1)})$ ▷ Neues Residuum
6: $\quad$ **if** $\|r^{(k+1)}\|$ hinreichend klein **then**
7: $\quad\quad$ Stop ▷ Abbruch, da hinreichend nah an der Lösung
8: $\quad$ **end if**
9: **end for**

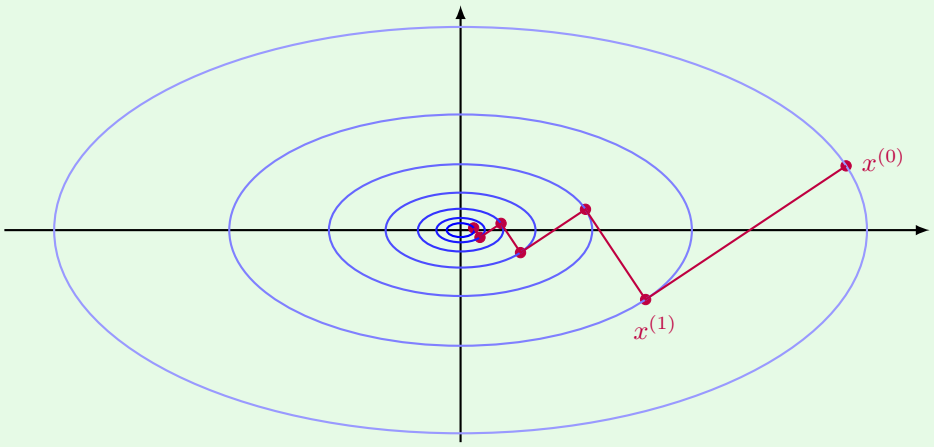
Ausgabe: Näherungslösung x von $Ax = b$

Beispiel 2.45. Wir veranschaulichen das Verfahren des steilsten Abstiegs im zweidimensionalen Fall für die Diagonalmatrix

$$A = \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix} \quad \text{für ein } a > 0$$

und $b = 0$, so dass die quadratische Funktion $Q\colon \mathbb{R}^2 \to \mathbb{R}$ gerade gegeben ist durch

$$Q(y) = \frac{1}{2}\left(ay_1^2 + y_2^2\right) \quad \text{für } y \in \mathbb{R}^2.$$



In diesem Fall liegen die Iterierten auf einem Zick-Zack-Kurs. Sind die Ellipsen dabei sehr langgestreckt (was für die Fälle $a \ll 1$ und $a \gg 1$ passiert, in denen auch die Konditionszahl von A sehr groß wird), so werden viele Schritte benötigt, um in die Nähe des Minimierers $x = 0$ der Funktion Q (und damit der Lösung des linearen Gleichungssystems $Ax = 0$) zu gelangen.

Verfahren der konjugierten Gradienten. Wir versuchen nun das Zick-Zack-Verhalten der Iterierten aus dem Verfahren des steilsten Gradienten zu vermeiden, indem wir die Suchrichtungen so geschickt wählen, dass sie zum einen linear unabhängig sind und dass zum anderen die neue Iterierte $x^{(k+1)}$ nicht nur optimal bezüglich der aktuellen Suchrichtung, sondern sogar optimal bezüglich aller früheren Suchrichtungen ist, also

$$Q(x^{(k+1)}) = \min\left\{Q\big(x^{(k+1)} + \alpha d^{(\ell)}\big) : \alpha \in \mathbb{R}\right\} \quad \text{für } \ell \in \{0, 1, \dots, k\} \tag{2.6}$$

gilt. Wir überlegen uns nun, welche Anforderungen dies an die Suchrichtungen im allgemeinen Abstiegsverfahren 2.41 stellt. Dass die neue Iterierte optimal in Richtung $d^{(\ell)}$ für $\ell \in \{0, 1, \dots, k\}$ sein soll, bedeutet wegen $\nabla Q(x^{(k+1)}) = Ax^{(k+1)} - b = -r^{(k+1)}$ gerade

$$0 = \nabla Q(x^{(k+1)})^T d^{(\ell)} \quad \Leftrightarrow \quad r^{(k+1)} \perp d^{(\ell)}$$

(vgl. Lemma 2.42 (ii)). Schreiben wir nun

$$x^{(k+1)} = x^{(\ell+1)} + \sum_{j=\ell+1}^{k} \alpha^{(j)} d^{(j)},$$

so erhalten wir für die entsprechenden Residuen

$$r^{(k+1)} = b - Ax^{(k+1)} = r^{(\ell+1)} + \sum_{j=\ell+1}^{k} \alpha^{(j)} A d^{(j)}. \tag{2.7}$$

Da nach Konstruktion des allgemeinen Abstiegsverfahrens 2.41 $r^{(\ell+1)} \perp d^{(\ell)}$ gilt, siehe Lemma 2.42 (ii), ist es für $r^{(k+1)} \perp d^{(\ell)}$ ausreichend, wenn $Ad^{(j)} \perp d^{(\ell)}$ für alle $j \in \{\ell + 1, \dots, k\}$ gilt. Da $\ell \in \{0, \dots, k\}$ beliebig war, folgern wir als hinreichende Bedingung für die Optimalität in alle frühere Suchrichtungen

$$Ad^{(j)} \perp d^{(\ell)} \quad \Leftrightarrow \quad d^{(\ell)T} A d^{(j)} = 0 \quad \text{für alle } j, \ell \in \{\ell + 1, \dots, k\} \text{ mit } j \neq \ell.$$

Solche Richtungen bezeichnet man als A-orthogonal:

Definition 2.46 (A-orthogonale Vektoren)**.** *Zwei Vektoren $v, w \in \mathbb{R}^n \setminus \{0\}$ heißen* A-konjugiert *oder* A-orthogonal *für eine Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$, falls*

$$Av \perp w \quad \Leftrightarrow \quad Aw \perp v \quad \Leftrightarrow \quad v^T Aw = \langle v, w \rangle_A = 0.$$

Bemerkung 2.47. *Man kann leicht nachweisen, dass paarweise A-konjugierte Vektoren $v^{(1)}, \dots, v^{(k)} \in \mathbb{R}^n \setminus \{0\}$ für ein $k \leq n$ linear unabhängig sind. Insbesondere bilden diese dann im Fall $k = n$ eine Basis des $\mathbb{R}^n$ und man kann einen gegebenen Vektor $v \in \mathbb{R}^n$ daher eindeutig schreiben als*

$$v = \sum_{j=1}^{n} \beta_j v^{(j)} \quad \text{mit Koeffizienten } \beta_j = \frac{v^T A v^{(j)}}{v^{(j)T} A v^{(j)}} = \frac{\langle v, v^{(j)} \rangle_A}{\langle v^{(j)}, v^{(j)} \rangle_A} \text{ für } j \in \{1, \dots, n\}.$$

Um nun paarweise A-orthogonale Suchrichtungen zu erzeugen, so dass die neue Iterierte die Optimalitätsbedingung (2.6) in alle früheren Richtungen erfüllt, starten wir anfangs mit der Richtung des steilsten Abstiegs und setzen dafür

$$d^{(0)} = r^{(0)} = -\nabla Q(x^{(0)}) = b - Ax^{(0)}.$$

Dann modifizieren wir allerdings das Verfahren des steilsten Abstiegs und nehmen vom Residuum lediglich den *A-orthogonalen Anteil zu den früheren Richtungen*. Damit setzen wir für $k \in \{0, 1, \dots, n-2\}$ mit $r^{(k+1)} \neq 0$

$$d^{(k+1)} = r^{(k+1)} - \sum_{j=0}^{k} \frac{\langle r^{(k+1)}, d^{(j)} \rangle_A}{\langle d^{(j)}, d^{(j)} \rangle_A} d^{(j)}.$$

So bekommen wir tatsächlich paarweise A-orthogonale Suchrichtungen:

Lemma 2.48. *Es sei $A \in \mathbb{R}^{n\times n}_{\text{spd}}$ und $b \in \mathbb{R}^n$. Ist $x^{(0)} \in \mathbb{R}^n$ ein beliebiger Startvektor und sind Suchrichtungen $d^{(0)}, d^{(1)}, \dots, d^{(k)}$ für ein $k \in \{0, 1, \dots, n-1\}$ wie oben definiert, so gelten die folgenden Eigenschaften:*

(i) *Die Suchrichtungen $d^{(0)}, d^{(1)}, \dots, d^{(k)}$ sind paarweise A-orthogonal, d.h. es gilt $\langle d^{(i)}, d^{(j)} \rangle_A = 0$ für alle $i, j \in \{0, 1, \dots, k\}$ mit $i \neq j$. Insbesondere sind sie damit linear unabhängig.*

(ii) *Es gilt $D_k = \operatorname{span}\{d^{(0)}, d^{(1)}, \dots, d^{(k)}\} = \operatorname{span}\{r^{(0)}, r^{(1)}, \dots, r^{(k)}\} =: R_k$.*

(iii) *Es gilt $r^{(k+1)} \perp D_k$.*

(iv) *Die Vektoren $r^{(k+1)}$ und $d^{(\ell)}$ sind A-konjugiert für jedes $\ell \in \{0, 1, \dots, k-1\}$, d.h. es gilt $\langle r^{(k+1)}, d^{(\ell)} \rangle_A = 0$, und für $\ell = k+1$ gilt $r^{(k+1)T} d^{(k+1)} = \|r^{(k+1)}\|_2^2$.*

★ *Beweis.* Die Aussagen (i)–(iii) beweisen wir über vollständige Induktion. Der Induktionsanfang $k = 0$ ist für (i) trivial, für (ii) wegen $d^{(0)} = r^{(0)}$ klar, und für (iii) die Aussage in Lemma 2.42 (ii). Wir nehmen nun an, dass die Aussagen (i)–(iii) jeweils für $k-1$ anstelle von k gelten.

(i) Für $\ell < k$ folgt nach Definition von $d^{(k)}$ sowie über die Induktionsannahme

$$\langle d^{(k)}, d^{(\ell)} \rangle_A = \langle r^{(k)}, d^{(\ell)} \rangle_A - \sum_{j=0}^{k-1} \frac{\langle r^{(k)}, d^{(j)} \rangle_A}{\langle d^{(j)}, d^{(j)} \rangle_A} \underbrace{\langle d^{(j)}, d^{(\ell)} \rangle_A}_{=\delta_{j\ell} \langle d^{(j)}, d^{(j)} \rangle_A} = \langle r^{(k)}, d^{(\ell)} \rangle_A - \langle r^{(k)}, d^{(\ell)} \rangle_A = 0.$$

Damit ist gezeigt, dass $d^{(k)}$ zu jeder der früheren Suchrichtung $d^{(\ell)}$ mit $\ell \in \{0, 1, \dots, k-1\}$ A-orthogonal ist. Über die Induktionsannahme ist damit die paarweise A-Orthogonalität aller Suchrichtungen $d^{(0)}, d^{(1)}, \dots, d^{(k)}$ bewiesen.

(ii) Nach Definition von $d^{(k)}$ haben wir mithilfe der Induktionsannahme

$$r^{(k)} = d^{(k)} + \sum_{j=0}^{k-1} \frac{\langle\, r^{(k)}, d^{(j)} \,\rangle_A}{\langle\, d^{(j)}, d^{(j)} \,\rangle_A}\, d^{(j)} \in D_k,$$

$$d^{(k)} = r^{(k)} - \sum_{j=0}^{k-1} \frac{\langle\, r^{(k)}, d^{(j)} \,\rangle_A}{\langle\, d^{(j)}, d^{(j)} \,\rangle_A}\, \underbrace{d^{(j)}}_{\in R_{k-1}} \in R_k.$$

Damit folgt sofort $D_k = R_k$.

(iii) Wegen der Darstellung $r^{(k+1)} = r^{(k)} - \alpha^{(k)} A d^{(k)}$ des Residuums (vgl. (2.7)) folgt zunächst $r^{(k+1)} \perp D_{k-1}$ aus der Induktionsannahme sowie der A-Orthogonalität von $d^{(0)}, d^{(1)}, \ldots, d^{(k)}$ aus (i). Da wir $r^{(k+1)} \perp d^{(k)}$ wieder nach Konstruktion des allgemeinen Abstiegsverfahrens haben, siehe Lemma 2.42 (ii), ist auch $r^{(k+1)} \perp D_k$ gezeigt.

Damit sind nun die Eigenschaften (i)–(iii) bewiesen. Die letzte Eigenschaft lässt Lemma (iv) sich nun einfach nachrechnen.

(iv) Wir schreiben zunächst $r^{(\ell+1)} - r^{(\ell)} = -\alpha^{(\ell)} A d^{(\ell)}$. Damit folgt für $\ell \in \{0, 1, \ldots, k-1\}$ über die Aussagen (ii) und (iii) dann

$$\alpha^{(\ell)} \langle\, r^{(k+1)}, d^{(\ell)} \,\rangle_A = \langle\, r^{(k+1)}, \alpha^{(\ell)} A d^{(\ell)} \,\rangle_2 = -\langle\, r^{(k+1)}, r^{(\ell+1)} - r^{(\ell)} \,\rangle_2 = 0.$$

Dies impliziert die Behauptung für $\ell \in \{0, 1, \ldots, k-1\}$, da $\alpha^{(\ell)} = 0$ zu $r^{(\ell+1)} = \ldots = r^{(k+1)} = 0$ führen würde. Für $\ell = k+1$ folgt nach Definition von $d^{(k+1)}$ und mithilfe von (iii)

$$r^{(k+1)T} d^{(k+1)} = r^{(k+1)T} r^{(k+1)} - \sum_{j=0}^{k} \frac{\langle\, r^{(k+1)}, d^{(j)} \,\rangle_A}{\langle\, d^{(j)}, d^{(j)} \,\rangle_A}\, \underbrace{r^{(k+1)T} d^{(j)}}_{=0} = \|r^{(k+1)}\|_2^2. \qquad \square$$

Bemerkung 2.49. *Man kann über Lemma* 2.48 *nun leicht nachrechnen, dass sich die Berechnungen der neuen Suchrichtung und der optimalen Schrittweite im Verfahren der konjugierten Gradienten für alle* $k \in \{0, 1, \ldots, n-2\}$ *mit* $r^{(k+1)} \neq 0$ *vereinfachen zu*

$$d^{(k+1)} = r^{(k+1)} + \frac{\|r^{(k+1)}\|_2^2}{\|r^{(k)}\|_2^2}\, d^{(k)} \quad \textit{und} \quad \alpha^{(k+1)} = \frac{\|r^{(k+1)}\|_2^2}{\|d^{(k+1)}\|_A^2}.$$

Satz 2.50. *Es sei* $A \in \mathbb{R}^{n\times n}_{\mathrm{spd}}$ *eine symmetrische, positiv definite Matrix und* $b \in \mathbb{R}^n$ *ein Vektor. Zu jedem Startvektor* $x^{(0)} \in \mathbb{R}^n$ *liefert das Verfahren der konjugierten Gradienten bei exakter Rechnung in maximal* n *Iterationsschritten die Lösung* $x \in \mathbb{R}^n$ *des linearen Gleichungssystems* $Ax = b$.

Beweis. Wir unterscheiden zwei Fälle. Erfüllt das Residuum im Verfahren der konjugierten Gradienten $r^{(\ell)} = b - Ax^{(\ell)} = 0$ für ein $\ell \in \{0, 1, \ldots, n-1\}$, so haben wir $Ax^{(\ell)} = b$, also ist die Iterierte $x^{(\ell)}$ bereits die Lösung von $Ax = b$. Verschwindet andernfalls keines der Residuen $r^{(0)}, r^{(1)}, \ldots, r^{(n-1)}$, so können wir wie oben die A-konjugierten Suchrichtungen $d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}$ definieren, die dann gemäß Bemerkung 2.47 eine Basis des $\mathbb{R}^n$ bilden. Da $r^{(n)} \perp d^{(0)}, d^{(1)}, \ldots, d^{(n-1)}$ nach Lemma 2.48 (iii) gilt, muss bereits $r^{(n)} = 0$ gelten, was entsprechend zum ersten Fall bedeutet, dass $x^{(n)}$ die Lösung zu $Ax = b$ ist. Damit ist mit diesem Verfahren in jedem Fall die Lösung des linearen Gleichungssystems $Ax = b$ nach höchstens n Iterationsschritten gefunden. $\square$

Bemerkungen 2.51.

(1) *Da das Verfahren der konjugierten Gradienten bei exakter Rechnung nach spätestens* n *Iterationen die Lösung darstellt, handelt es sich eigentlich um ein* direktes Verfahren (*das aber insbesondere wegen Rundungsfehlern in der Praxis iterativ angewendet wird*). *Für dünn besetzte Matrizen* $A \in \mathbb{R}^{n\times n}_{\mathrm{spd}}$ *handelt es sich dabei um eines der effizientesten Iterationsverfahren.*

(2) *Es gilt die* Fehlerabschätzung

$$\|x^{(k)} - x\|_A \leq 2\left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}\right)^k \|x^{(0)} - x\|_A \quad \textit{für } k \in \mathbb{N}$$

Eine deutliche Verbesserung der Konvergenzgeschwindigkeit *erreicht man über* Vorkonditionierung*, wobei man das äquivalente lineare Gleichungssystem*

$$\underbrace{V^{-1}AV^{-T}}_{=\tilde{A}}\underbrace{V^T x}_{=\tilde{x}} = \underbrace{V^{-1}b}_{=\tilde{b}}$$

für eine geeignete reguläre Matrix $V \in \mathbb{R}^{n\times n}$ betrachtet. Diese Matrix soll einerseits möglichst einfach zu bestimmen und zu invertieren sein, und andererseits soll die transformierte Matrix $\tilde{A}$ eine bessere Kondition als A besitzen.

Algorithmus 2.52: Verfahren der konjugierten Gradienten

Eingabe: Matrix $A \in \mathbb{R}^{n\times n}_{\text{spd}}$, rechte Seite $b \in \mathbb{R}^n$, maximale Anzahl N an Iterationsschritten, Startvektor $x^{(0)} \in \mathbb{R}^n$

$r^{(0)} := b - Ax^{(0)}$ ▷ Initiales Residuum
$d^{(0)} := r^{(0)}$ ▷ Initiale Suchrichtung
for $k = 0 : N-1$ **do** ▷ Iterationsschleife
  $\alpha := \frac{r^{(k)T} r^{(k)}}{d^{(k)T} A d^{(k)}}$ ▷ Optimale Schrittweite
  $x^{(k+1)} := x^{(k)} + \alpha d^{(k)}$ ▷ Neue Iterierte
  $r^{(k+1)} = r^{(k)} - \alpha A d^{(k)} (= b - Ax^{(k+1)})$ ▷ Neues Residuum
  $d^{(k+1)} = r^{(k+1)} + \frac{r^{(k+1)T} r^{(k+1)}}{r^{(k)T} r^{(k)}} d^{(k)}$ ▷ Neue Suchrichtung
  if $\|r^{(k+1)}\|$ hinreichend klein **then**
    Stop ▷ Abbruch, da hinreichend nah an der Lösung
  end if
end for

Ausgabe: Näherungslösung x von $Ax = b$

2.5 Lineare Ausgleichsrechnung

In vielen Anwendungen möchte man Modellparameter $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ finden, die sich typischerweise nur indirekt mithilfe einer Modellfunktion $\varphi(t, x_1, \ldots, x_n)$ messen lassen. Hat man den Zusammenhang $b(t) := \varphi(t, x_1, \ldots, x_n)$ (z.B. durch ein Materialgesetz), so misst man etwa

$$(t_1, b_1), \ldots, (t_m, b_m)$$

und möchte damit auf die Parameter $x_1, \ldots, x_n$ schließen. Typischerweise macht man dabei mehr Messungen, als es Unbekannte gibt, also $m \geq n$, und es treten normalerweise auch Messfehler auf, so dass man keine Parameter $x_1, \ldots, x_n \in \mathbb{R}$ mit

$$b_i = \varphi(t_i, x_1, \ldots, x_n) \quad \text{für alle } i \in \{1, \ldots, m\}$$

finden kann, die also die Messungen exakt widerspiegeln. Stattdessen suchen wir nur nach einer möglichst guten Lösung des Problems, bei der der Fehlervektor $\Delta \in \mathbb{R}^m$ mit

$$\Delta_i := b_i - \varphi(t_i, x_1, \ldots, x_n) \quad \text{für } i \in \{1, \ldots, m\}$$

in einer gegebenen Norm minimal wird. Wir wollen hier die (mathematisch relativ einfach zu behandelnde) *Gaußsche Methode der kleinsten Fehlerquadrate* diskutieren, bei der also der Ausdruck

$$\sum_{i=1}^m \Delta_i^2 = \sum_{i=1}^m \big(b_i - \varphi(t_i, x_1, \ldots, x_n)\big)^2$$

unter allen $(x_1, \dots, x_n) \in \mathbb{R}^n$ minimiert werden soll. Wir beschränken uns hierbei auf lineare Probleme in folgendem Sinne:

Definition 2.53 (lineares Ausgleichsproblem)**.** *Ist die Funktion φ oben linear in den Unbekannten $x_1, \dots, x_n \in \mathbb{R}$, d.h. gilt*

$$\varphi(t, x_1, \dots, x_n) = a_1(t)x_1 + \dots + a_n(t)x_n$$

für Funktionen $a_j \colon \mathbb{R} \to \mathbb{R}$ für $j \in \{1, \dots, n\}$, so sprechen wir von einem linearen Ausgleichsproblem.

Definieren wir in dieser Situation $A \in \mathbb{R}^{m \times n}$ über $a_{ij} \coloneqq a_j(t_i)$ für $i \in \{1, \dots, m\}$ und $j \in \{1, \dots, n\}$, so ist die Formulierung des linearen Ausgleichsproblem (mit Minimierung des Fehlervektors in $\|\cdot\|_2$) das Problem:

$$\text{finde } x \in \mathbb{R}^n \text{ mit } \|Ax - b\|_2 = \min\left\{\|Ay - b\|_2 \colon y \in \mathbb{R}^n\right\}. \tag{2.8}$$

Die Lösung x bezeichnet man dann auch als die *Lösung mit kleinsten Fehlerquadraten*. Zu beachten ist hierbei, dass es sich wegen $m \geq n$ im Allgemeinen um ein überbestimmtes Problem handelt.

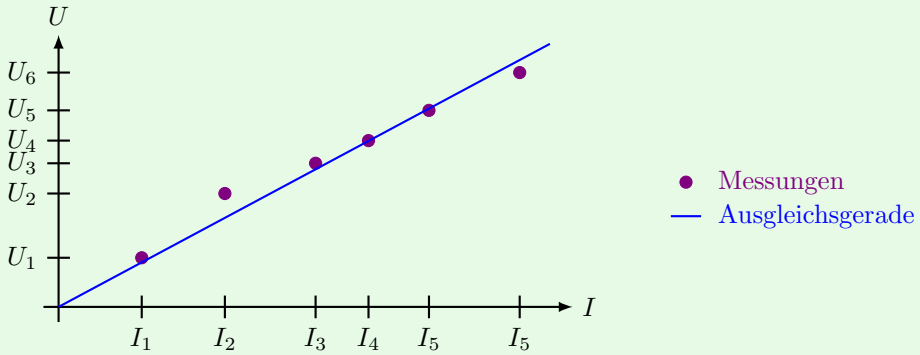
Beispiel 2.54 (Widerstandsbestimmung in einem Stromkreis)**.** Wir wollen den Widerstand R in einem Stromkreis durch eine Reihe von Messungen von anliegender Spannung U und hindurchfließendem Strom I bestimmen. Das *Ohmsche Gesetz* besagt, dass (zumindest unter idealisierten Bedingungen) der Widerstand konstant ist und als Quotient von Spannung und Stromstärke gegeben ist, also dass der Zusammenhang

$$U = R \cdot I$$

gilt. Messen wir nun den Strom I_i zu verschiedenen Spannungen U_i für $i \in \{1, \dots, m\}$, so würden wir ohne Messfehler erwarten, dass die Messungen alle auf einer Geraden liegen. In der Realität ist dies jedoch typischerweise nicht der Fall und wir müssen das lineare Ausgleichsproblem lösen, bei dem R nun so bestimmt wird, dass die Fehlerquadrate minimiert werden. Dazu minimiert man die Abbildung

$$R \mapsto \sum_{i=1}^{m} (R \cdot I_i - U_i)^2$$

was in der obigen Formulierung (2.8) dem Fall $n = 1$ mit den Wahlen $A = (I_1, \dots, I_m)^T$ und $b = (U_1, \dots, U_m)^T$ entspricht.



Das erste Ordnungskriterium für Minimalität zeigt, dass die Lösung R in diesem Fall die Gleichung

$$0 \overset{!}{=} \frac{d}{dR} \sum_{i=1}^{m} (R \cdot I_i - U_i)^2 = 2 \sum_{i=1}^{m} (R \cdot I_i - U_i) \cdot I_i$$

erfüllen muss, was zeigt, dass R gegeben ist durch

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

Wir überlegen uns nun äquivalente Bedingungen dafür, dass ein Vektor eine Lösung mit kleinsten Fehlerquadraten ist, zeigen deren Existenz und (unter einer Zusatzvoraussetzung an A) auch deren Eindeutigkeit.

Satz 2.55 (Existenz und Eindeutigkeit der Lösung des linearen Ausgleichsproblems)**.** *Gegeben sei eine Matrix $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und ein Vektor $b \in \mathbb{R}^m$. Dann sind die folgenden Aussagen für $x \in \mathbb{R}^n$ äquivalent:*

(i) *x ist eine Lösung des linearen Ausgleichsproblems* (2.8)*, also*

$$\|Ax - b\|_2 = \min\big\{\|Ay - b\|_2 \colon y \in \mathbb{R}^n\big\}.$$

(ii) *Es gilt $\langle\, Ax - b, Ay \,\rangle_2 = 0$ für alle $y \in \mathbb{R}^n$.*

(iii) *x löst die* Normalengleichung

$$A^T Ax = A^T b.$$

Insbesondere existiert die Lösung des linearen Ausgleichsproblems (2.8)*. Hat die Matrix außerdem vollen Rang, also gilt* $\operatorname{Rang}(A) = n$*, so ist diese Lösung auch eindeutig.*

Beweis. Implikation (i) $\Rightarrow$ (ii)*:* Löst $x \in \mathbb{R}^n$ das lineare Ausgleichsproblem und ist $y \in \mathbb{R}^n$ beliebig, so hat die Abbildung $\mathbb{R} \ni t \mapsto \|A(x+ty)-b\|_2^2$ bei $t = 0$ ein Minimum. Aus dem ersten Ordnungskriterium für Minimalität folgt damit

$$0 = \frac{d}{dt}\Big|_{t=0} \|A(x + ty) - b\|_2^2 = 2\langle\, A(x + ty) - b, Ay \,\rangle_2\Big|_{t=0} = 2\langle\, Ax - b, Ay \,\rangle_2.$$

Wegen der Beliebigkeit von $y \in \mathbb{R}^n$ ist damit (ii) gezeigt.

Implikation (ii) $\Rightarrow$ (i)*:* Für $x \in \mathbb{R}^n$ sei nun umgekehrt $\langle\, Ax - b, Ay \,\rangle_2 = 0$ für alle $y \in \mathbb{R}^n$ erfüllt. Für beliebiges $y \in \mathbb{R}^n$ bemerken wir damit

$$\|A(x + y) - b\|_2^2 = \|Ax - b\|_2^2 + 2\langle\, Ax - b, Ay \,\rangle_2 + \|Ay\|_2^2 = \|Ax - b\|_2^2 + \|Ay\|_2^2.$$

Da der zweite Term auf der rechten Seite immer nicht-negativ ist, löst also x wie behauptet das lineare Ausgleichsproblem.

Äquivalenz (ii) $\Leftrightarrow$ (iii)*:* Wir beobachten hier

$$\begin{aligned} 0 = \langle\, Ax - b, Ay \,\rangle_2 = (Ay)^T(Ax - b) = y^T(A^T Ax - A^T b) \quad &\text{für alle } y \in \mathbb{R}^n \\ \Leftrightarrow \quad A^T Ax - A^T b = 0 & \\ \Leftrightarrow \quad A^T Ax = A^T b. & \end{aligned}$$

Existenz der Lösung: Die Funktion $y \mapsto \|Ay - b\|_2$ ist stetig, konvex und offensichtlich auch nach unten beschränkt. Dadurch existiert immer ein Minimum.

Eindeutigkeit der Lösung: Ist nun A zusätzlich von vollem Rang, so ist die Matrix $A^T A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit, also insbesondere invertierbar. Damit besitzt die Normalengleichung $A^T Ax = A^T b$ die eindeutige Lösung $x = (A^T A)^{-1} b$, die wegen der zuvor gezeigten Äquivalenzen zugleich die einzige Lösung des linearen Ausgleichsproblems ist. □

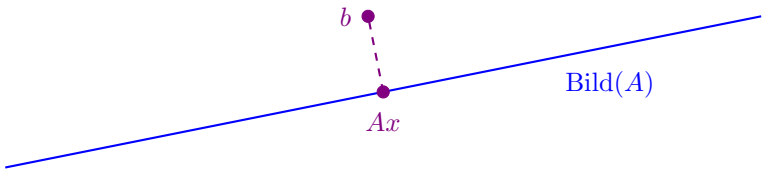
Bemerkungen 2.56.

(1) *Die Aussage von Satz* 2.55 (ii) *bedeutet, dass Ax die orthogonale Projektion von b auf* $\operatorname{Bild}(A)$ *ist. Daher kann man das lineare Ausgleichsproblem* $\min\{\|b - Ay\|_2 \colon y \in \mathbb{R}^n\}$ *auch als ein Projektionsproblem verstehen:*

$$\|Ax - b\|_2 = \min\big\{\|Ay - b\|_2 \colon y \in \mathbb{R}^n\big\} \quad \Leftrightarrow \quad Ax \text{ ist die orthogonale Projektion von } b \text{ auf } \operatorname{Bild}(A).$$

(2) *Wenn man die* Kondition des linearen Ausgleichsproblems *untersucht, so stellt man fest, dass Störungen von b zu umso stärkeren relativen Störungen der Lösung x führen, je größer der Winkel zwischen b und* Bild(A) *ist.*

(3) *Der Name* Normalengleichung *stammt aus der geometrischen Interpretation, nämlich dass der Vektor $Ax - b$ für die Lösung x des linearen Ausgleichsproblems gerade normal zu* Bild(A) *ist.*



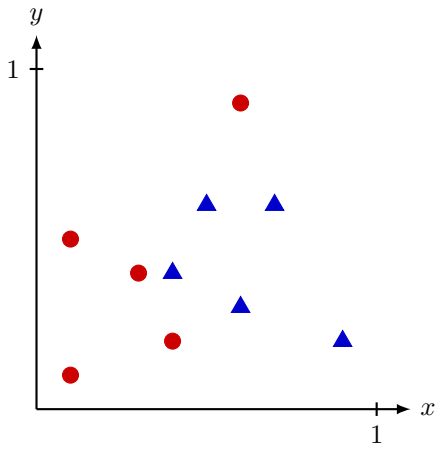
(4) *Die Normalengleichung ist zwar mit den zuvor vorgestellten Methoden lösbar, jedoch im Allgemeinen schlecht konditioniert (und der Algorithmus damit* instabil*). Mit der QR-*Zerlegung *der Matrix A gibt es jedoch auch Verfahren, die die Normalengleichung und ihre schlechte Konditionierung vermeiden. Diese Zerlegung existiert für den Fall, dass die Matrix A von vollem Rang ist, immer und kann explizit mittels Drehungen und Spiegelungen konstruiert werden.*

Kapitel 3

Datenklassifizierung über neuronale Netze

Deep Learning wird in vielen Bereichen eingesetzt, in denen große Datenmengen und komplexe Musterverarbeitung eine Rolle spielen. Einige weit verbreitete und inzwischen klassische Anwendungen sind aus der Bilderkennung (Gesichtserkennung, automatische Kennzeichenerkennung, medizinische Bildanalyse), Spracherkennung (Sprachassistenten), computergestützte Sprachverarbeitung (inklusive Chatbots) und autonomes Fahren. Anhand eines einfachen Beispiels der Klassifizierung von Daten werden wir hier kurz vorstellen, wie ein neuronales Netzwerk aufgebaut und anhand von bekannten Daten trainiert wird, um schließlich neue Daten zuordnen zu können.

Beispiel 3.1 (Punktklassifizierung in 2D)**.** Wir betrachten eine Menge aus Punkten im Einheitsquadrat $[0,1]^2 \subset \mathbb{R}^2$, die jeweils einer von zwei Klassen (● oder ▲) zugeordnet sind. Die beiden Achsen entsprechen dabei zwei (normierten) Werten einer messbaren Größe (etwa der Position oder der Ausdehnung in eine bestimmte Orientierung, dem Gewicht, der Dichte, der Wellenlänge im Farbspektrum, etc.). So könnte das Bild rechts etwa für Ölbohrungen stehen, bei denen ▲ eine erfolgreiche und ● eine erfolglose Bohrung markiert). Auf der Basis dieser vorgegebenen Daten suchen wir nun eine Kategorisierung, die jedem Punkt aus $[0,1]^2$ eine der beiden Klassen zuordnet.



3.1 Lösung über Ausgleichsrechnung

Wir versuchen das oben in Beispiel 3.1 beschriebene Problem der Punktklassifizierung zunächst über die in Kapitel 2.5 vorgestellte lineare Ausgleichsrechnung zu lösen. Über die bekannten Datenpunkte haben wir für m Parameter aus $[0,1]^2$ Werte $b_1, \ldots, b_m \in \{\pm 1\}$. Die Klassifizierung der Punkte soll nun über eine Funktion b in Abhängigkeit von der Variable $t = (t_1, t_2) \in \mathbb{R}^2$ modelliert werden, bei der $b(t) = 1$ für die Klasse ▲ und $b(t) = -1$ für die Klasse ● steht. Um das lineare Ausgleichsproblem zu formulieren, geben wir uns nun Ansatzfunktionen $a_1, \ldots, a_n$ (wieder in Abhängigkeit von $t \in [0,1]^2$) vor, wobei $n \leq m$ gelten sollte. Unser Modell für die Funktion b ist dann

$$b(t) = a_1(t)x_1 + \ldots + a_n(t)x_n$$

mit zu bestimmenden Koeffizienten $x_1, \ldots, x_n \in \mathbb{R}$. Sinnvolle Vorhersagen für unbekannte Punkte sind nur dann zu erwarten, wenn die Ansatzfunktionen gut gewählt sind. Folglich muss bei diesem Vorgehen ein gewisses Vorwissen über die Problemstellung von Anfang an mit eingebracht werden.

Beispiele 3.2. Wir überlegen uns nun ein paar Möglichkeiten für die Ansatzfunktionen.

(i) Erwarten wir eine affinlineare Abhängigkeit, so könnten wir mit einer konstanten und den linearen Ansatzfunktionen

$$a_1(t) \coloneqq 1, \quad a_2(t) \coloneqq t_1, \quad a_3(t) \coloneqq t_2$$

arbeiten. Dies führt auf eine Normalengleichung der Dimension 3.

(ii) Zu den Funktionen aus (i) nehmen wir nun noch eine vierte, quadratische Ansatzfunktion

$$a_4(t) \coloneqq t_1^2 + t_2^2$$

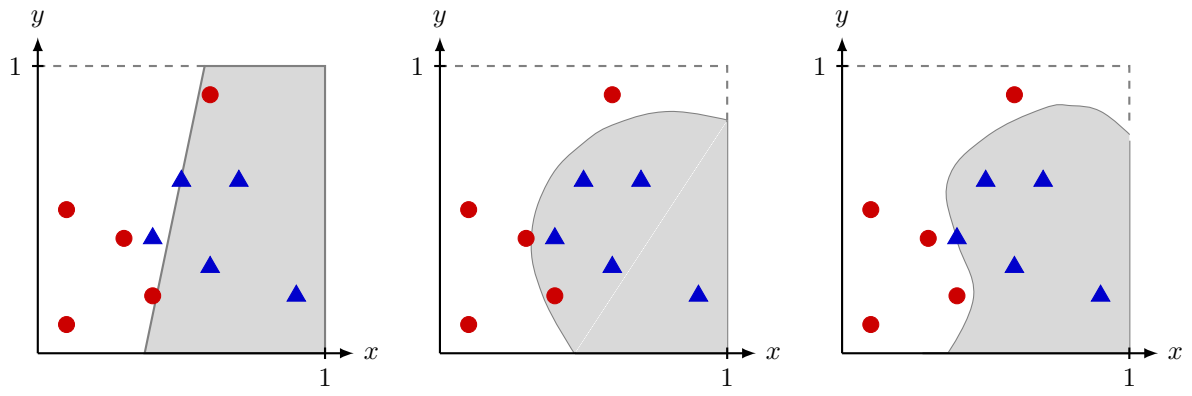
hinzu. Dann ist eine Normalengleichung der Dimension 4 zu lösen.

(iii) Schließlich betrachten wir als zusätzliche Ansatzfunktionen noch die trigonometrischen Funktionen

$$a_5(t) \coloneqq \sin(t_1), \quad a_6(t) \coloneqq \sin(t_2).$$

In diesem Fall ist die Normalengleichung entsprechend von der Dimension 6.

Die Ergebnisse für diese drei Ansätze sind in den Bildern unten illustriert. In diesen Fällen ist b eine stetige Funktion, so dass man den grauen Bereich $b > 0$ der Klasse ▲ und den weißen Bereich $b < 0$ der Klasse ● zuordnen würde. Wir erkennen dabei deutlich, dass die Lösung des Ausgleichsproblems, also die Klassifizierung, ganz entscheidend von der Wahl und der Anzahl der Ansatzfunktionen abhängt und entsprechend sehr unterschiedlich aussehen kann.

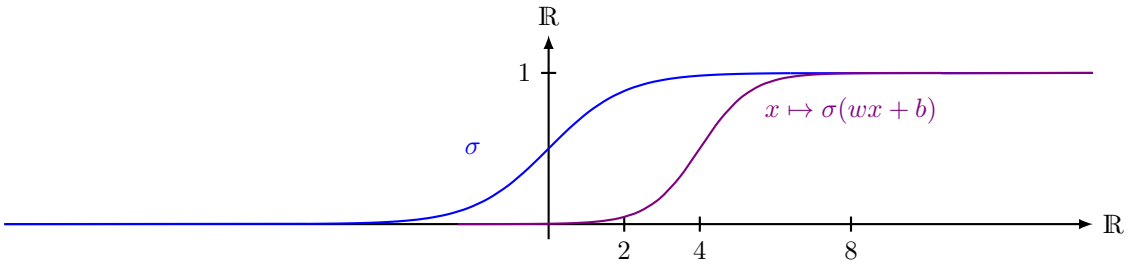


3.2 Lösung über neuronale Netze

Wir stellen nun einen anderen Lösungsansatz vor, der auf neuronalen Netzen basiert. Zunächst führen wir dazu eine *Aktivierungsfunktion* ein, die entscheidet, ob ein Neuron aktiviert wird oder nicht. Ein Beispiel für eine solche Aktivierungsfunktion ist die spezielle Sigmoid-Funktion $\sigma\colon \mathbb{R} \to \mathbb{R}$ mit

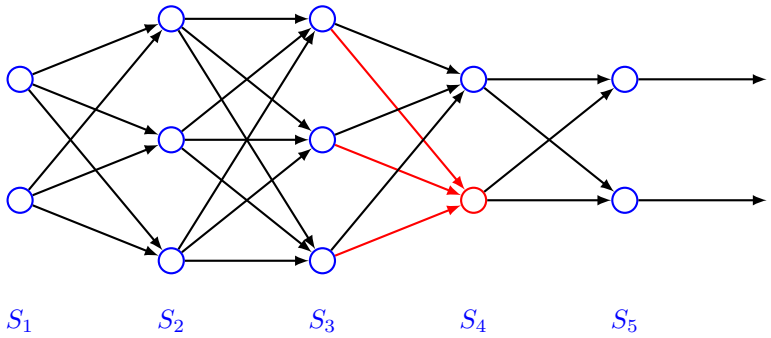
$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad \text{für } x \in \mathbb{R},$$

die eine geglättete Form der Stufenfunktion $\mathbb{1}_{\mathbb{R}^+}$ darstellt und nur Werte zwischen 0 und 1 annimmt. Über diese Sigmoid-Funktion werden die wesentlichen funktionellen Eigenschaften eines Neurons abgebildet, nämlich dass ein Neuron erst bei hinreichend großem Reiz aktiviert werden, dass dabei die Beziehung zwischen dem eingehenden Signal und der Aktivierung des Neurons nichtlinear ist und dass sich die Reaktion bei sehr niedrigen oder sehr hohen Reizen kaum ändert. Durch Skalieren (mit einem sogenannten Gewicht) und Verschieben (über einen sogenannten Bias) des Arguments der Sigmoid-Funktion können wir den Wechsel von inaktiv zu aktiv dabei schneller oder langsamer machen und verschieben.



Im Folgenden werden wir die Sigmoid-Funktion auch auf Vektoren anwenden und dies komponentenweise verstehen, also $\sigma\colon \mathbb{R}^m \to \mathbb{R}^m$ über $\sigma(z)_j := \sigma(z_j)$ für $j \in \{1,\dots,m\}$ definieren, wobei rechts in der Definition nun die zuvor auf $\mathbb{R}$ definierte Sigmoid-Funktion gemeint ist.

Neuronale Netzwerke bestehen nun aus mehreren Schichten (in untenstehendem Bild etwa aus fünf Schichten $S_1,\dots,S_5$), die jeweils wiederum aus mehreren Neuronen oder Knoten bestehen.



In jeder Schicht gibt jedes Neuron dabei eine skalare Zahl an alle Neuronen der nächsten Schicht aus. Dort erzeugt jedes Neuron seinen eigenen gewichteten Wert, versieht ihn mit seinem Bias, und wendet darauf eine Aktivierungsfunktion an, um die eigene Ausgabe an die nächste Schicht zu bestimmen. Wir beschreiben dies nun mathematisch und verwenden dabei der Einfachheit halber für jede Schicht die gleiche Aktivierungsfunktion (tatsächlich kann aber für jedes Neuron eine eigene Aktivierungsfunktion gewählt werden). Mathematisch wird die Ausgabe der ℓ-ten Schicht also über

$$\sigma\big(W^{[\ell]}x + b^{[\ell]}\big)$$

berechnet, wobei $x \in \mathbb{R}^{n_{\ell-1}}$ der Vektor der Ausgaben der $(\ell-1)$-ten Schicht ist, die aus $n_{\ell-1} \in \mathbb{N}$ Neuronen besteht, $W^{[\ell]} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ eine Matrix mit Gewichten ist und $b^{[\ell]} \in \mathbb{R}^{n_\ell}$ den Bias enthält. Um etwa die Ausgabe des rot markierten Knotens im vorherigen Bild des neuronalen Netzwerkes zu berechnen, benötigt man die drei Ausgaben der Knoten aus der dritten Schicht, die drei Gewichte aus der zweiten Zeile der Matrix $W^{[4]} \in \mathbb{R}^{2\times 3}$ sowie den zweiten Eintrag von $b^{[4]} \in \mathbb{R}^2$ als Bias.

In einem vorgegebenen neuronalen Netzwerk mit L Schichten wird die finale Ausgabe aus der initialen Eingabe über eine Funktion $F(=F_{W,b})\colon \mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$ berechnet, indem die Ausgaben von Schicht zu Schicht wie eben beschrieben weitergegeben und dort verarbeitet werden, also indem $F(x) = a^{[L]}$ für $x \in \mathbb{R}^{[n_1]}$ rekursiv ermittelt wird über

$$a^{[1]} := x \quad \text{und} \quad a^{[\ell]} := \sigma\big(W^{[\ell]}a^{[\ell-1]} + b^{[\ell]}\big) \quad \text{für } \ell \in \{2,\dots,L\}.$$

Damit hängt die finale Ausgabe von allen Gewichten und Biases ab. Im vorherigen Netzwerk etwa gibt es 25 Gewichte und 10 Biases, also hängt die Ausgabe dort insgesamt von 35 Parametern ab. Diese Abhängigkeit ist wegen der Nichtlinearität der Sigmoid-Funktion nichtlinear (und man kann sich auch leicht überlegen, dass lineare Aktivierungsfunktionen für die Lösung nichtlinearer Aufgaben nicht geeignet wären, da hintereinander ausgeführte lineare Funktionen linear bleiben).

Beispiel 3.3 (Punktklassifizierung in 2D)**.** In Beispiel 3.1 sind die bekannten Daten durch $m = 10$ Punkte $x^{(1)}, \dots, x^{(10)} \in [0,1]^2$ gegeben mit $y^{(i)} = (1,0)^T$, falls $x^{(i)}$ der Klasse ▲ angehört, und $y^{(i)} = (0,1)^T$, falls $x^{(i)}$ der Klasse ● angehört, für $i \in \{1, \dots, 10\}$. Die Zuordnung unbekannter Punkte $x \in [0,1]^2$ würde dann durch Auswertung der Funktion $F\colon [0,1]^2 \to \mathbb{R}^2$ erfolgen, und zwar zur Klasse ▲ im Fall $F_1(x) > F_2(x)$ und zur Klasse ● im Fall $F_1(x) < F_2(x)$ (sowie einer Regel für den Fall der Gleichheit beider Komponentenfunktionen).

Um zu beurteilen, wie gut die Vorhersage des neuronalen Netzes für gegebene Daten ist, verwenden wir wieder eine Kosten- oder Lossfunktion. Haben wir etwa m Datenpunkte, jeweils bestehend aus einem Input $x^{(i)}$ und einer Ausgabe $y^{(i)}$ für $i \in \{1, \dots, m\}$, so berechnet die Kostenfunktion die Abweichung der durch das vorgegebene neuronale Netz berechneten Ausgaben $\hat{y}^{(i)} = F(x^{(i)})$ von der echten Ausgabe $y^{(i)}$ für $i \in \{1, \dots, m\}$, etwa mit (typischerweise über die Trainingsdaten gemittelten) Fehlerquadraten, also

$$\operatorname{Cost}\left(W^{[2]}, \dots, W^{[L]}, b^{[2]}, \dots, b^{[L]}\right) := \frac{1}{m} \sum_{i=1}^{m} \|y^{(i)} - F(x^{(i)})\|_2^2 =: \frac{1}{m} \sum_{i=1}^{m} C_{x^{(i)}}.$$

Haben wir ein Modell zu bestimmten Inputs und Ausgaben gegeben, so wollen wir die Werte der Gewichte und Biase so wählen, dass die Kostenfunktion minimal wird. Dies bezeichnet man auch als *Training des Netzwerks*. Für eine vereinfachte Notation fassen wir nun alle zu optimierenden Parameter zu einem Vektor $p \in \mathbb{R}^s$ zusammen und betrachten daher als Kostenfunktion $\operatorname{Cost}\colon \mathbb{R}^s \to \mathbb{R}$. Im Allgemeinen ist diese Kostenfunktion zwar nicht-konvex und die Existenz von Minimierern keineswegs garantiert, jedoch gibt es für das Training des Netzwerks verschiedene Möglichkeiten.

Verfahren des steilsten Abstiegs. Für das Training des Netzwerks können wir ein Gradientenverfahren wie in Kapitel 2.4 verwenden und versuchen, eine Folge von Vektoren $\{p^{(k)}\}_{k \in \mathbb{N}}$ zu finden, die (zumindest lokal) gegen einen Minimierer der Kostenfunktion konvergiert. Ausgehend von einer Iterierten $p^{(k)}$ suchen wir dabei jeweils nach einem Differenzvektor $\Delta p^{(k)}$, so dass die neue Iterierte $p^{(k+1)} := p^{(k)} + \Delta p^{(k)}$ geringere Kosten hat. Verwenden wir die erste Ordnungsentwicklung der Kostenfunktion

$$\operatorname{Cost}(p + \Delta p) \approx \operatorname{Cost}(p) + \nabla \operatorname{Cost}(p)^T \Delta p = \operatorname{Cost}(p) + \sum_{r=1}^{s} \frac{\partial \operatorname{Cost}(p)}{\partial p_r} \Delta p_r,$$

so sollte man den Differenzvektor Δp in Richtung des negativen Gradienten $-\nabla \operatorname{Cost}(p)$ wählen. Da die erste Ordnungsentwicklung jedoch nur für Vektoren Δp kleiner Länge eine gute Approximation darstellt, setzt man

$$\Delta p^{(k)} = -\eta \nabla \operatorname{Cost}(p^{(k)})$$

und damit

$$p^{(k+1)} = p^{(k)} - \eta \nabla \operatorname{Cost}(p^{(k)})$$

für einen kleinen Parameter $\eta > 0$, den man auch als *Lernrate* bezeichnet. Dieses Vorgehen wird nun wiederholt, bis ein Abbruchkriterium erfüllt oder eine vorgegebenen Anzahl an Schritten durchgeführt ist, und die finale Iterierte wird dann als Parameter im neuronalen Netzwerk verwendet. Das Problem bei diesem Vorgehen ist jedoch die üblicherweise große Datenmenge. Bei einer großen Anzahl an zu optimierenden Parameter oder einer großen Anzahl an vorgegebenen Datenpunkten ist die Berechnung des Gradienten in jedem Iterationsschritt sehr aufwändig, so dass andere Methoden für das Training des Netzwerks verwendet werden sollten.

Stochastisches Gradientenverfahren. Eine Alternative zum "vollen" Verfahren des steilsten Abstiegs ist es, den Gradienten über alle Trainingspunkte durch den Gradienten nur eines einzigen, zufällig gewählten Trainingspunktes zu ersetzen. Wir wählen dazu in jedem Iterationsschritt also zufällig einen Index $i_k \in \{1, \dots, m\}$ und verwenden dann als neue Iterierte

$$p^{(k+1)} = p^{(k)} - \eta \nabla C_{x^{(i_k)}}(p^{(k)}).$$

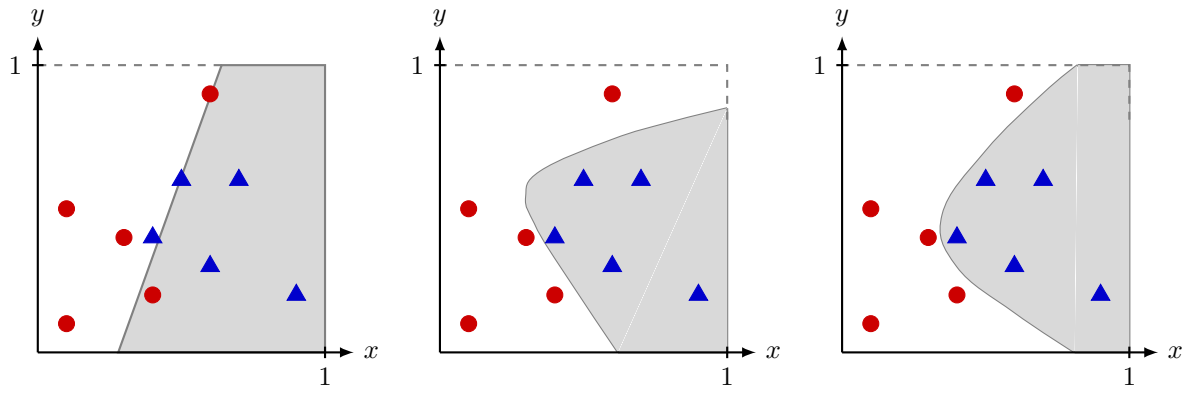
Alternativ kann die zufällige Auswahl auch "ohne Zurücklegen" durchgeführt werden. Zu beachten ist in jedem Fall jedoch, dass die Kosten für die neue Iterierte selbst für sehr kleine Lernraten nicht geringer zu sein brauchen als für die alte Iterierte, da eben nur der Gradient für einen Trainingspunkt und nicht wie zuvor für alle Trainingspunkte berücksichtigt wurde. In der Praxis wird daher meist ein Kompromiss verwendet und mehrere Indizes aus $\{1, \dots, m\}$ ausgewählt (ein sogenannter *Minibatch*), für die dann der Gradient und entsprechend die neue Iterierte gebildet wird. Ein solcher Schritt ist im Vergleich zum vollem Gradienten noch relativ billig, die neue Iterierte in der Regel im Vergleich zur Wahl nur eines Index aber deutlich besser.

Bemerkungen 3.4.

(1) *Die Größe der Lernrate ist entscheidend für die Konvergenz des Verfahrens und deren Geschwindigkeit. Da oft zu wenig über die Daten bekannt ist, kann die Lernrate auch während des Verfahrens adaptiv angepasst werden.*

(2) *In der Praxis ist oft nicht entscheidend, dass bekannte Daten möglichst gut repräsentiert werden, sondern dass unbekannte Daten möglichst gut vorhergesagt werden. Daher werden häufig bekannte Daten in Trainingsdaten für die Optimierung der Parameter und in Testdaten für die Validierung des Modells, also für die Überprüfung der Vorhersage für unbekannte Daten, unterteilt.*

Beispiel 3.5 (Punktklassifizierung in 2D)**.** Wir betrachten erneut die Punktklassifizierung aus Beispiel 3.1, mit der Zuordnung unbekannter Punkte wie in Beispiel 3.3. Zur Lösung dieses Problems verwenden wir nun ein neuronales Netz mit drei Schichten, die jeweils zwei Neuronen enthalten. Die Gewichte werden zufällig gemäß einer Standardnormalverteilung initialisiert und die Bias-Werte auf Null gesetzt. Für eine feste Anzahl $N = 2 \cdot 10^5$ an Iterationen des stochastischen Gradientenverfahrens zur Bestimmung der Netzwerkparameter sind unten Ergebnisse für verschiedene Lernraten $\eta = 0.01$, $\eta = 0.05$ und $\eta = 0.5$ dargestellt.



Kapitel 4

Nichtlineare Gleichungssysteme

Wir beschäftigen uns nun mit der Lösung *nichtlinearer* Gleichungssysteme und wollen für eine nicht notwendigerweise lineare Funktion $F\colon D \subset \mathbb{R}^n \to \mathbb{R}^n$ eine Nullstelle finden, d.h. ein $x^* \in D$ mit der Eigenschaft

$$F(x^*) = 0. \tag{4.1}$$

Ohne Einschränkung haben wir hier wie in Kapitel 2 wieder angenommen, dass die Anzahl der unabhängigen Variablen gleich der Anzahl der Gleichungen ist (ansonsten fügt man bei mehr Gleichungen als Variablen "Phantomvariablen" hinzu, von denen die Funktion gar nicht abhängt, oder man fügt bei mehr Variablen als Gleichungen einige "0-Phantomgleichungen" hinzu, die aus den anderen Gleichungen folgen oder immer erfüllt sind).

4.1 Banachscher Fixpunktsatz und Fixpunktiteration

Wir interessieren uns in diesem Abschnitt für Fixpunkte von Abbildungen auf $\mathbb{R}^n$ sowie für Kriterien für deren Existenz.

Definition 4.1 (Fixpunkt)**.** *Es sei X eine beliebige Menge und $\Phi\colon X \to X$ eine Abbildung. Wir nennen einen Punkt $x^* \in X$ einen* Fixpunkt *von Φ, falls $\Phi(x^*) = x^*$ gilt.*

Bemerkungen 4.2.

(1) *Anschaulich sind Fixpunkte gerade die Schnittpunkte des Graphen von Φ mit dem Graphen von $\mathbb{1}_X\colon X \to X$, definiert über $\mathbb{1}_X(x) = x$ für $x \in X$.*

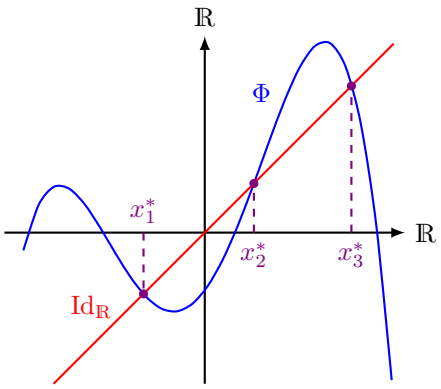
(2) *Das Nullstellenproblem (4.1) kann äquivalent als Fixpunktproblem formuliert werden. Definiert man etwa $\Phi_\lambda\colon D \subset \mathbb{R}^n \to \mathbb{R}^n$ über*

$$\Phi_\lambda(x) \coloneqq \lambda F(x) + x \quad \text{für } x \in D$$

mit beliebigem $\lambda \in \mathbb{R} \setminus \{0\}$, so haben wir

$$F(x^*) = 0 \quad \Leftrightarrow \quad \lambda F(x^*) = 0 \quad \Leftrightarrow \quad \Phi_\lambda(x^*) = x^*.$$

Also ist ein Punkt $x^* \in D$ genau dann eine Nullstelle von F, falls er ein Fixpunkt von Φ_λ ist.

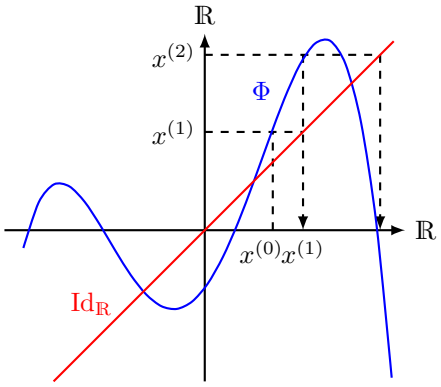


Im Folgenden nehmen wir an, dass die gegebene Abbildung (mindestens) einen Fixpunkt besitzt. Im Allgemeinen findet man einen solchen (oder eine Nullstelle) auch bei exakter Arithmetik nicht durch endlich viele Operationen, sondern nur über geeignete Iterationsverfahren. Eine naheliegende Iterationsvorschrift wird gerade über die Abbildung Φ definiert:

Definition 4.3 (Fixpunktiteration)**.** *Es sei X eine beliebige Menge und $\Phi\colon X \to X$ eine Abbildung. Ist $x^{(0)} \in X$ ein beliebiger Startvektor, so nennen wir die Folge $\{x^{(k)}\}_{k\in\mathbb{N}}$, die durch die Vorschrift*

$$x^{(k+1)} := \Phi(x^{(k)}) \quad \text{für } k \in \mathbb{N}_0$$

definiert wird, die zu $x^{(0)}$ gehörige Folge der Fixpunktierterten *oder* Fixpunktiteration*.*



Bemerkung 4.4. *Die Vorschrift der Fixpunktiteration bedeutet gerade, dass man für die neue Iterierte die Funktionsvorschrift mit der alten Iterierten im Argument auswerten muss.*

Ist nun eine Abbildung gegeben, so gibt es einige natürliche Fragestellungen bezüglich Fixpunkten:

(1) Existiert ein Fixpunkt und ist er eindeutig?

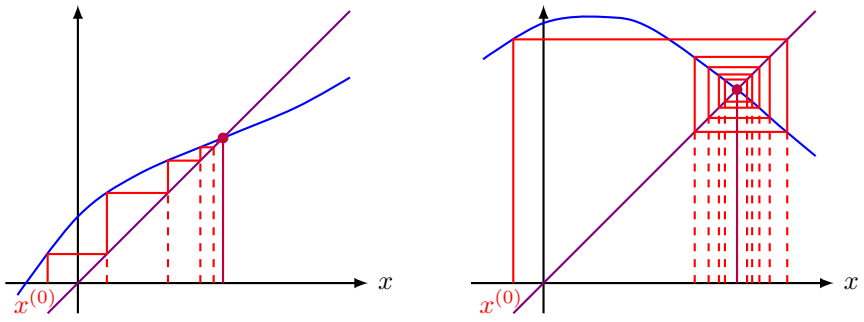
(2) Für welche Startwerte $x^{(0)}$ konvergiert die Fixpunktiteration gegen einen Fixpunkt?

(3) Wie schnell konvergiert die Fixpunktiteration in einem solchen Fall?

Beispiel 4.5. Schon für Funktionen $\Phi\colon \mathbb{R} \to \mathbb{R}$ muss die Fixpunktiteration nicht für beliebige Startwerte konvergieren, selbst wenn es einen Fixpunkt gibt: betrachtet man $\Phi(x) := x^2$ für $x \in \mathbb{R}$, so gibt es genau zwei Fixpunkte $x_1^* = 0$ und $x_2^* = 1$. Für die Fixpunktiteration ergibt sich $x^{(k)} = x_0^{2^k}$ für alle $k \in \mathbb{N}_0$, und damit sieht man:

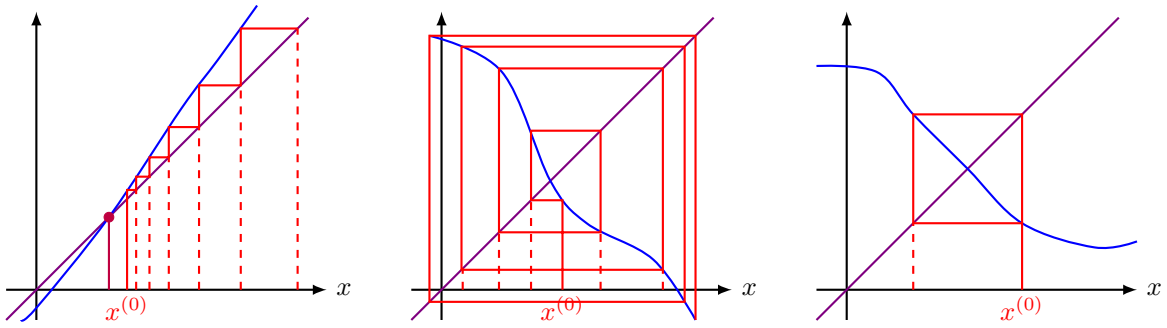
- für alle Startwerte $x^{(0)} \in (-1, 1)$ konvergiert die zugehörige Fixpunktiteration gegen x_1^*,
- für den Startwert $x^{(0)} = \pm 1$ konvergiert sie gegen x_2^* (und ist ab spätestens dem zweiten Folgenglied sogar konstant),
- für alle Startwerte $x^{(0)} \in \mathbb{R} \setminus [-1, 1]$ divergiert sie.

Für Funktionen, die (mindestens) einen Fixpunkt besitzen, kann also sowohl Konvergenz als auch Divergenz für Fixpunktiterationen vorliegen. Wir wollen nun anhand von Beispielen motivieren, dass das Wachstumsverhalten der Funktion (oder deren Ableitung im Falle von Differenzierbarkeit) in der Nähe eines Fixpunktes eine entscheidende Rolle spielt. Dazu betrachten wir zunächst *anziehende* Fixpunkte, bei denen Konvergenz der Fixpunktiteration vorliegt, wenn der Startpunkt hinreichend nahe am Fixpunkt gewählt ist.



Im linken Bild sind die Werte der Funktion (in der Nähe des Fixpunktes x^*) immer zwischen der Identitätsabbildung $\mathbb{1}_{\mathbb{R}}$ und dem Fixpunkt x^* selbst, was dazu führt, dass die Fixpunktiteration monoton gegen den Fixpunkt konvergiert. Im rechten Bild haben wir eine "gespiegelte" Situation, d.h. die Funktion $x \mapsto x^* - \Phi(x)$ zeigt ein solches Verhalten, so dass die Fixpunktiteration alternierend gegen den Fixpunkt konvergiert.

Wir betrachten nun einige Situationen für Divergenz von Fixpunktiteration.



In den beiden linken Bilder handelt es sich um *abstoßende* Fixpunkte, bei denen Divergenz der Fixpunktiteration vorliegt, unabhängig davon, wie nahe der Startpunkt $x^{(0)} \neq x^*$ beim Fixpunkt x^* gewählt ist. Hierbei ist die Divergenz im ersten Fall monoton und im zweiten Fall alternierend. Der letzte Fall ist etwas spezieller, dort liegt nämlich ein zyklisches Verhalten vor.

Um ein Kriterium für die Existenz eines Fixpunktes und die Konvergenz der Fixpunktiteration herzuleiten, ist es offensichtlich nützlich, wenn der Abstand zweier aufeinander folgender Folgenglieder quantifiziert kleiner wird, in folgendem Sinne:

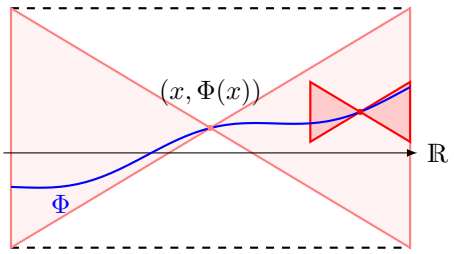
Definition 4.6 (Kontraktion)**.** *Es sei $D \subset \mathbb{R}^n$ eine beliebige Menge. Eine Abbildung $\Phi\colon D \to \mathbb{R}^n$ heißt* Kontraktion *oder* kontrahierend*, falls eine Konstante $\kappa \in [0,1)$ existiert, so dass*

$$|\Phi(x) - \Phi(\tilde{x})| \leq \kappa |x - \tilde{x}| \qquad \text{für alle } x, \tilde{x} \in D \text{ gilt.}$$

Die Konstante κ nennt man dabei auch Kontraktionszahl *von Φ auf D.*

Bemerkungen 4.7.

(1) *Im skalaren Fall $n = 1$ bedeutet die Kontraktionseigenschaft einer Funktion $\Phi\colon \mathbb{R} \to \mathbb{R}$ gerade, dass für jedes $x \in \mathbb{R}$ alle Punkte $(\tilde{x}, \Phi(\tilde{x}))$ in dem von $(x, \Phi(x))$ ausgehenden Doppelkegel mit Steigung $\pm\kappa$ liegen (und entsprechend auch in höheren Dimensionen in einem höherdimensionalen Kegel).*



(2) *Jede Kontraktion ist insbesondere stetig.*

(3) *Ist $D \subset \mathbb{R}^n$ eine offene, konvexe Menge und $\Phi\colon D \to \mathbb{R}^n$ eine stetig differenzierbare Funktion, so kann man zeigen, dass Φ genau dann eine Kontraktion ist, wenn $\sup_{x \in D} \|D\Phi(x)\| < 1$ gilt.*

Die eindeutige Existenz eines Fixpunktes wird unter geeigneten Annahmen über den Banachschen Fixpunktsatz sichergestellt:

Satz 4.8 (Banachscher Fixpunktsatz)**.** *Es sei $D \subset \mathbb{R}^n$ eine abgeschlossene Teilmenge. Ist $\Phi\colon D \to \mathbb{R}^n$ eine* Selbstabbildung*, also $\Phi(D) \subset D$, und* kontrahierend *mit Kontraktionszahl $\kappa \in [0,1)$, so gelten:*

(i) *Φ besitzt* genau einen Fixpunkt *$x^* \in D$;*

(ii) *Ist $x^{(0)} \in D$ ein beliebiger Startwert, so konvergiert die Fixpunktiteration $\{x^{(k)}\}_{k \in \mathbb{N}}$ aus Definition 4.3 gegen den Fixpunkt x^* von Φ auf D, und für $k \in \mathbb{N}$ gelten die* Fehlerabschätzungen

$$|x^{(k)} - x^*| \leq \frac{\kappa}{1-\kappa} |x^{(k-1)} - x^{(k)}| \quad (\textit{a posteriori Abschätzung}),$$

$$|x^{(k)} - x^*| \leq \frac{\kappa^k}{1-\kappa} |x^{(0)} - x^{(1)}| \quad (\textit{a priori Abschätzung}).$$

★ *Beweis.* Der Beweis ist konstruktiv und startet mit einem beliebigen $x^{(0)} \in D$ sowie der oben definierten Folge $\{x^{(k)}\}_{k\in\mathbb{N}}$ der Fixpunktiterierten. Nach Definition dieser Folge und wegen der Kontraktionseigenschaft von Φ gilt für den Abstand zweier aufeinander folgender Folgenglieder

$$|x^{(k)} - x^{(k+1)}| = |\Phi((x^{(k-1)}) - \Phi(x^{(k)})| \leq \kappa |x^{(k-1)} - x^{(k)}|.$$

Induktiv folgt daher

$$|x^{(j)} - x^{(j+1)}| \leq \kappa^{j-k+1} |x^{(k-1)} - x^{(k)}|$$

für $j \geq k$. Mit der Dreiecksungleichung und der geometrischen Reihe erhalten wir damit

$$\begin{aligned} |x^{(k)} - x^{(\ell)}| &\leq \sum_{j=k}^{\ell-1} |x^{(j)} - x^{(j+1)}| \\ &\leq \sum_{j=k}^{\ell-1} \kappa^{j-k+1} |x^{(k-1)} - x^{(k)}| \leq \frac{\kappa}{1-\kappa} |x^{(k-1)} - x^{(k)}| \leq \frac{\kappa^k}{1-\kappa} |x^{(0)} - x^{(1)}| \end{aligned} \tag{4.2}$$

für $\ell \geq k$. Da die rechte Seite wegen $\kappa \in [0,1)$ bei $k \to \infty$ gegen 0 konvergiert, ist $\{x^{(k)}\}_{k\in\mathbb{N}}$ eine Cauchy-Folge in $\mathbb{R}^n$ und konvergiert wegen der Vollständigkeit von $\mathbb{R}^n$ gegen ein $x^* \in \mathbb{R}^n$. Da D abgeschlossen ist und alle Folgenglieder aus D sind, schließt man außerdem $x^* \in D$. Es gelten auch die behaupteten Eigenschaften:

(i) Wegen der Stetigkeit von Φ gilt

$$x^* = \lim_{k\to\infty} x^{(k+1)} = \lim_{k\to\infty} \Phi(x^{(k)}) = \Phi(x^*),$$

also handelt es sich bei x^* tatsächlich um einen Fixpunkt von Φ. Der Fixpunkt ist sogar eindeutig: ist nämlich $x \in \mathbb{R}^n$ ein beliebiger Fixpunkt von Φ, also mit $\Phi(x) = x$, so folgt wegen

$$|x - x^*| = |\Phi(x) - \Phi(x^*)| \leq \kappa |x - x^*|$$

und $\kappa \in [0,1)$ bereits $|x - x^*| = 0$, also $x = x^*$.

(ii) Betrachtet man den Limes $\ell \to \infty$ in der Ungleichung (4.2) (die rechte Seite ist unabhängig von ℓ), so folgen mit

$$\lim_{\ell\to\infty} |x^{(k)} - x^{(\ell)}| = |x^{(k)} - x^*|$$

die a posteriori und die a priori Fehlerabschätzung. □

Bemerkung 4.9. *Mithilfe der* a priori Fehlerabschätzung *kann bereits nach einer Iteration eine Aussage darüber getroffen werden, wie viele Iterationen höchstens notwendig sind, um eine gegebene Genauigkeitsschranke bei der numerischen Approximation des Fixpunktes zu unterschreiten. Die* a posteriori Abschätzung *dagegen ist genauer und kann während der Iteration als Abbruchkriterium verwendet werden.*

Abschließend gehen wir kurz der Frage nach, wie schnell die Konvergenz der Fixpunktiteration ist. Zur Beurteilung dieser Geschwindigkeit wird untersucht, wie groß die Abweichung der neuen Iterierten $x^{(k+1)}$ vom Fixpunkt im Vergleich zur Abweichung der aktuellen Iterierten $x^{(k)}$ ist.

Definition 4.10. *Es sei $(V, \|\cdot\|)$ ein normierter Raum und $\{x^{(k)}\}_{k\in\mathbb{N}_0}$ eine Folge in V, die gegen ein $x \in V$ konvergiert. Wir sagen, die Folge $\{x^{(k)}\}_{k\in\mathbb{N}_0}$*

- konvergiert (mindestens) linear*, falls eine Konstante $C \in [0,1)$ existiert mit*

$$\|x^{(k+1)} - x\| \leq C \|x^{(k)} - x\| \quad \text{für alle } k \geq k_0 \text{ mit einem } k_0 \in \mathbb{N}.$$

- konvergiert von Ordnung $p > 1$, *falls eine Konstante* $C \geq 0$ *existiert mit*

$$\|x^{(k+1)} - x\| \leq C\|x^{(k)} - x\|^p \quad \textit{für alle } k \in \mathbb{N}$$

und p *dabei die größte Zahl ist, für die eine solche Abschätzung gültig ist. Im Fall* $p = 2$ *spricht man auch von* quadratischer Konvergenz.

In den beiden Beispielen der (monotonen und alternierenden) Konvergenz der Fixpunktiteration halbierte sich in jedem Schritt in etwa der Fehler. Dies ist tatsächlich ein typisches Verhalten und Fixpunktiterationen konvergieren üblicherweise nur linear (wenn überhaupt). Nur in besonderen Fällen, wenn Ableitungen verschwinden, ist die Konvergenz von höherer Ordnung. Daher sind wir an der Konstruktion von anderen, möglichst einfachen Iterationsverfahren für die Lösung nichtlinearer Gleichungen interessiert, die eine höheren Konvergenzordnung aufweisen.

Algorithmus 4.11: Fixpunktiteration

Eingabe: Funktion $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$, maximale Anzahl N an Iterationsschritten, Startvektor $x^{(0)} \in \mathbb{R}^n$

1: **for** $k = 0 : N - 1$ **do** ▷ Iterationsschleife
2: $\quad x^{(k+1)} := \Phi(x^{(k)})$ ▷ Anwendung der Funktion Φ
3: **end for**

Ausgabe: Iterierte $x^{(N)}$

4.2 Bisektionsverfahren

Wir schauen uns nun ein klassisches Verfahren zur Bestimmung von Nullstellen für eine stetige Funktion $f\colon [a, b] \to \mathbb{R}$ an, die auf einem Intervall $[a, b] \subset \mathbb{R}$ definiert ist und deren Werte an den Endpunkten $\{a, b\}$ des Intervalls unterschiedliche Vorzeichen haben, also $f(a) \cdot f(b) < 0$ erfüllt. In dieser Situation besagt der *Zwischenwertsatz*, dass es mindestens eine Nullstelle in (a, b) gibt. Das Bisektionsverfahren ist das einfachste Verfahren für das Auffinden einer solchen Nullstelle. Dabei konstruiert man eine Intervallschachtelung, die entweder bei einer Nullstelle abbricht, oder deren Längen sich in jedem Schritt halbieren und die jeweils die Voraussetzung der unterschiedlichen Vorzeichen an den Endpunkten erfüllen. Dazu verfahren wir nach dem folgenden Schema:

Verfahren 4.12 (Bisektionsverfahren)**.** Es sei $[a^{(0)}, b^{(0)}]$ ein Intervall und $f \in C([a^{(0)}, b^{(0)}])$ eine stetige Funktion mit $f(a^{(0)}) \cdot f(b^{(0)}) < 0$. Zu $k \in \mathbb{N}_0$

(1) berechnet man den Mittelpunkt

$$x^{(k)} := \frac{a^{(k)} + b^{(k)}}{2} \in [a^{(k)}, b^{(k)}],$$

(2) überprüft das Abbruchkriterium, ob $f(x^{(k)}) = 0$ gilt (und wählt in diesem Fall $[a^{(k+1)}, b^{(k+1)}] = [x^{(k)}, x^{(k)}]$),

(3) und setzt andernfalls

$$[a^{(k+1)}, b^{(k+1)}] = \begin{cases} [a^{(k)}, x^{(k)}] & \text{für } f(a^{(k)}) \cdot f(x^{(k)}) < 0, \\ [x^{(k)}, b^{(k)}] & \text{für } f(x^{(k)}) \cdot f(b^{(k)}) = -f(a^{(k)}) \cdot f(x^{(k)}) < 0. \end{cases}$$

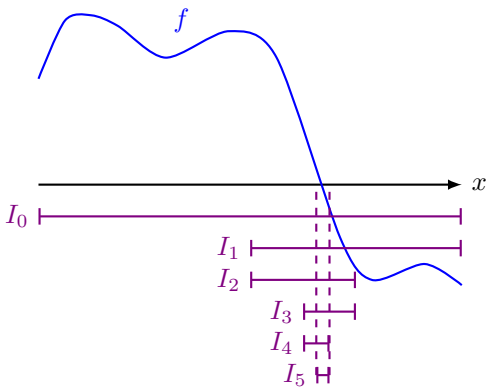
Bemerkungen 4.13.

(1) *Als Fehlerabschätzungen haben wir beim Bisektionsverfahren*

- *a priori:* $|x^{(k)} - x^*| \leq 2^{-k+1}(b-a)$,
- *a posteriori:* $|x^{(k)} - x^*| \leq 2^{-1}|x^{(k)} - x^{(k-1)}|$.

Damit halbiert sich der Fehler in jedem Schritt, so dass lineare Konvergenz gilt. Zwar ist das Verfahren dadurch sehr langsam, jedoch ist es auch sehr stabil und gering im Aufwand, so dass es sich gut als Startverfahren für andere Verfahren eignet.

(2) *Das Auffinden doppelter Nullstellen ist mit dem Bisektionsverfahren nicht möglich.*



Algorithmus 4.14: Bisektionsverfahren

Eingabe: Stetige Funktion $f\colon [a,b] \to \mathbb{R}$ mit $A := f(a)$ und $B := f(b)$, so dass $A \cdot B < 0$ gilt, Genauigkeit ε

```
while b − a < ε do                                   ▷ Iterationsschleife
    t := (a+b)/2                                     ▷ Berechnung des Mittelpunkts
    T := f(t)                                        ▷ Berechnung des Funktionswerts im Mittelpunkt
    if A · T < 0 then
        b := t und B := T                            ▷ Auswahl der linken Intervallhälfte
    end if
    if A · T > 0 then
        a := t und A := T                            ▷ Auswahl der rechten Intervallhälfte
    end if
    if A · T = 0 then
        a := t und b := t                            ▷ Nullstelle gefunden
    end if
end while
```

Ausgabe: Approximation $\frac{a+b}{2}$ einer Nullstelle von f

4.3 Newton-Verfahren

Um mit dem Newton-Verfahren die Nullstelle einer differenzierbaren Funktion zu finden, nutzt man die Ableitung der Funktion, um durch lokale Linearisierung in jedem Schritt eine verbesserte Näherung der Nullstelle zu berechnen. Dadurch erhält man ein Verfahren, das (bei guter Wahl des Startwertes) nicht nur deutlich schneller als etwa das Bisektionsverfahren konvergiert, sondern das auch im höherdimensionalen Fall anwendbar ist.

Newton-Verfahren auf $\mathbb{R}$. Für eine differenzierbare Funktion f auf einem Intervall $[a,b] \subset \mathbb{R}$ betrachtet man die Linearisierung von f um den Startwert $x^{(0)}$ (also das Taylor-Polynom der ersten Ordnung mit Entwicklungpunkt $x^{(0)}$)

$$x \mapsto f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)})$$

und bestimmt dafür (unter der Voraussetzung $f'(x^{(0)}) \neq 0$) die Nullstelle

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}.$$

Dieses Vorgehen iteriert man, berechnet also jeweils die Nullstelle der Linearisierung von f um den vorherigen Iterationspunkt, solange die Iterationspunkte im Definitionsbereich von $[a,b]$ liegen und die Ableitung von f in der Iterierten nicht verschwindet. Man geht damit nach dem folgenden Schema vor:

Verfahren 4.15 (Newton-Verfahren in einer Dimension)**.** Es sei $[a,b] \subset \mathbb{R}$ ein Intervall, $f \in C^1([a,b])$ eine differenzierbare Funktion und $x^{(0)} \in I$. Zu $k \in \mathbb{N}_0$ mit $x^{(k)} \in [a,b]$ und $f'(x^{(k)}) \neq 0$

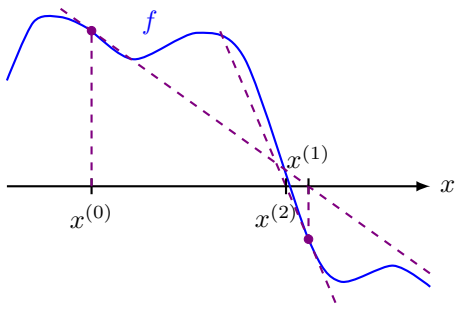
(1) berechnet man die Nullstelle der Tangente an den Punkte $(x^{(k)}, f(x^{(k)}))$

$$x^{(k+1)} := x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})},$$

(2) und überprüft das Abbruchkriterium, ob $f(x^{(k+1)}) = 0$ gilt.

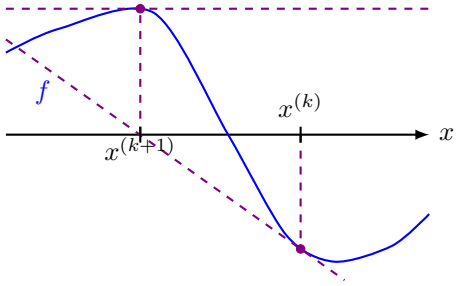
Bemerkungen 4.16.

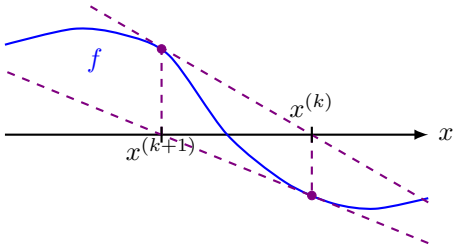
(1) *Das Newton-Verfahren konvergiert quadratisch und damit sehr schnell, jedoch muss dafür die Funktion hinreichend regulär und vor allem der Startwert geeignet gewählt sein (beispielsweise über ein anderes, langsameres Verfahren).*



(2) *Das Newton-Verfahren kann auch bei doppelten Nullstellen verwendet werden, jedoch konvergiert in diesem Fall das Standard-Verfahren nur linear (mit einer geringfügigen Modifikation kann die lokal quadratische Konvergenz aber wiederhergestellt werden).*

(3) *Bei schlechter Wahl des Startwertes ist es möglich, dass die Fixpunktiteration oszilliert oder divergiert.*





Beispiel 4.17 (Newton-Verfahren für Wurzelberechnung)**.** Wir wollen für $n \in \mathbb{Z} \setminus \{0\}$ die n-te Wurzel einer positiven Zahl $a > 0$ bestimmen. Dazu bemerken wir zunächst:

$$x^* = \sqrt[n]{a} \text{ ist eine Nullstelle der Funktion } f_n \colon \mathbb{R} \ni x \mapsto x^n - a.$$

Mit $f_n'(x) = nx^{n-1}$ für $x \in \mathbb{R}$ lautet die Iterationsvorschrift beim Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{f_n(x^{(k)})}{f_n'(x^{(k)})} = x^{(k)} - \frac{(x^{(k)})^n - a}{n(x^{(k)})^{n-1}} = \frac{1}{n}\left[(n-1)x^{(k)} + \frac{a}{(x^{(k)})^{n-1}}\right]$$

mit $k \in \mathbb{N}_0$. Als Spezialfälle haben wir mit dieser Vorschrift:

- Quadratwurzel für $n = 2$ mit $x^{(k+1)} = \frac{1}{2}[x^{(k)} + \frac{a}{x^{(k)}}]$,
- Kubikwurzel für $n = 3$ mit $x^{(k+1)} = \frac{1}{3}[2x^{(k)} + \frac{a}{(x^{(k)})^2}]$,

- Kehrwert für $n = -1$ mit $x^{(k+1)} = (2 - ax^{(k)})x^{(k)}$.

Dieses Verfahren wird numerisch für die Bestimmung von Wurzeln und Kehrwerten verwendet, wobei anhand eines Monotonieargumentes ersichtlich ist, dass das Verfahren sogar für jeden positiven Startwert $x^{(0)} > 0$ konvergiert (und zwar gegen den einzigen positiven Fixpunkt der Iterationsvorschrift, was gerade der gesuchte Wert ist).

Newton-Verfahren auf $\mathbb{R}^n$. Das Newton-Verfahren kann man auch anwenden, um Nullstellen einer (nicht-linearen) differenzierbaren Funktion $F\colon \mathbb{R}^n \supset D \to \mathbb{R}^n$ auf einem mehrdimensionalen Definitionsgebiet zu bestimmen. Auch hier wird die Funktion F lokal beim Startpunkt $x^{(0)}$ durch die affine Funktion ihres Taylor-Polynoms erster Ordnung mit Entwicklungspunkt $x^{(0)}$ approximiert und dessen Nullstelle (soweit existent) als neue Iterierte $x^{(1)}$ genommen. Da dieses Taylor-Polynom gegeben ist durch

$$x \mapsto F(x^{(0)}) + DF(x^{(0)})(x - x^{(0)}),$$

bestimmt sich $x^{(1)}$ (zumindest unter der Voraussetzung, dass die Matrix $DF(x^{(0)}) \in \mathbb{R}^{n\times n}$ invertierbar ist) über

$$x^{(1)} = x^{(0)} - \big(DF(x^{(0)})\big)^{-1} F(x^{(0)}).$$

Dieses Vorgehen wird iteriert und entsprechend eine Folge von (unterschiedlichen) linearen Gleichungssystemen gelöst, um schließlich (unter geeigneten Voraussetzungen) eine Nullstelle der Funktion F zu finden.

Verfahren 4.18 (Newton-Verfahren in mehreren Dimensionen)**.** Es sei $D \subset \mathbb{R}^n$ eine Menge, $F \in C^1(D;\mathbb{R}^n)$ eine differenzierbare Funktion und $x^{(0)} \in D$. Zu $k \in \mathbb{N}_0$ mit $x^{(k)} \in D$

(1) berechnet man den *Defekt* $F(x^{(k)})$ sowie $DF(x^{(k)})$,

(2) bestimmt (soweit möglich) das *Newton-Inkrement* $e^{(k)} \in \mathbb{R}^n$ durch Lösen des linearen Gleichungssystems
$$DF(x^{(k)})e^{(k)} = -F(x^{(k)}),$$

(3) und setzt dann
$$x^{(k+1)} = x^{(k)} + e^{(k)}.$$

Wie im linearen Fall hat man auch hier quadratische Konvergenz, und man kann sich auch leicht überlegen, dass es sich beim Newton-Verfahren um eine Fixpunktiteration handelt, und zwar für die Abbildung $\Phi\colon D \to \mathbb{R}^n$ definiert über

$$\Phi(x) \coloneqq x - (DF(x))^{-1}F(x) \quad \text{für } x \in D,$$

wobei wir ohne Einschränkung voraussetzen wollen, dass $DF(x)$ für alle $x \in D$ regulär ist (und die Iteration abgebrochen werden muss, wenn $x^{(k+1)} = \Phi(x^{(k)}) \notin D$ gilt).

Algorithmus 4.19: Newton-Verfahren

Eingabe: Differenzierbare Funktion $F\colon \mathbb{R}^n \supset D \to \mathbb{R}^n$, maximale Anzahl N an Iterationsschritten, Startvektor $x^{(0)} \in \mathbb{R}^n$

1: **for** $k = 0 : N - 1$ **do** ▷ Iterationsschleife
2: Berechne $F(x^{(k)})$ und $DF(x^{(k)})$
3: Bestimme die Lösung $e^{(k)}$ zu $DF(x^{(k)})e^{(k)} = -F(x^{(k)})$ ▷ (für $DF(x^{(k)})$ regulär)
4: $x^{(k+1)} \coloneqq x^{(k)} + e^{(k)}$ ▷ Update der Iterierten
5: **end for**

Ausgabe: $x^{(N)}$

Kapitel 5

Interpolation

Häufig ist man in einer Situation, in der man eine Reihe von Daten (etwa aus physikalischen Messungen, experimentellen Beobachtungen oder auch punktweiser Funktionsauswertung) hat und eine möglichst einfache Funktion sucht, die durch diese Datenpunkte geht. Ist $\mathcal{F}_n$ für $n \in \mathbb{N}_0$ eine "einfache" Klasse von Funktionen $f\colon X \to Y$ zwischen zwei Mengen X und Y, so dass

- jede Funktion in $\mathcal{F}_n$ durch $(n+1)$ Parameter identifiziert (und gespeichert) werden kann,
- jede Funktion in $\mathcal{F}_n$ durch "elementare" Operationen ausgewertet werden kann,

und sind

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n) \in X \times Y$$

die Daten mit paarweise verschiedenen *Knoten-* oder *Stützstellen* $\{x_i\}_{i\in\{0,1,\ldots,n\}}$ und zugehörigen *Stützwerten* $\{y_i\}_{i\in\{0,1,\ldots,n\}}$, so suchen wir im *Interpolationsproblem* nach einer Funktion $f_n \in \mathcal{F}_n$ mit

$$f_n(x_i) = y_i \quad \text{für } i \in \{0, 1, \ldots, n\}.$$

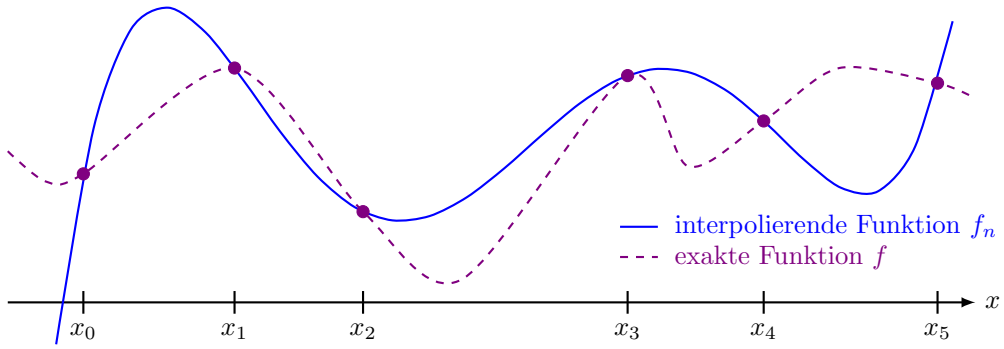
Es sind also nur die Funktionswerte in $n+1$ Punkten vorgeschrieben, und f_n soll eine gewissermaßen "sinnvolle" Fortsetzung außerhalb dieser Stützstellen darstellen. Einfache Funktionenklassen sind etwa Polynomfunktionen, Splines (stückweise polynomiale Funktionen) oder trigonometrische Funktionen (wie Sinus- und Cosinusfunktionen mit verschiedenen Periodenlängen). Für das Interpolationsproblem muss man sich dann typischerweise die folgenden Fragen stellen:

(1) Existiert die Interpolationsfunktion in der Klasse $\mathcal{F}_n$ und ist sie eindeutig?

(2) Wie bestimmt man die Interpolationsfunktion?

(3) Wie berechnet man die Interpolationsfunktion effizient außerhalb der gegebenen Stützstellen?

(4) Wie ist die Approximationsgüte (also welche Abschätzung gilt für den Fehler $\|f - f_n\|$, wenn die Stützwerte von einer Funktion f stammen)?



Die Anwendungen der Interpolation sind dabei sehr vielfältig, wie beispielsweise eine vereinfachte (approximative) Darstellung für "komplizierte" Funktionen, die Herleitung von Formeln zur numerischen Integration, die numerische Behandlung von (gewöhnlichen) Differentialgleichungen, die Erzeugung von Computer-Animationen oder die Bestimmung der Positionen von Himmelskörpern oder Satelliten in Astronomie und Raumfahrt.

5.1 Polynominterpolation

Als Raum einfacher Funktionen betrachten wir zunächst den Vektorraum aller Polynome auf $\mathbb{R}$ vom Grad höchstens $n \in \mathbb{N}$, also

$$\mathcal{P}_n := \Big\{ p(x) = \sum_{j=0}^{n} a_j x^j : a_j \in \mathbb{R} \text{ für } j \in \{0, 1, \ldots, n\} \Big\}.$$

Jedes Polynom in $\mathcal{P}_n$ ist eindeutig durch die $(n+1)$ Koeffizienten $a_0, a_1, \ldots, a_n$ bestimmt, so dass es sich bei $\mathcal{P}_n$ um einen $(n+1)$-dimensionalen Vektorraum handelt. Das *Polynominterpolationsproblem* ist nun also: finde zu gegebenen paarweise verschiedenen Stützstellen $x_0, x_1, \ldots, x_n \in \mathbb{R}$ und zugehörigen Stützwerten $y_0, y_1, \ldots, y_n$ in $\mathbb{R}$ ein Polynom $p_n \in \mathcal{P}_n$ (und damit also dessen Koeffizienten $a_0, a_1, \ldots, a_n$ in der Darstellung $p_n(x) = \sum_{j=0}^{n} a_j x^j$) mit

$$p_n(x_i) = y_i \quad \text{für } i \in \{0, 1, \ldots, n\}.$$

Existenz und Eindeutigkeit des Interpolationspolynoms. Wir überlegen uns zuerst, dass das Polynominterpolationsproblem immer eine eindeutige Lösung besitzt.

Satz 5.1. *Es sei $n \in \mathbb{N}$. Für jede Wahl von paarweise verschiedenen Stützstellen $x_0, x_1, \ldots, x_n$ in $\mathbb{R}$ und zugehörigen Stützwerten $y_0, y_1, \ldots, y_n$ in $\mathbb{R}$* existiert ein eindeutiges Polynom $p_n \in \mathcal{P}_n$ *mit*

$$p_n(x_i) = y_i \quad \textit{für alle } i \in \{0, 1, \ldots, n\}.$$

Beweis. Wir beweisen die Existenz eines solchen Polynoms über zwei verschiedene (konstruktive) Ansätze.

★ *Ansatz über Vandermonde-Matrix*: Das (lineare) Interpolationsproblem in $\mathcal{P}_n$

$$\sum_{j=0}^{n} a_j x_i^j = y_i \quad \text{für alle } i \in \{0, 1, \ldots, n\}$$

mit zu bestimmenden Koeffizienten $a_0, a_1, \ldots, a_n \in \mathbb{R}$ ist äquivalent zum Lösen des linearen Gleichungssystems

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

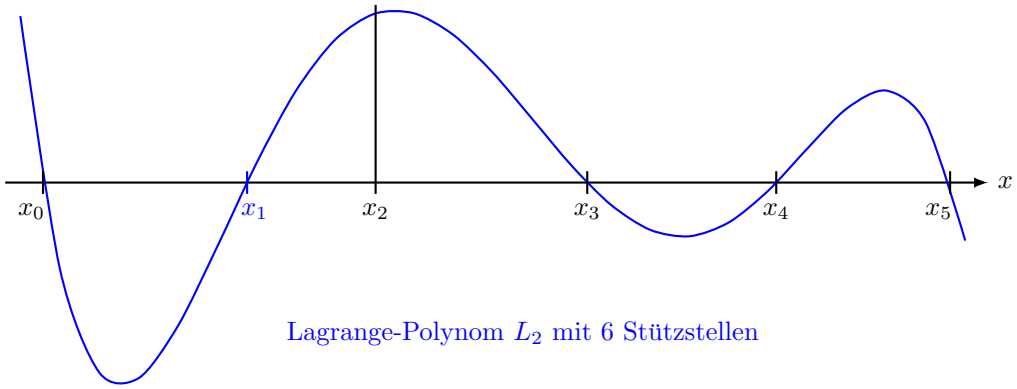
Die Matrix auf der linken Seite ist die *Vandermonde-Matrix* $V(x_0, x_1, \ldots, x_n)$, deren Determinante durch

$$\det(V(x_0, x_1, \ldots, x_n)) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_j - x_i) \tag{5.1}$$

gegeben ist (Beweis über vollständige Induktion). Da die Stützstellen nach Annahme paarweise verschieden sind, also $x_i \neq x_j$ für alle $i \neq j \in \{0, 1, \ldots, n\}$ gilt, ist $V(x_0, x_1, \ldots, x_n)$ eine reguläre Matrix. Folglich existiert eine eindeutige Lösung $(a_0, a_1, \ldots, a_n)$ des linearen Gleichungssystems und damit auch ein eindeutiges Polynom p_n vom Grad n, das das Polynominterpolationsproblem löst.

Ansatz über Lagrange-Polynome: Für $j \in \{0, 1, \ldots, n\}$ suchen wir zunächst ein Polynom $L_j \in \mathcal{P}_n$ vom Grad höchstens n, das in allen Stützstellen x_i für $i \in \{1, 2, \ldots, n\} \setminus \{j\}$ verschwindet und in x_j den Wert 1 annimmt, also mit

$$L_j(x_i) = \delta_{ij} \quad \text{für alle } i \in \{0, 1, \ldots, n\}.$$



Lagrange-Polynom L_2 mit 6 Stützstellen

Offensichtlich gilt dies mit der Wahl der sogenannten *Lagrange-Polynome*

$$L_j(x) = \prod_{\substack{\ell=0 \\ \ell \neq j}}^{n} \frac{x - x_\ell}{x_j - x_\ell} \tag{5.2}$$

Folglich ist eine Lösung des Interpolationsproblems dann gegeben durch

$$p_n(x) = \sum_{j=0}^{n} y_j L_j(x) = \sum_{j=0}^{n} y_j \prod_{\substack{\ell=0 \\ \ell \neq j}}^{n} \frac{x - x_\ell}{x_j - x_\ell} \in \mathcal{P}_n.$$

Eindeutigkeit des Interpolationspolynoms: Sind $p_n, \tilde{p}_n \in \mathcal{P}_n$ zwei Lösungen des Polynominterpolationsproblems, also mit

$$p(x_i) = y_i = \tilde{p}(x_i) \quad \text{für alle } i \in \{0, 1, \ldots, n\},$$

so hat die Funktion $p_n - \tilde{p}_n \in \mathcal{P}_n$ mindestens $(n+1)$ Nullstellen. Nach einem Korollar zum Fundamentalsatz der Algebra, der besagt, dass die Anzahl der Nullstellen eines Polynoms mit Vielfachheiten gezählt gerade dem Grad des Polynoms entspricht, muss also $p_n - \tilde{p}_n \equiv 0$ oder äquivalent dazu $p_n = \tilde{p}_n$ gelten. □

Bestimmung des Interpolationspolynoms. Wir beschäftigen uns nun mit der Frage, wie man – für gegebene Stützstellen und Stützwerte (x_i, y_i) mit $i \in \{0, 1, \ldots, n\}$ – das Interpolationspolynoms *numerisch sinnvoll* aufstellt, also mit einem stabilen Algorithmus, der möglichst wenige Operationen benötigt und bei dem das Hinzufügen von zusätzlichen Knoten einfach ist. Wir diskutieren nun die Vor- und Nachteile von verschiedenen Möglichkeiten der Berechnung. Die grundsätzliche Idee ist aber jeweils die gleiche:

(1) Wir wählen erst eine Basis $b_0, b_1, \ldots, b_n$ von $\mathcal{P}_n$, also so dass einerseits die Funktionen $b_0, b_1, \ldots, b_n$ linear unabhängig sind und sich andererseits jedes Polynom $p_n \in \mathcal{P}_n$ als Linearkombination der Funktionen $b_0, b_1, \ldots, b_n$ darstellen lässt.

(2) Wir bestimmen dann aus den Daten (x_i, y_i) für $i \in \{0, 1, \ldots, n\}$ die Koeffizienten für das Interpolationspolynom, also Zahlen $a_0, a_1, \ldots, a_n \in \mathbb{R}$ mit

$$y_i = p_n(x_i) = \sum_{j=0}^{n} a_j b_j(x_i) \quad \text{für } i \in \{0, 1, \ldots, n\},$$

was gerade dem Lösen des folgenden linearen Gleichungssystems entspricht:

$$\begin{pmatrix} b_0(x_0) & b_1(x_0) & \dots & b_n(x_0) \\ b_0(x_1) & b_1(x_1) & \dots & b_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ b_0(x_n) & b_1(x_n) & \dots & b_n(x_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

(a) Darstellung mit Monomen. Wählen wir aus $\mathcal{P}_n$ gerade die Monome $1, x, \dots, x^n$, so sind diese Funktionen offensichtlich linear unabhängig und spannen damit den $(n+1)$-dimensionalen Vektorraum $\mathcal{P}_n$ auf. Für die Bestimmung der Koeffizienten $a_0, a_1, \dots, a_n \in \mathbb{R}$ des Interpolationspolynoms

$$p_n(x) = \sum_{j=0}^{n} a_j x^j$$

in der Darstellung mit Monomen zu den Datenpunkten (x_i, y_i) mit $i \in \{0, 1, \dots, n\}$ tritt in dem linearen Gleichungssystem dann die Vandermonde-Matrix (vgl. den ersten Beweis von Satz 5.1) auf.

Beispiel 5.2. Wir wollen die Funktion $\mathbb{R} \ni x \mapsto \exp(x)$ an den Stützstellen $x_0 = 0$, $x_1 = 1$ und $x_2 = 2$ durch ein Polynom zweiten Grades approximieren. Damit muss das Polynom durch die Punkte

$$(0, \exp(0)) = (0, 1), \quad (1, \exp(1)) = (1, \mathrm{e}) \quad \text{und} \quad (2, \exp(2)) = (2, \mathrm{e}^2)$$

gehen. Mit den Monomen als Basis hat das Interpolationspolynom die Darstellung

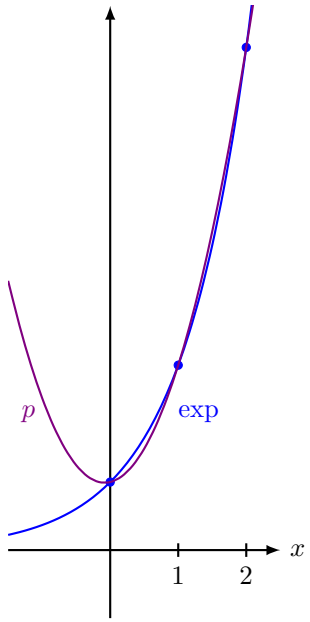
$$p_2(x) = a_0 + a_1 x + a_2 x^2,$$

wobei die Koeffizienten $a_0, a_1, a_2 \in \mathbb{R}$ über das Gleichungssystem

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \mathrm{e} \\ \mathrm{e}^2 \end{pmatrix}$$

bestimmt sind. Mit $a_0 = 1$, $a_1 = -\frac{\mathrm{e}^2}{2} + 2\mathrm{e} - \frac{3}{2}$ und $a_2 = \frac{\mathrm{e}^2}{2} - \mathrm{e} + \frac{1}{2}$ erhalten wir dann als Interpolationspolynom in der Darstellung mit Monomen die Funktion

$$p_2(x) = 1 + \left[-\frac{\mathrm{e}^2}{2} + 2\mathrm{e} - \frac{3}{2} \right] x + \left[\frac{\mathrm{e}^2}{2} - \mathrm{e} + \frac{1}{2} \right] x^2.$$



⊕ Die Basis ist einfach zu bestimmen (und zu ergänzen).

⊖ Die Vandermonde-Matrix ist typischerweise sehr schlecht konditioniert (insbesondere, wenn die Stützstellen nahe beieinander liegen).

⧖ Mit $O(n^3)$ für die Bestimmung des Interpolationspolynoms in Monomdarstellung (wegen der voll besetzten Vandermonde-Matrix) ist der Aufwand sehr hoch. Für die Auswertung des Interpolationspolynoms an einer (neuen) Stelle werden dann $O(n)$ Multiplikationen benötigt.

(b) Darstellung mit Lagrange-Polynomen. Wählen wir aus $\mathcal{P}_n$ die Lagrange-Polynome

$$L_j(x) \coloneqq \prod_{\substack{\ell=0 \\ \ell \neq j}}^{n} \frac{x - x_\ell}{x_j - x_\ell} \quad \text{für } j \in \{0, 1, \dots, n\}$$

aus (5.2), so sind diese Funktionen wegen $L_j(x_i) = \delta_{ij}$ für alle $i, j \in \{0, 1, \ldots, n\}$ linear unabhängig. Folglich spannen die Lagrange-Polynome $L_0, L_1, \ldots, L_n$ den $(n+1)$-dimensionalen Vektorraum $\mathcal{P}_n$ auf. Das Interpolationspolynom zu den Datenpunkten (x_i, y_i) mit $i \in \{0, 1, \ldots, n\}$ in der Darstellung mit Lagrange-Polynomen ist dann

$$p_n(x) = \sum_{j=0}^{n} y_j L_j(x)$$

(vgl. den zweiten Beweis von Satz 5.1).

Beispiel 5.3. Um die Funktion $\mathbb{R} \ni x \mapsto \exp(x)$ an den Stützstellen $x_0 = 0$, $x_1 = 1$ und $x_2 = 2$ durch ein Polynom zweiter Ordnung zu approximieren, bestimmen wir zunächst die Lagrange-Polynome

$$L_0(x) = \frac{1}{2}(x-1)(x-2), \quad L_1(x) = -x(x-2) \quad \text{und} \quad L_2(x) = \frac{1}{2}x(x-1).$$

Mit den Stützwerten $y_0 = 1$, $y_1 = \mathrm{e}$ und $y_2 = \mathrm{e}^2$ erhalten wir dann sofort

$$p_2(x) = 1 \cdot \frac{1}{2}(x-1)(x-2) - \mathrm{e} \cdot x(x-2) + \mathrm{e}^2 \cdot \frac{1}{2}x(x-1)$$

als Interpolationspolynom in der Darstellung mit Lagrange-Polynomen (das natürlich mit dem über die Monome in Beispiel 5.2 dargestellten Interpolationspolynom übereinstimmt).

- ⊕ Die Koeffizienten des Interpolationspolynoms mit Lagrange-Polynomen sind trivial.
- ⊕ Für theoretische Fragestellungen ist die Darstellung mit Lagrange-Polynomen praktisch (etwa zur Untersuchung der Kondition des Polynominterpolationsproblems).
- ⊖ Die Basisfunktionen sind aufwändig zu bestimmen. Zudem werden sie durch das Hinzufügen neuer Stützstellen alle geändert (mit leicht modifizierter Darstellung ist dieser Aspekt jedoch unproblematisch).
- ⧖ $O(n)$ Multiplikationen und Divisionen für die Auswertung eines Lagrange-Polynoms und damit $O(n^2)$ für die Auswertung des Interpolationspolynoms an einer (neuen) Stelle.

(c) Darstellung mit Newton-Polynomen. Wählen wir aus $\mathcal{P}_n$ die Newton-Polynome

$$N_j(x) \coloneqq \prod_{k=0}^{j-1}(x - x_k) \quad \text{für } j \in \{0, 1, \ldots, n\},$$

so sind diese Polynome alle von unterschiedlichem Grad (höchstens n) und damit linear unabhängig. Folglich spannen die Newton-Polynome $N_0, N_1, \ldots, N_n$ den $(n+1)$-dimensionalen Vektorraum $\mathcal{P}_n$ auf. Um nun die Koeffizienten $a_0, a_1, \ldots, a_n \in \mathbb{R}$ des Interpolationspolynoms

$$p_n(x) = \sum_{j=0}^{n} a_j N_j(x)$$

zu den Datenpunkten (x_i, y_i) mit $i \in \{0, 1, \ldots, n\}$ zu bestimmen, müssen wir die $(n+1)$ Gleichungen

$$p_n(x_i) = \sum_{j=0}^{i} a_j N_j(x_i) \overset{!}{=} y_i \quad \text{für } i \in \{0, 1, \ldots, n\}$$

lösen (wobei die Summation für die Stützstelle x_i mit $i \in \{0, 1, \ldots, n\}$ bei i abbricht, da nach Definition der Newton-Polynome $N_j(x_i) = 0$ für $j \in \{i+1, \ldots, n\}$ gilt). Dies führt auf das lineare Gleichungssystem

$La = y$ mit $L \in \mathbb{R}^{(n+1)\times(n+1)}$ in unterer Dreiecksform, mit

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 - x_0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \dots & (x_n - x_0) \cdot \ldots \cdot (x_n - x_{n-1}) \end{pmatrix}.$$

Dieses Gleichungssystem können wir durch Vorwärtssubstitution lösen. Für eine schöne Darstellung des gesuchten Interpolationspolynoms führen wir dividierte Differenzen ein:

Definition 5.4 (dividierte Differenzen)**.** *Es seien $x_0, x_1, \dots, x_n \in \mathbb{R}$ paarweise verschiedene Stützstellen mit zugehörigen Stützwerten $y_0, y_1, \dots, y_n \in \mathbb{R}$. Wir definieren die* dividierten Differenzen erster Ordnung *über*

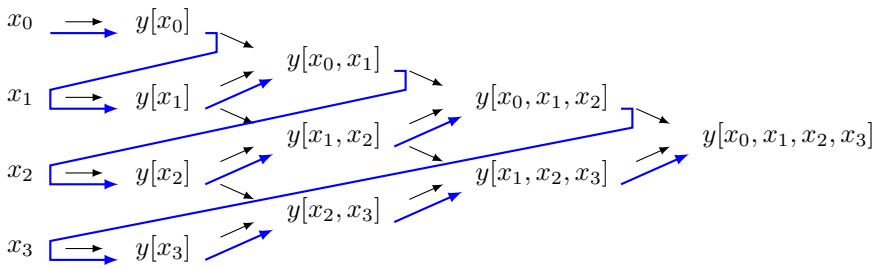
$$y[x_\ell] \coloneqq y_\ell \quad \text{für } \ell \in \{0, 1, \dots, n\}$$

und dann rekursiv für $i \in \{1, 2, \dots, n\}$ die dividierten Differenzen i-ter Ordnung *über*

$$y[x_\ell, \dots, x_{\ell+i}] \coloneqq \frac{y[x_{\ell+1}, \dots, x_{\ell+i}] - y[x_\ell, \dots, x_{\ell+i-1}]}{x_{\ell+i} - x_\ell} \quad \text{für } \ell \in \{0, 1, \dots, n-i\}.$$

Sind die Stützwerte über eine Funktion f gegeben, also $y_\ell = f(x_\ell)$ für $\ell \in \{0, 1, \dots, n\}$, so notieren wir die dividierten Differenzen i-ter Ordnung auch mit $f[x_\ell, \dots, x_{\ell+i}]$ anstelle von $y[x_\ell, \dots, x_{\ell+i}]$ für $i \in \{0, 1, 2, \dots, n\}$ und $\ell \in \{0, 1, \dots, n-i\}$.

Bemerkung 5.5 (Berechnung der dividierten Differenzen)**.** *Wir können die zu berechnenden dividierten Differenzen (sowie die Reihenfolge der Berechnung) auch* graphisch darstellen:



Über die dividierten Differenzen können wir nun die Darstellung des Interpolationspolynoms mit Newton-Polynomen angeben.

Satz 5.6 (Interpolationspolynom in Newton-Darstellung)**.** *Es seien $x_0, x_1, \dots, x_n \in \mathbb{R}$ paarweise verschiedene Stützstellen mit zugehörigen Stützwerten $y_0, y_1, \dots, y_n \in \mathbb{R}$. Dann ist das Interpolationspolynom $p_n \in \mathcal{P}_n$ mit $p_n(x_i) = y_i$ für $i \in \{0, 1, \dots, n\}$ gegeben durch*

$$\begin{aligned} p_n(x) &= y[x_0] + y[x_0, x_1](x - x_0) + \ldots + y[x_0, \dots, x_n](x - x_0) \cdot \ldots \cdot (x - x_{n-1}) \\ &= \sum_{j=0}^{n} y[x_0, \dots, x_j] N_j(x). \end{aligned}$$

★ *Beweisskizze.* Wir gehen über vollständige Induktion über $n \in \mathbb{N}_0$ vor. Der Induktionsanfang ist dabei wegen $p_0(x_0) = y[x_0] = y_0$ und $N_0 \equiv 1$ klar. Nun nehmen wir für ein beliebiges, aber festes $n \in \mathbb{N}$ an, dass das Interpolationspolynom durch die Punkte $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1})$ die Darstellung

$$p_{n-1}(x) = \sum_{j=0}^{n-1} y[x_0, \dots, x_j] N_j(x)$$

hat. Setzen wir nun

$$p_n(x) \coloneqq p_{n-1}(x) + a_n N_n(x)$$

mit einem noch zu bestimmenden Koeffizienten $a_n \in \mathbb{R}$, so stellen wir zunächst fest, dass nach Definition der Newton-Polynome $N_n(x_i) = 0$ für alle $i \in \{0, 1, \ldots, n-1\}$ gilt und damit aus der Induktionsannahme

$$p_n(x_i) = p_{n-1}(x_i) = y_i \quad \text{für } i \in \{0, 1, \ldots, n-1\}$$

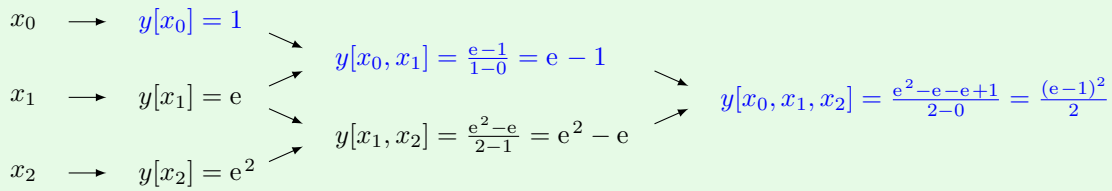
folgt. Wegen $N_n(x_n) \neq 0$ können wir zudem durch eine passende Wahl von $a_n \in \mathbb{R}$ sicherstellen, dass auch die letzte Forderung $p_n(x_n) = y_n$ erfüllt ist:

$$\begin{aligned}
a_n N_n(x_n) &\overset{!}{=} y_n - p_{n-1}(x_n) \\
&= \underbrace{y[x_n] - y[x_0]}_{=y[x_n,x_0](x_n-x_0)} - \sum_{j=1}^{n-1} y[x_0, \ldots, x_j] N_j(x_n) \\
&= \big(\underbrace{y[x_n, x_0] - y[x_0, x_1]}_{=y[x_n,x_0,x_1](x_n-x_1)}\big) N_1(x_n) - \sum_{j=2}^{n-1} y[x_0, \ldots, x_j] N_j(x_n) \\
&= \ldots = \big(\underbrace{y[x_n, x_0, \ldots, x_{n-2}] - y[x_0, \ldots, x_{n-1}]}_{=y[x_n,x_0,\ldots,x_{n-2}](x_n-x_{n-1})}\big) N_{n-1}(x_n) \\
&= y[x_n, x_0, \ldots, x_{n-1}] N_n(x_n).
\end{aligned}$$

Wegen $N_n(x_n) \neq 0$ zeigt dies $a_n = y[x_n, x_0, \ldots, x_{n-1}]$. Um den Beweis nun abzuschließen, muss man noch rechtfertigen, dass die Reihenfolge der Stützstellen für die Berechnung der dividierten Differenzen keine Rolle spielt. Damit haben wir dann wie behauptet als letzten Koeffizienten

$$a_n = y[x_n, x_0, \ldots, x_{n-1}] = y[x_0, \ldots, x_n]. \qquad \square$$

Beispiel 5.7. Um die Funktion $\mathbb{R} \ni x \mapsto \exp(x)$ an den Stützstellen $x_0 = 0$, $x_1 = 1$ und $x_2 = 2$ durch ein Polynom zweiten Grades zu approximieren, berechnen wir die dividierten Differenzen



und erhalten damit

$$p(x) = 1 + (\mathrm{e} - 1) \cdot x + \frac{(\mathrm{e} - 1)^2}{2} \cdot x \cdot (x - 1)$$

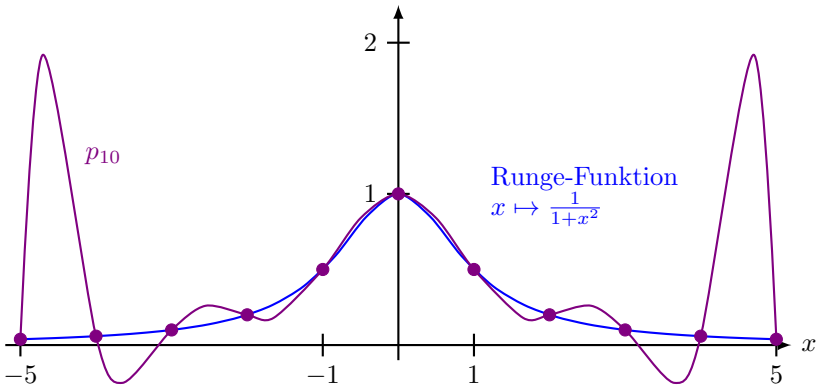
als Interpolationspolynom in der Darstellung mit Newton-Polynomen.

- ⊕ Weitere Stützstellen lassen sich einfach hinzufügen.
- ⊕ Der Algorithmus ist numerisch stabil.
- ⊕ Das Interpolationspolynom kann auch ohne explizite Berechnung gut außerhalb der Stützstellen ausgewertet werden (Aitken–Neville-Schema).
- ⊖ Bei Änderung der Stützwerte müssen neue dividierte Differenzen aufgestellt werden.
- ⧖ $O(n^2)$ Divisionen für das Berechnen der Koeffizienten $y[x_0, \ldots, x_j]$ für $j \in \{0, 1, \ldots, n\}$, $O(n)$ Multiplikationen für die Auswertung des Interpolationspolynoms an einer (neuen) Stelle mit dem Horner-Schema.

Approximationsgüte bei der Interpolation von Funktionen. Wir nehmen nun an, dass x_0, $x_1, \ldots, x_n$ paarweise verschiedene Stützstellen aus einem Intervall I sind und dass die Stützwerte $y_0, y_1, \ldots, y_n \in \mathbb{R}$ von einer C^{n+1}-Funktion $f\colon I \to \mathbb{R}$ kommen, also $y_j = f(x_j)$ für $j \in \{0, 1, \ldots, n\}$ gilt. In diesem Fall kann man zeigen, dass der Interpolationsfehler abgeschätzt ist über

$$|f(x) - p_n(x)| \leq \frac{1}{(n+1)!} \sup_I |f^{(n+1)}||N_{n+1}(x)|.$$

Man kann zwar über die Lage der Stützstellen optimieren, so dass der Term $N_{n+1}(x)$ betragsmäßig minimal wird, jedoch führt ein höherer Grad des Interpolationspolynoms gerade an den äußeren Stützstellen zu großen Oszillationen des Interpolationspolynoms (*Runges Phänomen*), so dass man anstelle der Polynominterpolation besser mit stückweise polynomialer Interpolation, der sogenannten Spline-Interpolation, arbeiten sollte.



5.2 Spline-Interpolation

Wir haben gesehen, dass sich die Polynominterpolation nicht immer gut eignet, um Funktionen zu approximieren, da es bei Vermehrung der Stützstellen zu starken Oszillationen und damit auch zu einem großen Approximationsfehler kommen kann. Dies liegt daran, dass die Polynominterpolation "global" definiert ist, also alle Stützstellen gleichzeitig berücksichtigt werden und das Interpolationsproblem damit durch ein einziges Polynom gelöst werden, und recht "steif" dadurch ist, dass bei der Polynominterpolation Glattheit an allen Stützstellen gefordert wird. Für eine lokale Betrachtung und zur Reduzierung der Steifheit wollen wir das Interpolationsproblem nun mit stückweise polynomialen Funktionen, den Splines, lösen, die an den Stütz- oder Verklebestellen nur noch bestimmte Regularitätsbedingungen erfüllen, die an den Polynomgrad der Kurve zwischen den Stützstellen angepasst sind.

Grundlagen zu Splines. Der Begriff "Spline" stammt ursprünglich aus dem Handwerk und bezeichnet dort ein biegsames Kurvenlineal, das zum Zeichnen gleichmäßig geschwungener Kurven verwendet werden kann.

Definition 5.8 (Spline). *Es sei $\Delta = \{x_0, x_1, \ldots, x_n\}$ ein* Gitter *des Intervalls $[a, b]$, d.h. mit*

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

Ein Spline *oder eine* Splinefunktion vom Grad $k \in \mathbb{N}$ (bzw. der Ordnung $k+1$) bezüglich Δ *ist eine Funktion $s\colon [a, b] \to \mathbb{R}$, so dass*

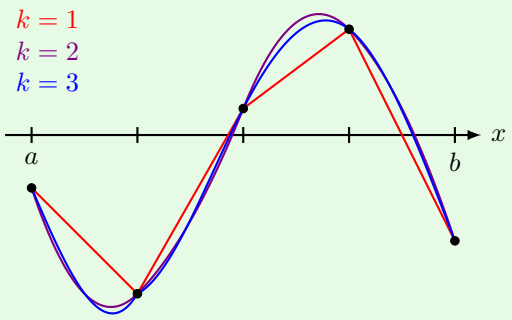
- *$s \in C^{k-1}([a, b])$ gilt, also $(k-1)$-mal stetig differenzierbar (bzw. im Fall $k = 1$ stetig) ist,*
- *$s|_{[x_j, x_{j+1}]} \in \mathcal{P}_k$ für $j \in \{0, 1, \ldots, n-1\}$ gilt, also stückweise aus Polynomen vom Grad höchstens k besteht.*

Den Vektorraum aller Splines vom Grad $k \in \mathbb{N}$ bezüglich Δ bezeichnen wir ferner mit $\mathcal{S}_{k,\Delta}$.

Bemerkung 5.9. *Ist $s \in \mathcal{S}_{k,\Delta}$ ein Spline vom Grad $k \in \mathbb{N}$ bezüglich eines Gitters Δ aus $n+1$ Punkten, so ist dieser durch $n(k+1)$ Koeffizienten beschrieben (nämlich die Koeffizienten der Polynome vom Grad k auf n Intervallen zwischen aufeinanderfolgenden Gitterpunkten). An den inneren Knoten werden dann aber noch $(n-1)k$ Stetigkeits- und Differenzierbarkeitsbedingungen gestellt. Damit ist $\mathcal{S}_{k,\Delta}$ ein Vektorraum der Dimension $n(k+1)-(n-1)k = n+k$.*

Beispiele 5.10. An den Stützstellen

- stimmen für $k = 1$ (*lineare Splines*) nur die Funktionswerte überein (und lineare Splines sind damit in der Regel nicht differenzierbar),
- stimmen für $k = 2$ (*quadratische Splines*) die Funktionswerte und die erste Ableitung überein,
- stimmen für $k = 3$ (*kubische Splines*) die Funktionswerte, die erste und die zweite Ableitung überein (so dass kubische Splines als C^2-Funktionen vom Auge als "glatt" wahrgenommen werden).



Wir kehren nun wieder zum ursprünglichen Interpolationsproblem zurück und wollen zu vorgegebenen Stützstellen $a = x_0 < x_1 < \ldots < x_n = b$ mit zugehörigen Stützwerten $y_0, \ldots, y_n \in \mathbb{R}$ eine Interpolationsfunktion s in der Klasse $\mathcal{S}_{k,\Delta}$ aller Splines vom Grad $k \in \{0, 1, \ldots, n-1\}$ finden, die also

$$s(x_j) = y_j \qquad \text{für alle } j \in \{0, 1, \ldots, n\} \tag{5.3}$$

erfüllt. Wir konzentrieren uns dabei auf den Fall $k = 3$ der kubischen Splines, mit denen in der Praxis häufig gearbeitet wird.

Existenz (und Eindeutigkeit) interpolierender kubischer Splines. Wir fassen zunächst zusammen, welche Unbekannten es bei der Lösung des Interpolationsproblems über kubische Splines gibt und welche Bedingungen erfüllt sein müssen:

- Es sind n kubische Funktionen $x \mapsto a_j + b_j x + c_j x^2 + d_j x^3$ auf den Intervallen $[x_j, x_{j+1}]$ mit $j \in \{0, \ldots, n-1\}$ zu finden, so dass wir insgesamt $4n$ *Unbekannte* haben.
- Mit der Forderung $s(x_j) = y_j$ für $j \in \{0, \ldots, n\}$ gibt es $(n+1)$ *Interpolationsbedingungen.*
- Wegen der Forderung $s \in C^2([a,b])$ ergeben sich an den inneren Punkten $3(n-1)$ *Stetigkeits- und Differenzierbarkeitsbedingungen*, nämlich

$$\begin{aligned} \lim_{x \nearrow x_j} s(x) &= \lim_{x \searrow x_j} s(x) \\ \lim_{x \nearrow x_j} s'(x) &= \lim_{x \searrow x_j} s'(x) \\ \lim_{x \nearrow x_j} s''(x) &= \lim_{x \searrow x_j} s''(x) \end{aligned}$$

  für $j \in \{1, \ldots, n-1\}$.

Man kann sich leicht überlegen, dass diese Bedingungen linear in den Unbekannten und linear unabhängig sind, so dass man für die Existenz eines *eindeutigen* interpolierenden kubischen Splines noch

$$4n - (n+1) - 3(n-1) = 2$$

zusätzliche (linear unabhängige) Bedingungen stellen muss (dies ist auch klar angesichts der Tatsache, dass $S_{3,\Delta}$ nach Bemerkung 5.9 ein Vektorraum der Dimension $n+3$ ist, es sich aber beim Interpolationsproblem nur um $(n+1)$ Bedingungen handelt, die linear im Spline sind). Typische Fälle sind die folgenden Randbedingungen:

(a) *periodische Randbedingung*:

$$s'(a) = s'(b) \quad \text{und} \quad s''(a) = s''(b)$$

(insbesondere dann relevant, wenn man Periodizität mit Periode $b - a$ erwartet, z.B. wenn die Stützwerte $y_0, \dots, y_n$ die Funktionswerte einer $(b-a)$-periodischen Funktion sind),

(b) *vollständige Randbedingung*:

$$s'(a) = y'_0 \quad \text{und} \quad s'(b) = y'_n$$

für vorgegebene Werte $y'_0, y'_n \in \mathbb{R}$,

(c) *natürliche Randbedingung*

$$s''(a) = 0 \quad \text{und} \quad s''(b) = 0$$

(in diesem Fall kann man den Spline s in $\{x < a\}$ und in $\{x > b\}$ durch affine Funktionen zu einer C^2-Funktion auf ganz $\mathbb{R}$ fortsetzen).

Jede der Randbedingungen (a), (b) und (c) führen auf lineare Gleichungssysteme für die unbekannten Koeffizienten des kubischen Splines, die eindeutig lösbar sind. Damit folgt die eindeutige Existenz kubischer Splines zum Interpolationsproblem unter jeder dieser Randbedingungen:

Satz 5.11. *Zu jeder der obigen Randbedingungen* (a)–(c) *existiert genau ein Spline $s \in S_{3,\Delta}$, der das Interpolationsproblem* (5.3) *löst.*

Bemerkung 5.12. *Für die Berechnung der Splines gibt es verschiedene effiziente Verfahren.*

5.3 Trigonometrische Interpolation

Zur Interpolation bei periodischen Daten (etwa zur Interpolation von periodischen Funktionen) ist es sinnvoll, mit trigonometrischen Polynomen zu arbeiten, also mit Funktionen der Form

$$t_{n-1}(x) = \sum_{k=0}^{n-1} c_k \exp(\mathrm{i}\, xk) \quad \text{für } x \in [0, 2\pi)$$

mit $n \in \mathbb{N}$ und Koeffizienten $c_0, \dots, c_{n-1} \in \mathbb{C}$ (beachte, dass die Funktion t_{n-1} damit ein Polynom in der Größe $\exp(\mathrm{i}\, x)$ ist). Wir beschränken uns hier auf den Fall der Periodenlänge 2π (und wir erhalten daraus dann durch Transformation den allgemeinen Fall mit beliebiger Periodenlänge). Beim *trigonometrischen Interpolationsproblem* suchen wir nun zu paarweise verschiedenen Stützstellen $x_0, \dots, x_{n-1} \in [0, 2\pi)$ mit zugehörigen (komplexen) Stützwerten $y_0, \dots, y_{n-1} \in \mathbb{C}$ ein trigonometrisches Polynom t_{n-1} mit

$$t_{n-1}(x_j) = y_j \quad \text{für } j \in \{0, \dots, n-1\}. \tag{5.4}$$

Berechnung des trigonometrischen Interpolationspolynoms bei äquidistanten Stützstellen. Wir diskutieren nun, wie man im Fall der äquidistanten Stützstellen

$$x_j \coloneqq \frac{2\pi j}{n} \quad \text{für } j \in \{0, 1, \dots, n-1\}$$

die Koeffizienten des trigonometrischen Interpolationspolynoms mithilfe von *diskreter Fourier-Analysis* berechnen kann.

Definition 5.13 (diskrete Fourier-Transformation)**.** *Es sei $n \in \mathbb{N}$. Die* diskrete Fourier-Transformation (DFT) *$\mathcal{F}_n \colon \mathbb{C}^n \to \mathbb{C}^n$, $(y_0, y_1, \dots, y_{n-1}) \mapsto (c_0, c_1, \dots, c_{n-1})$ wird definiert über die Vorschrift*

$$c_k \coloneqq \frac{1}{n} \sum_{\ell=0}^{n-1} y_\ell \exp\left(-\frac{2\pi \,\mathrm{i}\, k\ell}{n} \right) \quad \text{für } k \in \{0, 1, \dots, n-1\}.$$

Bemerkung 5.14 (Vergleich mit der Fourier-Transformation)**.** *Die* Fourier-Transformierte *einer 2π-periodischen integrierbaren Funktion $f \colon \mathbb{R} \to \mathbb{C}$ ist über*

$$d_k = \mathcal{F}(f)(k) \coloneqq \frac{1}{2\pi} \int_0^{2\pi} f(x) \exp(-\mathrm{i}\, kx)\, dx \quad \text{für } k \in \mathbb{Z}$$

definiert. Die diskrete Fourier-Transformation entspricht einer Approximation dieses Integrals für eine Funktion f mit $f(2\pi j/n) = y_j$ für $j \in \{0, 1, \dots, n-1\}$ mit äquidistant gewählten Stützstellen. Die Abbildung der Fourier-Transformation ist (unter geeigneten Voraussetzungen an die Funktion f) auch invertierbar, d.h. man kann aus der Fourier-Transformation auf die ursprüngliche Funktion f zurückschließen, und dies erfolgt über die Fourier-Reihe

$$\sum_{k=-\infty}^{\infty} d_k \exp(\mathrm{i}\, kx) \quad \text{für } x \in [0, 2\pi).$$

Die diskrete Fourier-Transformation entspricht einer Matrix-Vektor-Multiplikation, und diese kann analog zum kontinuierlichen Fall auch invertiert werden.

Lemma 5.15 (inverse diskrete Fourier-Transformation)**.** *Es sei $n \in \mathbb{N}$. Die Umkehrabbildung $\mathcal{F}_n^{-1} \colon \mathbb{C}^n \to \mathbb{C}^n$, $(c_0, c_1, \dots, c_{n-1}) \mapsto (y_0, y_1, \dots, y_{n-1})$ zu $\mathcal{F}_n$ existiert, ist gegeben durch*

$$y_j = \sum_{k=0}^{n-1} c_k \exp\left(\frac{2\pi \,\mathrm{i}\, jk}{n} \right) \quad \text{für } j \in \{0, 1, \dots, n-1\}$$

und wird auch die inverse diskrete Fourier-Transformation (iDFT) *genannt.*

★ *Beweis.* Wir bemerken für die Wahl der äquidistanten Stützstellen, dass $\exp(\mathrm{i}\, x_j) = \exp(\frac{2\pi \,\mathrm{i}\, j}{n})$ für $j \in \{0, 1, \dots, n-1\}$ gerade den *n-ten Einheitswurzeln* entspricht. Für $m \in \mathbb{Z}$ beliebig haben wir hier mit Teleskopsummeneigenschaft

$$\left(1 - \exp\left(\frac{2\pi \,\mathrm{i}\, m}{n}\right)\right) \sum_{k=0}^{n-1} \exp\left(\frac{2\pi \,\mathrm{i}\, km}{n}\right) = 1 - \exp\left(\frac{2\pi \,\mathrm{i}\, nm}{n}\right) = 0.$$

Folglich muss

$$\exp\left(\frac{2\pi \,\mathrm{i}\, m}{n}\right) = 1 \quad \Rightarrow \quad m \in n\mathbb{Z} \quad \text{und} \quad \sum_{k=0}^{n-1} \exp\left(\frac{2\pi \,\mathrm{i}\, km}{n}\right) = \sum_{k=0}^{n-1} 1 = n$$

oder

$$\sum_{k=0}^{n-1} \exp\left(\frac{2\pi \,\mathrm{i}\, km}{n}\right) = 0$$

gelten. Für $j \in \{0, 1, \dots, n-1\}$ folgt mit dieser Vorüberlegung die Behauptung über Vertauschung der Summationen

$$\begin{aligned} \sum_{k=0}^{n-1} c_k \exp\left(\frac{2\pi \,\mathrm{i}\, jk}{n}\right) &= \frac{1}{n} \sum_{k=0}^{n-1} \sum_{\ell=0}^{n-1} y_\ell \exp\left(-\frac{2\pi \,\mathrm{i}\, k\ell}{n}\right) \exp\left(\frac{2\pi \,\mathrm{i}\, jk}{n}\right) \\ &= \frac{1}{n} \sum_{\ell=0}^{n-1} y_\ell \underbrace{\sum_{k=0}^{n-1} \exp\left(\frac{2\pi \,\mathrm{i}\, k(j-\ell)}{n}\right)}_{=n\delta_{j\ell}} = y_j. \end{aligned}$$

□

Mit der Aussage von Lemma 5.15 haben wir nun das trigonometrische Interpolationsproblem (5.4) gelöst: die gesuchten Koeffizienten $c_0, c_1, \ldots, c_{n-1} \in \mathbb{C}$ sind gerade die diskrete Fourier-Transformation des Vektors $(y_0, y_1, \ldots, y_{n-1}) \in \mathbb{C}^n$ der gegebenen Stützwerte.

Bemerkung 5.16 (effiziente Berechnung der DFT)**.** *Die naive Berechnung der diskreten Fourier-Transformation erfordert insgesamt $O(n^2)$ Rechenoperationen. Ist n eine Zweierpotenz, so kann man die zu berechnenden Summen in der diskreten Fourier-Transformation in gerade und ungerade Terme zerlegen und so geschickt aufsummieren, dass man Partialsummen wiederverwenden kann. Iteriert man dies, so liefert dies einen Algorithmus zur effizienten Berechnung der diskreten Fourier-Transformation, der insgesamt nur $O(n \log(n))$ Rechenoperationen benötigt und den man als* schnelle Fourier-Transformation (*abgekürzt* FFT *für* Fast Fourier Transform) *bezeichnet.*

Kapitel 6

Numerische Integration

In diesem Kapitel diskutieren wir die numerische Approximation von Integralen

$$I(f) \coloneqq \int_a^b f(x)\,dx \quad \text{für } f \in C([a,b]),$$

was auch als *Quadratur* bezeichnet wird. Dies ist oft insbesondere notwendig, wenn der Integrand nicht elementar integrierbar ist (wie etwa $x \mapsto \exp(-x^2)$ beim Gauß-Integral), wenn die Auswertung der Stammfunktion numerisch sehr aufwändig ist, oder wenn anstelle des Integranden auf ganz $[a,b]$ nur dessen Werte an einigen diskreten Punkten bekannt sind (etwa bei Messreihendaten).

Die allgemeine Idee ist für die Approximation des Integrals ist es, den Integranden (ggf. auf Teilintervallen) durch ein Interpolationspolynom zu approximieren und dieses anschließend exakt zu integrieren. Dies führt auf gewichtete Summen von Funktionswerten des Integranden in ausgewählten Punkten.

Definition 6.1 (Quadraturformel)**.** *Eine* Quadraturformel *ist eine lineare Abbildung* $\hat{I}_n = \hat{I}\colon C([a,b]) \to \mathbb{R}$ *mit* $n \in \mathbb{N}$*, die als gewichtete Summe von Funktionswerten von der Form*

$$\hat{I}_n(f) = (b-a)\sum_{j=0}^{n} \lambda_j f(x_j) \quad \textit{für } f \in C([a,b])$$

gegeben ist, wobei $x_0, x_1, \ldots, x_n$ *paarweise verschiedene Punkte* (Quadraturpunkte *oder* Knoten) *in* $[a,b]$ *und* $\lambda_0, \lambda_1, \ldots, \lambda_n$ Gewichte *in* $\mathbb{R}$ *sind. Wird die Quadraturformel über Approximation des Integranden durch ein Interpolationspolynom an den Knoten und dessen anschließende Integration erhalten, so sprechen wir auch von einer* interpolatorischen Quadraturformel*.*

Zur Bestimmung der "Qualität" einer Quadraturformel muss man den Fehler von der Quadraturformel im Vergleich zum exakten Integralwert kontrollieren. Da man Polynome leicht exakt integrieren kann, überprüft man typischerweise, bis zu welchem Polynomgrad kein Fehler auftritt und die Quadraturformel damit exakt ist.

Definition 6.2 (Quadraturfehler und polynomialer Exaktheitsgrad)**.** *Für eine Quadraturformel* $\hat{I}\colon C([a,b]) \to \mathbb{R}$ *bezeichnen wir*

$$I(f) - \hat{I}(f) \quad \left(= \int_a^b f(x)\,dx - (b-a)\sum_{j=0}^{n} \lambda_j f(x_j) \right)$$

als Quadraturfehler*. Wir sagen außerdem, dass die Quadraturformel den* polynomialen Exaktheitsgrad $k \in \mathbb{N}_0$ *hat, falls*

- $\hat{I}(p_k) = I(p_k)$ *für alle Polynome* $p_k \in \mathcal{P}_k$ *vom Grad höchstens* k *gilt (und damit der Quadraturfehler verschwindet),*
- *ein Polynom* $p_{k+1} \in \mathcal{P}_{k+1}$ *vom Grad* $k+1$ *mit* $\hat{I}(p_{k+1}) \neq I(p_{k+1})$ *existiert.*

Bemerkung 6.3. *Wegen der Linearität des Integrals und der Quadraturformel ist die polynomiale Exaktheit vom Grad k äquivalent dazu, dass*

- $\hat{I}(x^\ell) = I(x^\ell)$ *für jedes* $\ell \in \{0, 1, \dots, k\}$, *aber*
- $\hat{I}(x^{k+1}) \neq I(x^{k+1})$ *gilt.*

Wir überlegen uns nun, wie sich die entsprechenden Gewichte in der interpolatorischen Quadraturformel in Definition 6.1 berechnen lassen:

Lemma 6.4 (Gewichte interpolatorischer Quadraturformeln)**.** *Ist $\Delta = \{x_0, x_1, \dots, x_n\}$ ein Gitter des Intervalls $[a, b]$, so ist die dazugehörige interpolatorische Quadraturformel von der Form*

$$\hat{I}_n(f) = (b-a) \sum_{j=0}^{n} \lambda_j^{\text{int}} f(x_j) \quad \text{für } f \in C([a,b])$$

mit Gewichten

$$\lambda_j^{\text{int}} \coloneqq \frac{1}{b-a} \int_a^b \prod_{\substack{\ell=0 \\ \ell \neq j}}^{n} \frac{x - x_\ell}{x_j - x_\ell} \, dx = \frac{1}{b-a} \int_a^b L_j(x) \, dx \quad \text{für } j \in \{0, 1, \dots, n\},$$

wobei $L_0, L_1, \dots, L_n$ die Lagrange-Polynome aus (5.2) bezeichnen.

Beweis. Das Interpolationspolynom durch die Punkte $(x_0, f(x_0)), \dots, (x_n, f(x_n))$ in der Darstellung mit Lagrange-Polynomen ist gegeben durch

$$p_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x).$$

Damit folgt nach den Definitionen der interpolatorischen Quadraturformel und der Gewichte $\lambda_0^{\text{int}}, \dots, \lambda_n^{\text{int}}$ die Behauptung über

$$\hat{I}_n(f) = \int_a^b p_n(x) \, dx = \sum_{j=0}^{n} f(x_j) \int_a^b L_j(x) \, dx = (b-a) \sum_{j=0}^{n} \lambda_j^{\text{int}} f(x_j). \qquad \square$$

Bemerkungen 6.5.

(1) *Ist der Integrand f ein Polynom vom Grad höchstens $n \in \mathbb{N}_0$, so gilt $f = p_n$ und damit $I(f) = \hat{I}_n(f)$. Folglich besitzt die interpolatorische Quadraturformel $\hat{I}_n$ mindestens den polynomialen Exaktheitsgrad n.*

(2) *Besitzt umgekehrt eine Quadraturformel $\hat{I}_n$ mit $(n+1)$ paarweise verschiedenen Quadraturpunkten $x_0, x_1, \dots, x_n$ mindestens den polynomialen Exaktheitsgrad n, so werden insbesondere alle Lagrange-Polynome $L_0, L_1, \dots, L_n$ als Polynome vom Grad n exakt integriert. Damit folgt über $L_j(x_\ell) = \delta_{j\ell}$ für $j, \ell \in \{0, 1, \dots, n\}$ dann*

$$(b-a)\lambda_j = \hat{I}_n(L_j) = I(L_j) = \int_a^b L_j(x) \, dx,$$

so dass die Quadraturformel tatsächlich interpolatorisch ist.

(3) *Unabhängig von der Wahl der Knoten ist die Funktion $\sum_{j=0}^{n} L_j$ ein Polynom vom Grad n, das in allen $(n+1)$ Stützstellen den Wert 1 annimmt. Da ein Polynom vom Grad höchstens n eindeutig durch $(n+1)$ Werte an paarweise verschiedenen Stützstellen festgelegt ist, muss damit $\sum_{j=0}^{n} L_j \equiv 1$ gelten. Für die Gewichte ergibt sich daher die Normierungsbedingung*

$$\sum_{j=0}^{n} \lambda_j^{\text{int}} = \frac{1}{b-a} \int_a^b \sum_{j=0}^{n} L_j(x) \, dx = \frac{1}{b-a} \int_a^b 1 \, dx = 1.$$

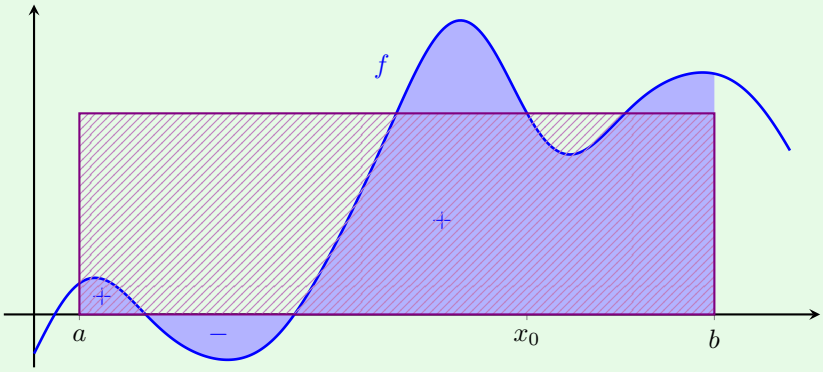
Alternativ kann man über die polynomiale Exaktheit vom Grad 0 *argumentieren: da die konstante Funktion* 1 *exakt integriert wird, ergibt sich*

$$(b-a) = I(1) = \hat{I}_n(1) = (b-a)\sum_{j=0}^{n} \lambda_j^{\text{int}} \quad \Rightarrow \quad \sum_{j=0}^{n} \lambda_j^{\text{int}} = 1.$$

Prinzipiell können einzelne Gewichte dabei negativ werden, was zu Problemen für die numerische Stabilität führen kann (nämlich Auslöschungsphänomene auch für rein positive Integranden). In jedem Fall hängen die Gewichte der interpolatorischen Quadraturformel nach der Formel in Lemma 6.4 *aber nur von der relativen Verteilung der Knoten in* $[a,b]$ *ab.*

Beispiel 6.6 (Rechteck- und Mittelpunktsregel)**.** Die einfachste interpolatorische Quadraturformel ergibt sich für den Fall mit nur einem Gitterpunkt x_0 im Intervall $[a,b]$, für den das Gewicht nach Bemerkung 6.5 (3) dann zwangsläufig gleich 1 sein muss. Entsprechend erhalten wir dann die sogenannte *Rechteckregel*

$$\hat{I}_0(f) = (b-a)f(x_0) \quad \text{für } f \in C([a,b]).$$



Für jede Wahl von $x_0 \in [a,b]$ werden damit Konstanten (also Polynome vom Grad 0) exakt integriert, und für das lineare Monom $p_1 \colon x \mapsto x$ gelten bei exakter Integration bzw. bei Auswertung der Quadraturformel

$$I(p_1) = \int_a^b x\,dx = \frac{b^2-a^2}{2} = (b-a)\frac{b+a}{2} \quad \text{und} \quad \hat{I}_0(p_1) = (b-a)x_0.$$

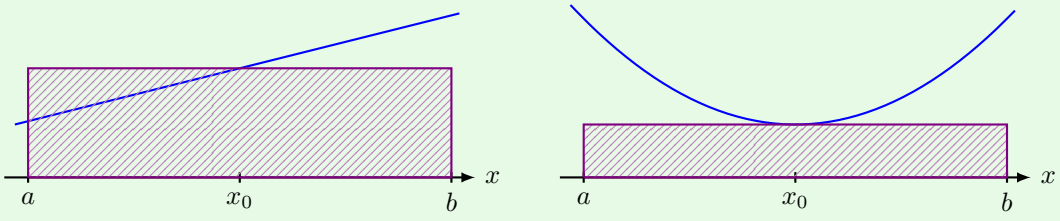
Exakte Integration des Monoms p_1 gilt folglich genau dann, wenn als Gitterpunkt gerade der Mittelpunkt $x_0 = \frac{a+b}{2}$ des Intervalls $[a,b]$ gewählt wird. Die entsprechende Quadraturformel wird dann als *Mittelpunktsregel* bezeichnet. In diesem Fall gilt für das quadratische Monom $p_2 \colon x \mapsto x^2$

$$I(p_2) = \int_a^b x^2\,dx = \frac{b^3-a^3}{3} \quad \text{und} \quad \hat{I}_0(p_2) = (b-a)\frac{(a+b)^2}{4} = \frac{b^3+b^2a-ba^2-a^3}{4},$$

so dass der Quadraturfehler gerade

$$I(p_2) - \hat{I}_0(p_2) = \frac{1}{12}\left(b^3 - 3b^2a + 3ba^2 - a^3\right) = \frac{(b-a)^3}{12}$$

ist. Die Mittelpunktsregel besitzt damit den polynomialen Exaktheitsgrad 1.



Newton–Cotes-Formeln. Für *äquidistante Stützstellen* des Gitters bezeichnen wir die interpolatorischen Quadraturformeln als *Newton–Cotes-Formeln* und die zugehörigen, in Lemma 6.4 berechneten Gewichte (die nun universell gegeben sind) als *Newton–Cotes-Gewichte*. Bei den sogenannten *abgeschlossenen* Formeln sind die Endpunkte $\{a, b\}$ Gitterpunkte, und die Stützstellen sind damit gegeben durch

$$x_j = a + \frac{(b-a)j}{n} \quad \text{für } j \in \{0, 1, \dots, n\}.$$

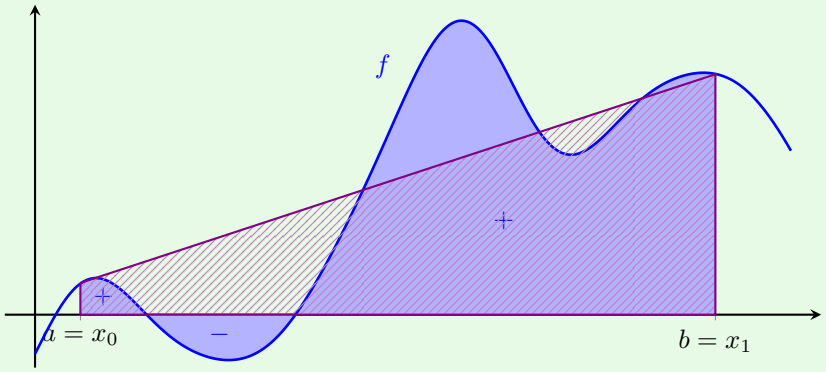
Beispiel 6.7 (Trapezregel)**.** Die einfachste abgeschlossene Newton–Cotes-Formel ergibt sich für $n = 1$ mit nur den beiden Endpunkten $\{a, b\}$ als Gitterpunkten. Als Gewichte berechnen wir gemäß der Formel in Lemma 6.4

$$\lambda_0^{\text{int}} \coloneqq \frac{1}{b-a}\int_a^b \frac{x-b}{a-b}\,dx = \frac{1}{b-a}\,\frac{-(a-b)^2}{2(a-b)} = \frac{1}{2} = \lambda_1^{\text{int}}.$$

Die so erhaltene Quadraturformel

$$\hat{I}_1(f) = \frac{b-a}{2}\big(f(a) + f(b)\big) \quad \text{für } f \in C([a,b])$$

wird als *Trapezregel* bezeichnet.

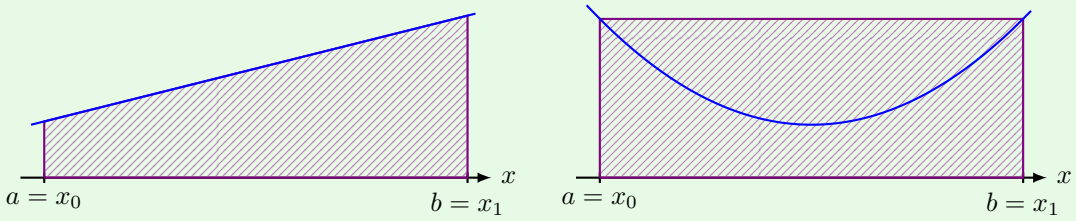


Als interpolatorische Quadraturformel mit zwei Gitterpunkten x_0, x_1 besitzt die Trapezregel nach Bemerkung 6.5 (1) mindestens den polynomialen Exaktheitsgrad 1. Dass der polynomiale Exaktheitsgrad genau 1 ist, ergibt sich dann einfach analog wie für die Mittelpunktsregel in Beispiel 6.6. Für das quadratische Monom $p_2 \colon x \mapsto x^2$ gilt hier

$$I(p_2) = \int_a^b x^2\,dx = \frac{b^3 - a^3}{3} \quad \text{und} \quad \hat{I}_1(p_2) = \frac{b-a}{2}(b^2 + a^2) = \frac{b^3 - b^2a + ba^2 - a^3}{2}.$$

Damit ist der Quadraturfehler in diesem Fall

$$I(p_2) - \hat{I}_0(p_2) = \frac{1}{6}\Big(-b^3 + 3b^2a - 3ba^2 + a^3\Big) = -\frac{(b-a)^3}{6}.$$



Der polynomiale Exaktheitsgrad ist damit der gleiche wie bei der Mittelpunktsregel, der Quadraturfehler für die Funktion p_2 betragsmäßig jedoch doppelt so groß. Allgemeiner kann man sich für konvexe Funktionen überlegen, dass der Quadraturfehler bei der Trapezregel immer größer als bei der Mittelpunktsregel ist, wobei das exakte Integral bei der Trapezregel überschätzt und bei der Mittelpunktsregel unterschätzt wird.

Beispiele 6.8. Wir sammeln hier nun die Gewichte, Stützstellen (bezogen auf das Einheitsintervall $[0,1]$) und polynomiale Exaktheitsgrade (mit Fehlerabschätzung für die Größe $|I(f) - \hat{I}_n(f)|$ unter der Annahme hinreichender Regularität an den Integranden) für einige Newton–Cotes-Formeln:

n	Name	Stützstellen	Gewichte	Exaktheitsgrad (Fehler)
1	Trapezregel	$0, 1$	$\frac{1}{2}, \frac{1}{2}$	1 $(\leq \frac{1}{12}\|f''\|_{\infty;[0,1]})$
2	Simpsonregel	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	3 $(\leq \frac{1}{2880}\|f^{(4)}\|_{\infty;[0,1]})$
3	3/8-Regel	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	3 $(\leq \frac{1}{6480}\|f^{(4)}\|_{\infty;[0,1]})$
4	Milne-Regel	$0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	5 $(\leq \frac{1}{1935360}\|f^{(6)}\|_{\infty;[0,1]})$

Bemerkungen 6.9 (Fazit für die interpolatorische Quadratur)**.**

(1) *Für eine kleine Anzahl an Stützstellen und für kleine Intervalle hat man numerische Stabilität und gute Fehlerabschätzungen.*

(2) *Für eine große Anzahl an Stützstellen ist die Polynominterpolation für den Integranden bereits numerisch kritisch (vgl. Rungephänomen für äquidistante / schlechte Wahl der Stützstellen). Zusätzlich sind die interpolatorischen Quadraturformeln wegen des möglichen Auftretens negativer Gewichte (bei den abgeschossenen Newton–Cotes Formeln etwa bereits ab $n \geq 8$) numerisch nicht mehr stabil.*

Summierte Quadraturformeln. Eine Lösungsmöglichkeit für die in Bemerkung 6.9 (2) angesprochenen Problematiken ist es, das Integrationsintervall $[a, b]$ in kleinere Teilintervalle zu zerlegen, dort jeweils interpolatorische Quadraturformeln (wie die Newton–Cotes-Formeln) niedrigerer Ordnung anzuwenden und die Ergebnisse auf den Teilintervallen am Ende zu summieren. Bei diesem Vorgehen spricht man dann von *summierten Quadraturformeln.*

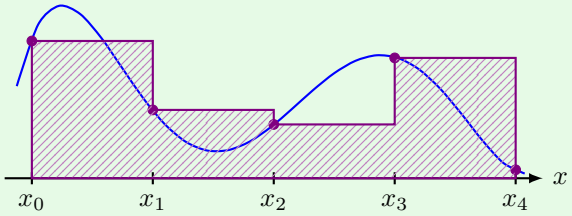
Beispiele 6.10. Wir geben nun die summierten Quadraturformeln für die Rechtecksregel und die Trapezregel für eine äquidistante Stützstellenwahl mit Stützstellenabstand $h := \frac{b-a}{n}$ an.

(i) *Summierte (linksseitige) Rechteckregel*: auf jedem Einzelintervall $[x_j, x_{j+1}]$ mit $j \in \{0, 1, \ldots, n-1\}$ approximiert man nach Beispiel 6.6

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

und erhält als summierte Quadraturformel dann

$$\hat{I}_n^{(0)}(f) = \sum_{j=0}^{n-1} (x_{j+1} - x_j) f(x_j) = h \sum_{j=0}^{n-1} f(x_j).$$

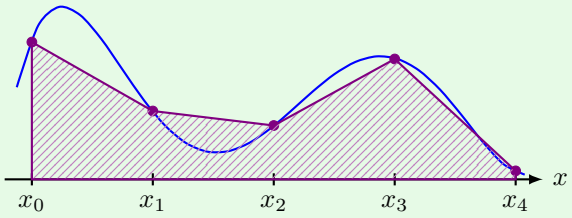


(ii) *Summierte Trapezregel*: auf jedem Einzelintervall $[x_j, x_{j+1}]$ mit $j \in \{0, 1, \ldots, n-1\}$ approximiert man nach Beispiel 6.7

$$\int_{x_j}^{x_{j+1}} f(x)\,dx \approx \frac{x_{j+1} - x_j}{2} \big(f(x_j) + f(x_{j+1})\big)$$

und erhält als summierte Quadraturformel dann

$$\hat{I}_n^{(1)}(f) = \sum_{j=0}^{n-1} \frac{x_{j+1} - x_j}{2} \big(f(x_j) + f(x_{j+1})\big) = \frac{h}{2} f(a) + h \sum_{j=1}^{n-1} f(x_j) + \frac{h}{2} f(b).$$



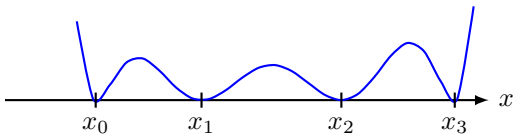
Gauß-Quadratur. Bei der interpolatorischen Quadratur haben wir zu *gegebenen* paarweise verschiedenen Stützstellen $x_0, x_1, \ldots, x_n$ eine Quadraturformel hergeleitet, bei der die Gewichte eindeutig gemäß Lemma 6.4 bestimmt waren. Diese interpolatorische Quadraturformeln $\hat{I}_n$ waren dann nach Konstruktion mindestens polynomial exakt vom Grad n. Bei der Gauß-Quadratur soll die *Lage der Stützstellen* so optimiert werden, dass die *polynomiale Exaktheit möglichst groß* ist.

Lemma 6.11 (maximaler polynomialer Exaktheitsgrad)**.** *Sind $x_0, x_1, \ldots, x_n$ für ein $n \in \mathbb{N}_0$ paarweise verschiedene Stützstellen in einem Intervall $[a, b]$, so besitzt jede zugehörige Quadraturformel $\hat{I}_n$ höchstens den polynomialen Exaktheitsgrad $2n + 1$.*

★ *Beweis.* Es ist ausreichend $I(p) \neq \hat{I}_n(p)$ für ein Polynom p vom Grad $2n + 2$ zu zeigen. Wir weisen dies für die spezielle Wahl

$$p(x) \coloneqq \prod_{j=0}^{n} (x - x_j)^2 \quad \text{für } x \in \mathbb{R}$$

nach. Diese Funktion ist offensichtlich ein Polynom vom Grad $2n + 2$ und erfüllt $p(x_j) = 0$ für alle $j \in \{0, 1, \ldots, n\}$ sowie $p(x) > 0$ für alle $x \in \mathbb{R} \setminus \{x_0, x_1, \ldots, x_n\}$.



Nach Definition der Quadraturformel mit den Stützstellen $x_0, x_1, \ldots, x_n$ und wegen der Stetigkeit von p gilt folglich

$$\hat{I}_n(p) = (b - a) \sum_{j=0}^{n} \lambda_j p(x_j) = 0 < \int_a^b p(x)\, dx = I(p).$$

Damit haben wir insbesondere gezeigt, dass das Integral $I(p)$ und die Quadraturformel $\hat{I}_n(p)$ nicht übereinstimmen, also dass für die Quadraturformel $\hat{I}_n$ keine polynomiale Exaktheit vom Grad $2n + 2$ gelten kann. □

Sind die Stützstellen einmal gewählt, so sind die Gewichte der interpolatorischen Quadraturformel eindeutig durch Lemma 6.4 bestimmt, so dass die Schwierigkeit der Gauß-Quadratur also nur in der Bestimmung von optimalen Stützstellen liegt. Da diese jedoch *nichtlinear* in die Quadraturformel eingehen, ist es a priori nicht klar, ob man pro Stützstelle tatsächlich einen polynomialen Exaktheitsgrad gewinnen kann. Für $n = 0$ besitzt die Mittelpunktsregel (im Gegensatz zu jeder anderen Rechtecksregel) den maximalen polynomialen Exaktheitsgrad $1 = 2 \cdot 0 + 1$ (vgl. Beispiel 6.6). Für allgemeine $n \in \mathbb{N}$ ist die Konstruktion solcher Gauß-Quadraturformeln ebenfalls möglich: die sogenannten *Legendre-Polynome* $\{\ell_{n+1}\}_{n \in \mathbb{N}}$ haben die Eigenschaft, dass ℓ_{n+1} für $n \in \mathbb{N}$ ein Polynom vom Grad $n + 1$ ist, das $(n + 1)$ paarweise verschiedene Nullstellen im Intervall $[-1, 1]$ hat. Bildet man die Quadraturformel mit diesen Nullstellen als Stützstellen (oder entsprechend transformierten Stützstellen für allgemeine Intervalle $[a, b]$), so wird der maximale polynomiale Exaktheitsgrad $2n + 1$ tatsächlich erreicht. Eine solche Konstruktion von Gauß-Quadraturformeln ist allgemeiner auch für uneigentliche Integrale und mit einer

positiven Gewichtsfunktion möglich. Als zusätzlicher numerischer Vorteil neben dem höheren Exaktheitsgrad gilt dabei, dass die Gewichte stets positiv sind und damit *keine Auslöschung* auftritt:

Lemma 6.12 (Positivität der Gewichte bei polynomialem Exaktheitsgrad $n+1$)**.** *Es seien $x_0, x_1, \ldots, x_n$ paarweise verschiedene Stützstellen in $[a,b]$. Ist die Quadraturformel polynomial exakt vom Grad $2n+1$, so sind die Gewichte stets positiv und es gilt die Darstellung*

$$\lambda_j = \frac{1}{b-a}\int_a^b L_j(x)\,dx = \frac{1}{b-a}\int_a^b L_j^2(x)\,dx \quad \text{für } j \in \{0,1,\ldots,n\}.$$

★ *Beweis.* Wegen $L_j(x_\ell) = \delta_{j\ell}$ für $j,\ell \in \{0,1,\ldots,n\}$ und der polynomialen Exaktheit der Quadraturformel $\hat{I}_n$ vom Grad $2n+1$ (und damit auch vom Grad $2n$, von dem die Funktionen L_j^2 sind) gilt für $j \in \{0,1,\ldots,n\}$

$$\lambda_j = \sum_{\ell=0}^{n} \lambda_\ell L_j^2(x_\ell) = \frac{1}{b-a}\hat{I}_n(L_j^2) = \frac{1}{b-a}I(L_j^2) = \frac{1}{b-a}\int_a^b L_j^2(x)\,dx. \qquad \square$$

Kapitel 7

Gewöhnliche Differentialgleichungen

Bei gewöhnlichen Differentialgleichungen handelt es sich um Gleichungen, die eine oder mehrere (unbekannte) Funktionen in einer Variablen mit einigen ihrer Ableitungen in Relation setzt. Der Begriff *gewöhnlich* bezieht sich hierbei auf die Tatsache, dass die Funktionen lediglich durch *eine* unabhängige Variable beschrieben werden. Da über Differentialgleichungen insbesondere Änderungsverhalten von Größen miteinander in Verbindung gebracht werden können, lassen sich viele Phänomene der Natur-, Sozial- oder Wirtschaftswissenschaften (zumindest näherungsweise) in Form solcher Gleichungen beschreiben, etwa in der Physik (Bewegungsgesetze wie die Newtonschen Bewegungsgleichungen oder Schwingungsgleichungen), Astronomie (Himmelsmechanik), Biologie (Modelle für das Wachstum von Populationen, aus der Epidemiologie und der Genetik), Meteorologie (Wettermodellierung), Chemie (Zerfallsraten) und Ökonomie (Finanzmarktmodelle). In der Regel sind diese gewöhnlichen Differentialgleichungen nicht explizit lösbar (auch wenn man oftmals noch qualitative Aussagen treffen kann). Aus diesem Grund ist man auf numerische Verfahren angewiesen, die zumindest eine näherungsweise Lösung des Problems liefern können. Die grundsätzliche Idee ist dabei, das kontinuierliche Ausgangsproblem durch ein diskretisiertes Problem zu ersetzen, das man dann (oftmals mit Methoden aus den vorherigen Kapiteln) in endlich vielen Schritten auf einem Computer lösen kann. Wir beschränken uns hier auf *explizite Differentialgleichungen*, bei denen eine Formel für die höchste Ableitung als Funktion von der unabhängigen Variable sowie den niedrigeren Ableitungen vorgegeben wird (da man implizite Gleichungen unter geeigneten Annahmen nach dem höchsten Ordnungsterm auflösen kann). Für solche Differentialgleichungen führen wir zunächst den Lösungsbegriff ein.

Definition 7.1 (Lösung, Existenzintervall)**.** *Es sei $I \subset \mathbb{R}$ ein Intervall. Eine k-fach differenzierbare Funktion $u\colon I \to \mathbb{R}^N$ heißt* Lösung *der gewöhnlichen Differentialgleichung*

$$u^{(k)} = f(t, u, u', \dots, u^{(k-1)})$$

mit einer Strukturfunktion $f\colon D \to \mathbb{R}^N$ für eine Menge $D \subset \mathbb{R}^{1+kN}$, falls $(t, u(t), u'(t), \dots, u^{(k-1)}(t)) \in D$ mit

$$u^{(k)}(t) = f(t, u(t), \dots, u^{(k-1)}(t)) \quad \text{für alle } t \in I$$

erfüllt ist. Das Intervall I nennen wir das zur Lösung u gehörige Lösungs- *oder* Existenzintervall.

Wir betrachten im Folgenden nur gewöhnliche Differentialgleichungen *erster Ordnung*, bei denen Ableitungen der unbekannten Funktion nur bis zur ersten Ordnung auftreten (also den Fall $k = 1$ in Definition 7.1). Zu einer gegebenen Strukturfunktion $f\colon D \to \mathbb{R}^N$ mit $D \subset \mathbb{R}^{1+N}$ suchen wir also eine Funktion $u\colon I \to \mathbb{R}^N$ für ein Intervall $I \subset \mathbb{R}$ mit

$$u'(t) = f(t, u(t)) \quad \text{für alle } t \in I.$$

Tatsächlich kann man Gleichungen höherer Ordnung immer in ein System von Gleichungen erster Ordnung überführen – muss dabei aber als Preis zahlen, mit einer um die Ordnung multiplizierten Anzahl von Gleichungen und unbekannten Funktionen zu arbeiten.

Satz 7.2 (Reduktion der Ordnung)**.** *Jedes (explizite) System von gewöhnlichen Differentialgleichungen k-ter Ordnung*

$$u^{(k)} = f(t, u, u', \dots, u^{(k-1)}) \tag{7.1}$$

für eine unbekannte Funktion $u\colon I \to \mathbb{R}^N$ ist äquivalent zu einem (expliziten) System erster Ordnung, mit k-facher Anzahl an unbekannten Funktionen und k-facher Anzahl an Gleichungen:

$$\begin{cases} v_1' = & v_2 \\ & \vdots \\ v_{k-1}' = & v_k \\ v_k' = & f(t, v_1, v_2, \dots, v_k) \end{cases} \tag{7.2}$$

für unbekannte Funktionen $v_j\colon I \to \mathbb{R}^N$ für $j \in \{1, \dots, k\}$.

Beweisidee. Die wesentliche Idee hinter dieser Umformulierung ist der Zusammenhang $(v_1, \dots, v_k) \leftrightarrow (u, u', \dots, u^{(k-1)})$:

- Ist u eine Lösung zu (7.1), so lösen die Funktionen $(v_1, \dots, v_k) \coloneqq (u, u', \dots, u^{(k-1)})$ nach ihrer Definition (mit $v_j' = (u^{(j-1)})' = u^{(j)} = v_{j+1}$ für $j \in \{1, \dots k-1\}$) die ersten $(k-1)$ Gleichungen aus (7.2), und die letzte Gleichung folgt direkt aus (7.1).
- Ist umgekehrt $(u_1, \dots, u_k)$ eine Lösung zu (7.2), so setzt man $u \coloneqq v_1$. Anhand ersten $(k-1)$ Gleichungen aus (7.2) sieht man dann, dass u k-fach differenzierbar ist, und über die letzte Gleichung verifiziert man dann die Lösungseigenschaft für (7.1). □

Beispiele 7.3.

(i) Die gewöhnliche Differentialgleichung $u' = \lambda u$ besitzt für $\lambda \in \mathbb{R}$ die allgemeine Lösung $u(t) = u_0 \exp(\lambda(t - t_0))$ für beliebiges $u_0 \in \mathbb{R}$ (auf dem Existenzintervall $I = \mathbb{R}$).

(ii) Die gewöhnliche Differentialgleichung $u' = u^2$ besitzt die allgemeine Lösung $u(t) = ((t_0 - t) + \frac{1}{u_0})^{-1}$ für $t_0, u_0 \in \mathbb{R}$ mit $u_0 \neq 0$ (auf Intervallen, so dass dieser Ausdruck wohldefiniert ist).

Die Lösungseigenschaft dieser Funktionen lässt sich dabei einfach über die Berechnung der Ableitung verifizieren, und die Funktionen erfüllen dabei alle die Bedingung $u(t_0) = u_0$.

7.1 Existenz von Lösungen

Wir stellen hier kurz einige wichtige Resultate aus der Theorie der gewöhnlichen Differentialgleichungen zur Existenz von Lösungen zum *Anfangswertproblem* (*AWP*)

$$\begin{cases} u' &= f(t, u) \\ u(t_0) &= u_0 \end{cases} \tag{7.3}$$

mit gegebener Strukturfunktion $f\colon D \to \mathbb{R}^N$ auf einer offenen Menge $D \subset \mathbb{R} \times \mathbb{R}^N$ und mit Anfangsbedingung $(t_0, u_0) \in D$ vor, soweit diese für die numerische Behandlung relevant sind. Damit wir sinnvoll numerische Lösungen diskutieren können, muss zunächst die Existenz einer (exakten) Lösung $u\colon I \to \mathbb{R}^N$ auf einem geeigneten Intervall I um die Anfangszeit t_0 im Sinne von Definition 7.1 sichergestellt sein, die auch die Anfangsbedingung $u(t_0) = u_0$ erfüllt. Äquivalent kann das Anfangswertproblem (7.3) auch als Integralgleichung formuliert werden, wobei wir uns hierbei auf den einfachsten Fall $D = I \times \mathbb{R}^N$ beschränken. Diese Umformulierung als Integralgleichung hat den Vorteil, dass man formal lediglich nach einer stetigen (aber a posteriori dann differenzierbaren) anstelle einer a priori differenzierbaren Lösung sucht.

Satz 7.4 (Umformung als Integralgleichung)**.** *Es sei $N \in \mathbb{N}$, $I \subset \mathbb{R}$ ein Intervall mit $t_0 \in I$, $u_0 \in \mathbb{R}^N$ und $f\colon I \times \mathbb{R}^N \to \mathbb{R}^N$ eine stetige Funktion. Eine Funktion $u\colon I \to \mathbb{R}^N$ ist genau dann eine Lösung des Anfangswertproblems* (7.3)*, falls sie stetig ist und für jedes $t \in I$ die folgende Integralgleichung erfüllt:*

$$u(t) = u_0 + \int_{t_0}^{t} f(s, u(s))\, ds.$$

Beweis. Die Äquivalenz gilt sofort durch Integration der Differentialgleichung von t_0 bis t bzw. durch Differentiation der Integralgleichung nach dem Hauptsatz der Differential- und Integralrechnung. □

Unter einer *lokalen Lösung* verstehen wir eine Lösung auf einem Intervall der Form $[t_0 - \alpha, t_0 + \alpha]$ für eine positive Zahl $\alpha > 0$. Unter der Annahme der Stetigkeit der Strukturfunktion ist deren Existenz gewährleistet, und gilt sogar die stärkere Bedingung der Lipschitz-Stetigkeit bezüglich der Lösungsvariable, so ist diese auch eindeutig.

Satz 7.5 (Existenzsatz von Peano)**.** *Es sei $N \in \mathbb{N}$, $D \subset \mathbb{R}^{1+N}$ offen und $(t_0, u_0) \in D$. Zum Anfangswertproblem* (7.3) *mit stetiger Strukturfunktion $f\colon D \to \mathbb{R}^N$* existiert eine lokale Lösung.

Satz 7.6 (Existenz- und Eindeutigkeitssatz von Picard–Lindelöf)**.** *Es sei $N \in \mathbb{N}$, $D \subset \mathbb{R}^{1+N}$ offen und $(t_0, u_0) \in D$. Zum Anfangswertproblem* (7.3) *mit stetiger Strukturfunktion $f\colon D \to \mathbb{R}^N$, die Lipschitz-stetig bezüglich der Lösungsvariable ist, also die für jedes $(t, x) \in D$ eine Abschätzung der Form*

$$|f(t,x) - f(t,\tilde{x})| \le L|x - \tilde{x}| \quad \text{für alle } \tilde{x} \in \mathbb{R}^N \text{ mit } (t,\tilde{x}) \in D \text{ und } |x - \tilde{x}| < \delta$$

mit einer Konstante $L \ge 0$ und für ein $\delta > 0$ erfüllt, existiert eine eindeutige lokale Lösung.

★ *Beweisidee.* Fasst man die rechte Seite der Umformung des Anfangswertproblems (7.3) als Integralgleichung als $T(u)$ auf, wobei T auf dem Raum der stetigen Funktionen operiert, so stellen wir fest, dass Lösungen des Anfangswertproblems (7.3) genau die Fixpunkte von T sind. Zeigt man nun, dass T zumindest für kleine Zeitintervalle $|t - t_0|$ eine kontraktive Selbstabbildung ist, so folgt die Existenz und Eindeutigkeit einer Lösung des Anfangswertproblems durch die Anwendung des Banachschen Fixpunktsatzes 4.8. □

Bemerkungen 7.7.

(1) *Jede differenzierbare Funktion ist wegen des Mittelwertsatzes insbesondere auch Lipschitz-stetig. Mit Differenzierbarkeit der Strukturfunktion bezüglich der Lösungsvariable hat man daher ein einfach zu überprüfendes Kriterium für die Bedingung der Lipschitz-Stetigkeit.*

(2) *Man kann selbst für glatte Strukturfunktionen im Allgemeinen nicht erwarten, dass eine Lösung für alle Zeiten existiert, siehe Beispiel* 7.3 (ii).

(3) *Unter den gleichen Voraussetzungen wie beim Satz* 7.6 *von Picard–Lindelöf hat man auch die stetige Abhängigkeit von den Anfangsdaten (t_0, u_0) und der Strukturfunktion f, d.h. kleine Störungen in diesen Daten beeinflussen die zugehörige Lösung nur minimal (und in stetiger Weise). Damit ist das Anfangswertproblem* (7.3) *dann* wohlgestellt*, so dass man nach einem stabilen numerischen Lösungsalgorithmus suchen kann.*

Bei *numerischen Verfahren zur Lösung des Anfangswertproblems* (7.3) berechnet man die Werte der approximativen Lösung typischerweise auf einem diskreten Gitter

$$\Delta \coloneqq \{t_0 < t_1 < \ldots < t_n = T\} \quad \text{mit maximaler Gitterweite } h_\Delta \coloneqq \max\{t_j - t_{j-1} \colon j \in \{1, \ldots, n\}\}$$

des Intervalls $[t_0, T]$, d.h. anstelle der exakten Lösung $u(t_j)$ bestimmt man nur eine Näherung u_j mit *Gitterfehler* $u(t_j) - u_j$ für $j \in \{0, 1, \ldots, n\}$. Die Norm $\max\{|u(t_j) - u_j| \colon j \in \{0, 1, \ldots, n\}\}$ dieses

Gitterfehlers möchte man dann in Abhängigkeit von der maximalen Gitterweite h_Δ kontrollieren. Von *Konvergenz der Ordnung* $p > 0$ sprechen wir, wenn

$$\max\{|u(t_j) - u_j| \colon j \in \{0, 1, \dots, n\}\} = O(h_\Delta^p) \quad \text{bei } h_\Delta \to 0$$

(also für immer feiner werdende Gitter) gilt. Welche Konvergenzordnung gilt, hängt dabei in der Regel vom verwendeten numerischen Verfahren ab. Zur Konstruktion solcher Verfahren approximiert man typischerweise das Integral in der Integralformulierung des Anfangswertproblems aus Satz 7.4 zwischen zwei Gitterpunkten. Dabei gibt es grundsätzlich

- *Einschrittverfahren*: die Berechnung von u_{j+1} nutzt nur den letzten Wert u_j,
- *Mehrschrittverfahren*: die Berechnung von u_{j+1} nutzt die letzten $m \in \mathbb{N}_{\geq 2}$ Werte $u_{j-m+1}, \dots, u_j$.

Innerhalb dieser Verfahren unterscheiden wir noch

- *explizite Verfahren*: für u_{j+1} lässt sich direkt eine Formel zur Berechnung angeben,
- *implizite Verfahren*: u_{j+1} ist nur implizit gegeben, im einfachsten Fall muss hier noch ein lineares Gleichungssystem gelöst werden.

7.2 Euler-Verfahren

Die exakte Lösung erfüllt für alle Gitterpunkte nach Satz 7.4 die Integralgleichung

$$u(t_{j+1}) = u(t_j) + \int_{t_j}^{t_{j+1}} f(s, u(s))\, ds. \tag{7.4}$$

Beim Euler-Verfahren approximiert man das Integral auf der rechten Seite nun durch eine Rechtecksregel, siehe Beispiel 6.6. Da wir nur Werte von u in Gitterpunkten bestimmen wollen, kommt nur die linke und die rechte Rechtecksregel in Frage, also die Approximation des Integranden auf dem ganzen Intervall $[t_j, t_{j+1}]$ durch den konstanten Wert $f(t_j, u(t_j))$ bzw. $f(t_{j+1}, u(t_{j+1}))$. Bei beiden Verfahren wird damit für die Berechnung des Lösungswertes im neuen Gitterpunkt nur der vorherige Wert verwendet, so dass es sich jeweils um ein Einschrittverfahren handelt. Je nach Wahl des rechten oder linken Randpunktes ergibt sich aber entweder ein direktes oder ein indirektes Verfahren.

Verfahren 7.8 (Euler-Verfahren)**.** Es sei $f \colon D \to \mathbb{R}^N$ für eine offene Menge $D \subset \mathbb{R} \times \mathbb{R}^N$ und $(t_0, u_0) \in D$ eine Anfangsbedingung. Zur Bestimmung der numerischen Lösung des Anfangswertproblems (7.3) über das Euler-Verfahren folgt man, ausgehend vom Startwert $u_0 \in \mathbb{R}^N$, der folgenden Vorschrift:

(a) *Explizites Euler-Verfahren*: für jedes $j \in \mathbb{N}_0$ mit $(t_j, u_j) \in D$ setzt man

$$u_{j+1} \coloneqq u_j + (t_{j+1} - t_j) f(t_j, u_j).$$

(b) *Implizites Euler-Verfahren*: für jedes $j \in \mathbb{N}_0$ wählt man $u_{j+1} \in \mathbb{R}^N$ mit $(t_{j+1}, u_{j+1}) \in D$ und

$$u_{j+1} = u_j + (t_{j+1} - t_j) f(t_{j+1}, u_{j+1}).$$

Bemerkung 7.9 (Berechnungsaufwand des Euler-Verfahrens)**.** *Für das explizite Euler-Verfahren muss man nur eine Funktionsauswertung machen, wohingegen man beim impliziten Euler-Verfahren im Allgemeinen ein nichtlineares Gleichungssystem (etwa mit dem Newton-Verfahren) lösen muss. Dieser Mehraufwand ist für sogenannte* steife Differentialgleichungen*, bei denen die Jacobi-Matrix* $\nabla_u f(t, u)$ *einen Eigenwert* λ *mit* $\operatorname{Re}(\lambda) \ll 0$ *besitzt, durchaus gerechtfertigt, da bei solchen Problemen nur implizite Verfahren stabil sind (siehe Beispiel* 7.10*).*

Beispiel 7.10 (Vergleich explizites vs. implizites Euler-Verfahren)**.** Wir betrachten zu $\lambda \in \mathbb{R}$ das Anfangswertproblem

$$\begin{cases} u' &= \lambda u \\ u(0) &= 1 \end{cases}$$

mit exakter Lösung $u(t) = \exp(\lambda t)$ für $t \in \mathbb{R}$ und berechnen dazu auf dem äquidistanten Gitter mit Gitterpunkten $t_j := jh$ für $j \in \mathbb{N}_0$ und Schrittweite $h > 0$ die numerischen Lösungen mit dem expliziten und impliziten Euler-Verfahren.

(i) *Explizites Euler-Verfahren*: es gilt

$$u_{j+1} = u_j + h\lambda u_j = (1 + h\lambda)u_j$$

und damit wegen $u_0 = u(0) = 1$

$$u_{j+1} = (1 + h\lambda)^{j+1}.$$

(ii) *Implizites Euler-Verfahren*: es gilt

$$v_{j+1} = v_j + h\lambda v_{j+1} \quad \Leftrightarrow \quad (1 - h\lambda)v_{j+1} = v_j$$

und damit für $1 - h\lambda \neq 0$ wegen $v_0 = u(0) = 1$

$$v_{j+1} = (1 - h\lambda)^{-(j+1)}.$$

Für die exakte Lösung haben wir im Fall $\lambda < 0$ die Asymptotik

$$\lim_{t\to\infty} u(t) = \lim_{t\to\infty} \exp(\lambda t) = 0,$$

die auch für die numerischen Lösungen gelten sollten. Daher bestimmen wir nun deren Asymptotik bei $j \to \infty$ für feste Schrittweite $h > 0$. Für das explizite Euler-Verfahren gilt

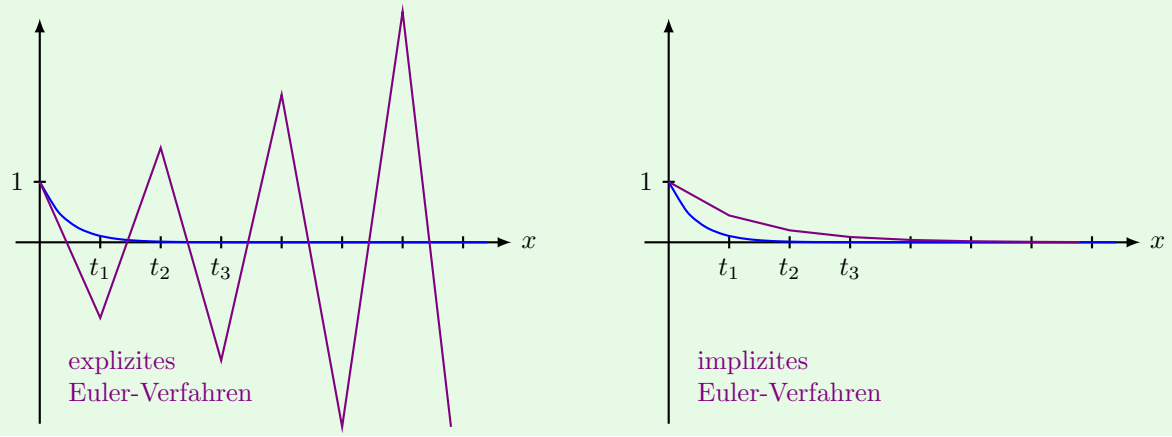
$$\lim_{j\to\infty} u_j = \lim_{j\to\infty} (1 + h\lambda)^j = 0$$

nur im Fall

$$|1 + h\lambda| < 1 \quad \overset{\lambda<0<h}{\Leftrightarrow} \quad h < -\frac{2}{\lambda},$$

wohingegen die Lösung für Schrittweiten $h \geq -\frac{2}{\lambda}$ oszilliert. Für $\lambda \ll 0$ hat man beim expliziten Euler-Verfahren daher an die Schrittweite sehr starke Restriktionen. Für das implizite Euler-Verfahren gilt für *jede* Schrittweite das korrekte asymptotische Verhalten

$$\lim_{j\to\infty} v_j = \lim_{j\to\infty} (1 - h\lambda)^{-j} = 0.$$



Bemerkung 7.11 (Konvergenzordnung des Euler-Verfahrens)**.** *Sowohl das explizite als auch das implizite Euler-Verfahren besitzen die* Konvergenzordnung $p = 1$*, d.h. sie erfüllen*

$$\max\{|u(t_j) - u_j| \colon j \in \{0, 1, \dots, n\}\} = O(h_\Delta) \quad \text{bei } h_\Delta \to 0.$$

Diese Abschätzung für den globalen Fehler *leitet man (unter der Generalvoraussetzung der Lipschitz-Stetigkeit der Strukturfunktion) aus einer Abschätzung für den* lokalen Fehler *her, also dem Fehler*

$$|u(t_1) - u_1| = \left| \int_{t_0}^{t_1} f(s, u(s))\, ds - (t_1 - t_0) f(t_0, u_0) \right| = O(h^2) \quad \text{für } h \coloneqq t_1 - t_0,$$

den man in einem Schritt macht. Die Heuristik dahinter ist, dass man bis zur Zeit t (bei einem äquidistanten Gitter) etwa $(t - t_0) \cdot \frac{1}{h}$ Schritte machen muss und sich dabei die lokalen Fehler aufsummieren, was dann zu einem globalen Fehler von der Größenordnung $\frac{1}{h} \cdot h^2 = h$ führt.

7.3 Einschrittverfahren höherer Ordnung

Indem man andere Quadraturformeln mit besseren Fehlerabschätzungen anstelle der Rechtecksregel für die Approximation des Integrals in der Integralgleichung (7.4) wählt, kann man Einschrittverfahren erhalten, die von höherer Ordnung im Vergleich zum Euler-Verfahren konvergieren. Approximieren wir das Integral mithilfe der Trapezregel

$$\int_{t_j}^{t_{j+1}} f(s, u(s))\, ds \approx \frac{t_{j+1} - t_j}{2} \big(f(t_j, u(t_j)) + f(t_{j+1}, u(t_{j+1})) \big),$$

die den polynomialen Exaktheitsgrad 1 besitzt, so führt dies unmittelbar auf ein implizites Verfahren.

Verfahren 7.12 (Trapez-Verfahren)**.** Es sei $f \colon D \to \mathbb{R}^N$ für eine offene Menge $D \subset \mathbb{R} \times \mathbb{R}^N$ und $(t_0, u_0) \in D$ eine Anfangsbedingung. Zur Bestimmung der numerischen Lösung des Anfangswertproblems (7.3) über das (implizite) Trapez-Verfahren wählt man, ausgehend vom Startwert $u_0 \in \mathbb{R}^N$, für jedes $j \in \mathbb{N}_0$ ein $u_{j+1} \in \mathbb{R}^N$ mit $(t_{j+1}, u_{j+1}) \in D$ und

$$u_{j+1} \coloneqq u_j + \frac{t_{j+1} - t_j}{2} \big(f(t_j, u_j) + f(t_{j+1}, u_{j+1}) \big).$$

Um ein explizites Verfahren zu erhalten, ersetzen wir im impliziten Teil $f(t_{j+1}, u_{j+1})$ aus dem Trapez-Verfahren die neue Iterierte durch einen expliziten Euler-Schritt, also

$$f(t_{j+1}, u_{j+1}) \approx f(t_{j+1}, u_j + (t_{j+1} - t_j) f(t_j, u_j)).$$

Dadurch erhalten wir ein explizites Verfahren, das wieder nur die Strukturfunktion und die alte Iterierte u_j zur Berechnung von u_{j-1} benötigt.

Verfahren 7.13 (Verfahren von Heun)**.** Es sei $f \colon D \to \mathbb{R}^N$ für eine offene Menge $D \subset \mathbb{R} \times \mathbb{R}^N$ und $(t_0, u_0) \in D$ eine Anfangsbedingung. Zur Bestimmung der numerischen Lösung des Anfangswertproblems (7.3) über das (explizite) Verfahren von Heun setzt man, ausgehend vom Startwert $u_0 \in \mathbb{R}^N$, für jedes $j \in \mathbb{N}_0$ mit $(t_j, u_j) \in D$

$$u_{j+1} \coloneqq u_j + \frac{t_{j+1} - t_j}{2} \big(f(t_j, u_j) + f(t_{j+1}, u_j + (t_{j+1} - t_j) f(t_j, u_j)) \big).$$

Verwenden wir statt der Trapezregel die Mittelpunktregel (die ebenfalls den polynomialen Exaktheitsgrad 1 besitzt) zur Approximation des Integrals

$$\int_{t_j}^{t_{j+1}} f(s, u(s))\, ds \approx (t_{j+1} - t_j) f\left(\frac{t_j + t_{j+1}}{2}, \frac{u(t_j) + u(t_{j+1})}{2} \right),$$

so führt dies auf ein weiteres implizites Verfahren.

Verfahren 7.14 (Mittelpunkt-Verfahren)**.** Es sei $f\colon D \to \mathbb{R}^N$ für eine offene Menge $D \subset \mathbb{R} \times \mathbb{R}^N$ und $(t_0, u_0) \in D$ eine Anfangsbedingung. Zur Bestimmung der numerischen Lösung des Anfangswertproblems (7.3) über das (implizite) Mittelpunkt-Verfahren wählt man, ausgehend vom Startwert $u_0 \in \mathbb{R}^N$, für jedes $j \in \mathbb{N}_0$ ein $u_{j+1} \in \mathbb{R}^N$ mit $((t_j + t_{j+1})/2, (u_j + u_{j+1})/2) \in D$ und

$$u_{j+1} \coloneqq u_j + (t_{j+1} - t_j) f\left(\frac{t_j + t_{j+1}}{2}, \frac{u_j + u_{j+1}}{2}\right).$$

Bemerkung 7.15 (Konvergenzordnung des Trapez-Verfahrens, des Verfahrens von Heun und des Mittelpunkt-Verfahrens)**.** *Alle drei Verfahren* 7.12*,* 7.13 *und* 7.14 *haben die* Konvergenzordnung $p = 2$*.*

Mit dem Ziel, numerische Verfahren zur Lösung des Anfangswertproblems (7.3) zu konstruieren, die von höherer Ordnung sind, kann man die ineinander geschachtelten Auswertungen der Strukturfunktion f wie beim Verfahren von Heun verallgemeinern. Bei einem *Runge–Kutta-Verfahren der Stufe* $m \in \mathbb{N}$ sind eine Matrix $A \in \mathbb{R}^{m \times m}$ (die sogenannte *Runge–Kutta-Matrix*) sowie zwei Vektoren $b, c \in \mathbb{R}^m$ (die sogenannten *Gewichte* und *Knoten*) gegeben, die üblicherweise verkürzt im *Butcher-Schema* geschrieben werden als

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}.$$

Da die Punkte $t + c_\ell h$ für $\ell \in \{1, \dots, m\}$ einem Gitter in $[t, t + h]$ entsprechen sollen, betrachten wir hier nur Vektoren $c \in \mathbb{R}^m$ deren Einträge aufsteigend geordnet und aus $[0, 1]$ sind. Man sucht dann "Steigungsvektoren" $k_1, \dots, k_m$ mit

$$k_\ell = f\Big(t_j + c_\ell h, u_j + h \sum_{i=1}^{m} a_{\ell i} k_i\Big) \quad \text{für } \ell \in \{1, \dots, m\}$$

(dies soll eine Approximation von $u'(t_j + c_\ell h)$ sein, die für die exakte Lösung gerade $f(t_j + c_\ell h, u(t_j + c_\ell h))$ ist) und setzt dann

$$u_{j+1} \coloneqq u_j + h \sum_{\ell=1}^{m} b_\ell k_\ell.$$

Die zuvor betrachteten Verfahren stellen sich nun tatsächlich als spezielle Runge–Kutta-Verfahren heraus. Wir haben etwa

- das explizite Euler-Verfahren mit $m = 1$ und

$$A = 0, \quad b = 1, \quad c = 0 \quad \Rightarrow \quad \text{Butcher-Schema} \quad \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array},$$

- das implizite Euler-Verfahren mit $m = 1$ und

$$A = 1, \quad b = 1, \quad c = 1 \quad \Rightarrow \quad \text{Butcher-Schema} \quad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}.$$

Dass das erste Butcher-Schema dem expliziten Euler-Verfahren entspricht, ist direkt aus der Definition des Runge–Kutta-Verfahren klar, denn es gilt $k_1 = f(t_j, u_j)$ und dann

$$u_{j+1} = u_j + h k_1 = u_j + h f(t_j, u_j).$$

Beim zweiten Butcher-Schema haben wir zunächst $k_1 = f(t_j + h, u_j + h k_1)$ und erhalten das implizite Euler-Verfahren dann über

$$u_{j+1} = u_j + h k_1 = u_j + h f(t_j + h, u_j + h k_1) = u_j + h f(t_j + h, u_{j+1}).$$

Auf analoge Weise identifiziert man die anderen Verfahren:

- Trapez-Verfahren mit $m = 2$ und

$$A = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad b = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \text{Butcher-Schema} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array},$$

- Verfahren von Heun mit $m = 2$ und

$$A = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \text{Butcher-Schema} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array},$$

- Mittelpunkt-Verfahren mit $m = 1$ und

$$A = \frac{1}{2}, \quad b = 1, \quad c = \frac{1}{2} \quad \Rightarrow \quad \text{Butcher-Schema} \quad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}.$$

Bemerkungen 7.16 (Parameterwahl für A, b, c)**.**

(1) *Insgesamt muss man m gekoppelte Gleichungssysteme (mit je N Gleichungen) für die Bestimmung der Vektoren $k_1, \ldots, k_m \in \mathbb{R}^N$ lösen. Dies vereinfacht sich zu sukzessive nacheinander ausgeführten Funktionsauswertungen, wenn die Matrix A eine* strikte untere Dreiecksmatrix *ist. In diesem Fall handelt es sich also um ein* explizites Runge–Kutta-Verfahren*, andernfalls um ein* implizites*. Die neue Approximation u_{j+1} ergibt sich im Anschluss in jedem Fall als Linearkombination der alten Approximation u_j sowie der Vektoren $k_1, \ldots, k_m$.*

(2) *Für explizite bzw. implizite Runge–Kutta-Verfahren hat man bei der Matrix $A \in \mathbb{R}^{m \times m}$ sowie den Vektoren $b, c \in \mathbb{R}^m$*

$$\frac{(m-1)m}{2} + m + m \quad \textit{bzw.} \quad m^2 + m + m \quad \textit{Freiheitsgrade},$$

die man nun so wählen möchte, dass die Konvergenzordnung möglichst hoch ist. Der Vektor c spielt dabei keine Rolle (wenn die Strukturfunktion f nicht explizit von der Zeitvariable abhängt, ist die Wahl nämlich egal), so dass

$$c_\ell = \sum_{i=1}^{m} a_{\ell i} \quad \textit{für } \ell \in \{1, \ldots, m\}$$

hier die kanonische Wahl ist. Für A und b dagegen stellt man entsprechende Bedingungsgleichungen auf und löst diese dann. Dies wird mit zunehmender Ordnung immer komplexer. Eine Bedingung an die Wahl des Vektors b ist dabei

$$\sum_{\ell=1}^{m} b_\ell = 1,$$

was sicherstellt, dass der Fehler in einem Schritt hinreichend klein ist (Konsistenz)*. Nur in diesem Fall bekommt man insbesondere für konstante Steigung der Lösung, also $f \equiv k$ für einen Vektor $k \in \mathbb{R}^N$, das korrekte Ergebnis für die approximative Lösung. Allgemein gilt, dass ein explizites m-stufiges Runge–Kutta-Verfahren höchstens die Konvergenzordnung m besitzt, ein implizites dagegen bis zu $2m$.*

Beispiel 7.17 (klassisches Runge–Kutta-Verfahren der Ordnung $p = 4$)**.** Das klassische Runge–Kutta-Verfahren (RK4) ist ein explizites Runge–Kutta-Verfahren der Stufe $m = 4$ und hat das Butcher-Schema

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

(wobei nicht eingetragene Einträge der Matrix A Nulleinträgen entsprechen).

Die in diesem Abschnitt vorgestellten Einschrittverfahren, bei denen also die Berechnung der Approximation u_{j+1} der Lösung beim Zeitschritt t_{j+1} nur auf der Approximation u_j (sowie f und den Zeitschritte) basiert, haben bei der numerischen Implementierung die folgenden Vor- und Nachteile:

- $\oplus$ hohe Flexibilität,
- $\oplus$ relativ einfache Implementierung,
- $\oplus$ einfache Steuerung der Schrittweite (so groß wie möglich, unter Einhaltung einer vorgegebenen Fehlerschranke, aber so klein wie nötig bei vertretbaren Aufwand für eine Kontrolle der Fortpflanzung von Fehlern),
- $\ominus$ Gedächtnislosigkeit (d.h. bereits gewonnene Informationen über die Strukturfunktion f werden außer für die Schrittweitensteuerung nicht weiter verwendet), und daraus resultierend
- $\ominus$ großer Aufwand bei den Funktionsauswertungen.

7.4 Lineare Mehrschrittverfahren

Ein lineares (m-) Mehrschrittverfahren für das Anfangswertproblem (7.3) wird festgelegt über die Angabe von

- m Startwerten $u_0, u_1 = u(t_1), \ldots, u_{m-1} = u(t_{m-1}) \in \mathbb{R}^N$ des Verfahren, sowie
- $2m + 2$ Koeffizienten $\alpha_0, \alpha_1, \ldots, \alpha_m \in \mathbb{R}$ und $\beta_0, \beta_1, \ldots, \beta_m \in \mathbb{R}$ mit $\alpha_0 = 1$

und der Rekursionsgleichung

$$u_{j+1} + \sum_{\ell=1}^{m} \alpha_\ell u_{j+1-\ell} = h \sum_{\ell=0}^{m} \beta_\ell f(t_{j+1-\ell}, u_{j+1-\ell}) \quad \text{für } j \in \mathbb{N} \text{ mit } j \geq m-1.$$

Bemerkungen 7.18.

(1) *Die Startwerte $u_1, \ldots, u_{m-1}$ sind nicht a priori vorgegeben, sondern sie werden in einer sogenannten Anlaufrechnung durch andere Näherungsverfahren wie etwa ein Einschrittverfahren bestimmt.*

(2) *Für $\beta_0 = 0$ handelt es sich um ein* explizites lineares Mehrschrittverfahren*, bei dem man die Rekursionsgleichung also nach der neu zu berechnenden approximativen Lösung u_{j+1} auflösen kann. Für $\beta_0 \neq 0$ dagegen ist das Verfahren* implizit *und in der Vorschrift für die Bestimmung von u_{j+1} kommt auch $f(t_{j+1}, u_{j+1})$ vor, so dass im Allgemeinen ein nichtlineares Gleichungssystem gelöst werden muss.*

(3) *Es gibt zur Konstruktion von linearen Mehrschrittverfahren verschiedene Möglichkeiten, etwa durch numerische Quadratur oder numerische Differentiation. Bei der Konstruktion über numerische Quadratur wird statt über die Strukturfunktion f über ein interpolierendes Polynom integriert, das durch mehrere alte Stützpunkte geht.*

Beispiel 7.19 (Verfahren von Adams–Bashforth)**.** Das Verfahren von Adams–Bashforth ist ein bekanntes explizites lineares Mehrschrittverfahren, bei dem das Integral in der Integralgleichung (7.4) approximiert wird, indem über das Interpolationspolynom durch die letzten m Stützpunkte $(t_{j-m+1}, f(t_{j-m+1}, u_{j-m+1})), \dots, (t_j, f(t_j, u_j))$ integriert wird. Im Fall $m = 2$ und äquidistante Stützstellen mit Abstand h ist dieses Interpolationspolynom vom Grad höchstens 1 und gegeben durch

$$t \mapsto f(t_j, u_j) + \frac{f(t_j, u_j) - f(t_{j-1}, u_{j-1})}{h}(t - t_j),$$

so dass wir als Approximation

$$\int_{t_j}^{t_{j+1}} f(s, u(s))\, ds \approx h f(t_j, u_j) + \frac{h}{2}\big(f(t_j, u_j) - f(t_{j-1}, u_{j-1})\big)$$

erhalten. Damit ergibt sich die Rekursionsgleichung

$$u_{j+1} = u_j + h\left(\frac{3}{2} f(t_j, u_j) - \frac{1}{2} f(t_{j-1}, u_{j-1})\right),$$

also als Koeffizienten im Zweischrittverfahren

$$\alpha_0 = 1,\ \alpha_1 = -1,\ \alpha_2 = 0 \quad \text{und} \quad \beta_0 = 0,\ \beta_1 = \frac{3}{2},\ \beta_2 = -\frac{1}{2}.$$

Man kann zeigen, dass dieses Verfahren die *Konvergenzordnung* $p = 2$ besitzt.

Kapitel 8

Partielle Differentialgleichungen

Partielle Differentialgleichungen sind Gleichungen, bei denen eine oder mehrere unbekannte Funktionen in mehreren Variablen über einige ihrer partiellen Ableitungen in Beziehung stehen. Mit solchen Differentialgleichungen lassen sich insbesondere fundamentale physikalische Gesetze beschreiben. Einige Beispiele solcher klassischen Modellgleichungen (mit $x \in \Omega \subset \mathbb{R}^d$ der Ortsvariable für $d \in \{2, 3\}$ für typische Anwendungen und $t \in \mathbb{R}$ der Zeitvariable) sind etwa:

- Die *Poissongleichung*
$$-\Delta u \coloneqq -\sum_{i=1}^{d} \partial_{x_i}^2 u = f$$
beschreibt stationäre, also zeitunabhängige Gleichgewichtszustände, beispielsweise von Ladungsverteilungen. Eine Herleitung dieser partiellen Differentialgleichung über das Prinzip der Massenerhaltung sowie einen linearen Zusammenhang zwischen der Flussdichte und dem Gradienten haben wir bereits in der Einleitung gesehen.

- Die *Wärmeleitungsgleichung*
$$\partial_t u - \Delta u = f$$
modelliert die (zeitliche und räumliche) Veränderung der Temperaturverteilung in einem Körper oder Raum, unter dem Einfluss einer Wärmequelle f.

- Die *Wellengleichung*
$$\partial_t^2 u - \Delta u = 0$$
beschreibt allgemein Schwingungsvorgänge in elastischen Körpern, wie etwa die Ausbreitung von Schall-, Licht- und Wasserwellen.

- Die *Navier–Stokes-Gleichungen*
$$\begin{cases} \partial_t u + (u \cdot \nabla)u - \Delta u = -\nabla p \\ \operatorname{div} u = 0 \end{cases}$$
erfassen die (zeitliche und räumliche) Bewegung von Fluiden und Gasen.

Für die eindeutige Lösbarkeit verlangt man typischerweise noch *Randbedingungen* im Ort (also für $x \in \partial\Omega$), wie etwa vorgeschriebene Funktionswerte, und falls das Problem zeitabhängig ist, dann auch noch *Anfangsbedingungen* zu einer fixierten Startzeit, wie etwa eine Anfangsverteilung (und bei der Wellengleichung noch zusätzlich deren Zeitableitung). Für die numerische Behandlung partieller Differentialgleichungen muss man typischerweise diskretisieren. Wir stellen hier kurz einige Ideen der meistgenutzten Verfahren vor.

8.1 Finite-Differenzen-Methode

Die zentrale Idee von Finite-Differenzen-Verfahren besteht darin, die exakte Lösung einer partiellen Differentialgleichung nur an diskreten Punkten zu approximieren, indem die in der Differentialgleichung auftretenden Ableitungen durch Differenzenquotienten ersetzt werden. Dadurch wird die (kontinuierliche) partielle Differentialgleichung in ein System algebraischer Gleichungen übergeführt, das dann (unter geeigneten Bedingungen) gelöst werden kann. Für die Differenzenquotienten gibt es dabei mehrere (problemabhängige) Möglichkeiten, etwa die Differenzenquotienten vorwärts oder rückwärts zu bilden.

Beispiel 8.1 (Differenzenquotienten in einer Dimension)**.** Für eine differenzierbare Funktion $u\colon \mathbb{R} \to \mathbb{R}$ und eine (kleine) positive Zahl $h > 0$ gilt etwa für die erste Ableitung

$$u'(x) \approx \frac{u(x \pm h) - u(x)}{\pm h},$$

wobei der Approximationsfehler im Limes $h \to 0$ verschwindet. Diese Approximation bezeichnet man üblicherweise für das positive Vorzeichen als *vorwärts* bzw. für das negative Vorzeichen als *rückwärts gebildeten Differenzenquotienten*. Entsprechend gilt unter der Annahme der zweifachen Differenzierbarkeit durch Approximation der ersten Ableitung über einmal vorwärts und einmal rückwärts gebildete Differenzenquotienten (die Reihenfolge spielt dabei keine Rolle)

$$\begin{aligned} u''(x) &\approx \frac{u'(x+h) - u'(x)}{h} \\ &\approx \frac{\frac{u(x+h-h)-u(x+h)}{-h} - \frac{u(x-h)+u(x)}{-h}}{h} = \frac{u(x+h) - 2u(x) - u(x-h)}{h^2}. \end{aligned}$$

Dieses Vorgehen lässt sich auch auf Funktionen auf $\mathbb{R}^d$ in mehreren Raumdimensionen verallgemeinern, um dann entsprechende Approximationen der (partiellen) Ableitungen zu erhalten.

Finite-Differenzen-Methode für die Poissongleichung. Wir betrachten die Poissongleichung in einer Modellsituation auf dem offenen zwei-dimensionalen Einheitswürfel $\Omega = (0,1)^2$, also

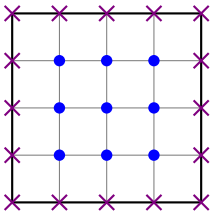
$$-\Delta u = f \quad \text{in } \Omega \tag{8.1}$$

mit der Randbedingung

$$u = g \quad \text{auf } \Gamma \coloneqq \partial\Omega, \tag{8.2}$$

für gegebene Funktionen $f\colon \Omega \to \mathbb{R}$ und $g\colon \Gamma \to \mathbb{R}$. Das Gebiet Ω und seinen Rand Γ diskretisieren wir nun über ein äquidistantes Gitter

$$\begin{aligned} \Omega_h &\coloneqq \big\{x_{j,\ell} \coloneqq (jh, \ell h)\colon j, \ell \in \{1, \dots, N-1\}\big\}, \\ \Gamma_h &\coloneqq \big\{x_{j,\ell} \coloneqq (jh, \ell h)\colon j \text{ oder } \ell \in \{0, N\}\big\}, \end{aligned}$$



mit Schrittweite $h \coloneqq \frac{1}{N}$ für ein $N \in \mathbb{N}$.

Anstelle der kontinuierlichen Lösung $u\colon \overline{\Omega} \to \mathbb{R}$ der Poissongleichung (8.1) mit den Randwerten (8.2) suchen wir nun eine approximative Lösung, nämlich eine Gitterfunktion $u_h\colon \Omega_h \cup \Gamma_h \to \mathbb{R}$, die eine diskrete Version dieses Problems löst. Diskretisieren wir jede der partiellen Ableitung $\partial^2_{x_1} u$ und $\partial^2_{x_2} u$ wie in Beispiel 8.1, so erhalten wir den *diskreten Laplace-Operator* Δ_h, der auf die Funktion u_h über

$$\begin{aligned} \Delta_h u_h(jh, \ell h) \coloneqq \frac{1}{h^2}\big(&-4u_h(jh, \ell h) + u_h((j+1)h, \ell h) + u_h((j-1)h, \ell h) \\ &+ u_h(jh, (\ell+1)h) + u_h(jh, (\ell-1)h)\big) \end{aligned}$$

für $j, \ell \in \{1, \dots, N-1\}$ wirkt. Dies kann man alternativ auch über den *Differenzenstern* oder 5-*Punkt-Stern* als

$$\Delta_h u_h(jh, \ell h) = \frac{1}{h^2}\left(\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} u_h \right)(jh, \ell h)$$

darstellen, wobei die Einträge in der ersten und letzten Spalte der Matrix für um eines nach rechts bzw. links verschobene x_1-Argumente und die Einträge in der ersten und letzten Zeile für um eines nach oben oder unten verschobene x_2-Argumente stehen. Diskretisieren wir noch die rechte Seite f sowie die vorgegebenen Randwerte g in den Gitterpunkten aus Ω_h bzw. Γ_h über

$$f_h(jh, \ell h) \coloneqq f(jh, \ell h) \quad \text{für } (jh, \ell h) \in \Omega_h$$

und

$$g_h(jh, \ell h) \coloneqq g(jh, \ell h) \quad \text{für } (jh, \ell h) \in \Gamma_h,$$

so erhalten wir als Diskretisierung der Poissongleichung (8.1)

$$-\Delta_h u_h(jh, \ell h) = f_h(jh, \ell h) \quad \text{für } (jh, \ell h) \in \Omega_h, \tag{8.3}$$

und anstelle der Randbedingung (8.2) fordern wir

$$u_h(jh, \ell h) = g_h(jh, \ell h) \quad \text{für } (jh, \ell h) \in \Gamma_h. \tag{8.4}$$

Die Werte der diskreten Lösung u_h auf den $4N$ Randpunkten auf Γ_h sind über (8.4) eindeutig festgelegt, und (8.3) entspricht einem Gleichungssystem mit $(N-1)^2$ Gleichungen in den $(N-1)^2$ Unbekannten der inneren Punkte in Ω_h. Jede Gleichung involviert dabei jeweils höchstens fünf unbekannte Werte.

Satz 8.2 (eindeutige Lösbarkeit der diskreten Poissongleichung)**.** *Die diskrete Poissongleichung* (8.3) *mit der Randbedingung* (8.4) *besitzt eine eindeutige Lösung* $u_h \colon \Omega_h \cup \Gamma_h \to \mathbb{R}$.

Bemerkung 8.3. *Im Fall von Nullrandwerten* $g = 0$ *und damit* $g_h = 0$ *handelt es sich bei* (8.3) *um ein Gleichungssystem, bei dem nur die (unbekannten) Gitterwerte der inneren Punkte* Ω_h *vorkommen: mit lexikographischer Nummerierung (also zeilenweise von oben nach unten und innerhalb jeder Zeile von links nach rechts) hat die Matrix* $A_h \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$ *des resultierenden Gleichungssystems dabei die Form*

$$A_h = \frac{1}{h^2}\begin{pmatrix} B & -I & O & \cdots & O \\ -I & B & -I & \ddots & \vdots \\ O & -I & B & \ddots & O \\ \vdots & \ddots & \ddots & \ddots & -I \\ O & \cdots & O & -I & B \end{pmatrix} \quad \textit{mit} \quad B = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & -1 & 4 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 4 \end{pmatrix}$$

sowie der Einheitsmatrix $I \coloneqq \mathbb{1}_{(N-1)\times(N-1)} \in \mathbb{R}^{(N-1)\times(N-1)}$ *und der Nullmatrix* $O \coloneqq \mathbb{O}_{(N-1)\times(N-1)} \in \mathbb{R}^{(N-1)\times(N-1)}$. *Die Matrix* A_h *hat dabei die Eigenschaft, dass sie strikt diagonaldominant ist, also die Diagonalelemente im Betrag jeweils größer als die Summe der Beträge aller anderen Elemente in der Zeile sind. Für solche Matrizen kann man relativ einfach zeigen, dass sie regulär sind. Folglich ist das lineare Gleichungssystem zu* (8.3) *eindeutig lösbar (und dies sogar effizient, da es nur wenige nicht-verschwindende Einträge gibt).*

Finite-Differenzen-Methode für die Wärmeleitungsgleichung. Möchte man über eine finite Differenzen-Methode eine zeitabhängige partielle Differentialgleichungen numerisch lösen, so muss man auch die Zeitableitung durch einen Differenzenquotienten approximieren, also im einfachsten Fall das explizite (oder implizite) Euler-Verfahren anwenden. Wir betrachten nun die Wärmeleitungsgleichung in

der oben bereits für die Poissongleichung behandelten Modellsituation $\Omega = (0,1)^2$ auf einem Zeitintervall $[0,T]$, also

$$\partial_t u - \Delta u = f \quad \text{auf } \Omega \times [0,T]$$

für eine Rand- und Anfangsbedingung

$$\begin{aligned} u &= g \quad \text{auf } \Gamma \times [0,T], \\ u(\cdot,0) &= u_0 \quad \text{auf } \Omega, \end{aligned}$$

für gegebene Funktionen $f\colon \Omega \times [0,T] \to \mathbb{R}$, $g\colon \Gamma \times [0,T] \to \mathbb{R}$ und Anfangswerte $u_0\colon \Omega \to \mathbb{R}$. Mit der oben beschriebenen äquidistanten Diskretisierung des (räumlichen) Gebietes Ω und seines Randes Γ mit Schrittweite $h = \frac{1}{N}$ erhalten wir aus der gegebenen partiellen Differentialgleichung zunächst das System gewöhnlicher Differentialgleichungen

$$\partial_t u_h - \Delta_h u_h = f_h \quad \text{auf } \Omega_h \times [0,T]$$

(wobei f zu jedem Zeitpunkt $t \in [0,T]$ analog zu oben diskretisiert wird). Diskretisieren wir nun zusätzlich das Zeitintervall $[0,T]$ mit Zeitschrittweite $\tau \coloneqq \frac{T}{n}$ für ein $n \in \mathbb{N}$, über

$$[0,T]_\tau \coloneqq \big\{ t_k \coloneqq k\tau \colon k \in \{0,1,\dots,n\} \big\},$$

so können wir eines der numerischen Verfahren zur Lösung des Systems zu (im Ort) diskretisierten Anfangswerte u_0 aus Kapitel 7 anwenden. Mit der Notation $u_h^k \coloneqq u_h(t_k)$ für die diskrete Lösung und $f_h^k \coloneqq f_h(t_k)$ für die rechte Seite führt etwa das explizite Euler-Verfahren auf die Berechnungsvorschrift

$$u_h^{k+1} = u_h^k + \tau\big(\Delta_h u_h^k + f_h^k\big)$$

und das implizite Euler-Verfahren auf

$$u_h^{k+1} = u_h^k + \tau\big(\Delta_h u_h^{k+1} + f_h^{k+1}\big).$$

Beim expliziten Euler-Verfahren kann folglich die Lösung zum nächsten Zeitschritt wieder direkt aus der Lösung zum vorherigen Zeitschritt bestimmt werden, wohingegen beim impliziten Euler-Verfahren ein lineares Gleichungssystem gelöst werden muss. Grundsätzlich können (und werden in der Praxis) für die Zeit-Diskretisierung auch andere explizite und implizite Verfahren verwendet, um bessere Stabilitäts-und Konvergenzeigenschaften zu gewährleisten. Analog zur Diskussion in Kapitel 7 ist das allgemeine Problem bei expliziten Verfahren aber, dass die Zeitschrittweite in der Regel sehr klein (auch in Abhängigkeit von der Gitterweite h im Ort) gewählt werden muss, damit es nicht zu unerwünschten Oszillationen oder Explosionen in der numerischen Lösung kommt.

8.2 Finite-Elemente-Methode

Bei der Finite-Elemente-Methode wird nicht die Ableitung durch Differenzenquotienten, sondern die Lösung durch einfachere Funktionen approximiert, nämlich als Linearkombination von endlich vielen linear unabhängigen (mindestens stetig differenzierbaren) Basisfunktionen $\varphi_1, \dots, \varphi_N$

$$u_h \coloneqq \sum_{j=1}^{N} \lambda_j \varphi_j \tag{8.5}$$

mit Koeffizienten $\lambda_1, \dots, \lambda_N \in \mathbb{R}$. Diese Koeffizienten werden durch den Übergang von der partiellen Differentialgleichung auf eine schwache (Integral-)Formulierung bestimmt. Betrachten wir wieder die Poissongleichung $-\Delta u = f$ aus (8.1) mit Nullrandwerten, so multiplizieren wir für deren Herleitung die Gleichung für die exakte Lösung mit einer glatten Testfunktion $\varphi \in C_0^1(\Omega)$ mit Nullrandwerten und integieren diese dann über Ω:

$$-\int_\Omega \Delta u \varphi \, dx = \int_\Omega f \varphi \, dx.$$

Mit partieller Integration folgt also

$$\int_\Omega \nabla u \cdot \nabla \varphi \, dx = \int_\Omega f \varphi \, dx \quad \text{für alle } \varphi \in C_0^1(\Omega).$$

Für die numerische Lösung u_h wird die Gültigkeit dieser Identität nur auf dem durch die Basisfunktionen $\varphi_1, \ldots, \varphi_N$ aufgespannten endlich-dimensionalen Unterraum verlangt, was wegen der Linearität in der Testfunktion gerade

$$\int_\Omega \nabla u_h \cdot \nabla \varphi_k \, dx = \int_\Omega f \varphi_k \, dx \quad \text{für } k \in \{1, \ldots, N\}$$

bedeutet. Über den Ansatz (8.5) der approximativen Lösung als Linearkombination aus Basisfunktionen wird also

$$\sum_{j=1}^{N} \lambda_j \int_\Omega \nabla \varphi_j \cdot \nabla \varphi_k \, dx = \int_\Omega f \varphi_k \, dx \quad \text{für } k \in \{1, \ldots, N\}$$

gefordert. Dies entspricht gerade dem linearen Gleichungssystem $A\lambda = b$, wobei die Matrix $A \in \mathbb{R}^{N \times N}$ und der Vektor $b \in \mathbb{R}^N$ über die Einträge

$$a_{k,j} \coloneqq \int_\Omega \nabla \varphi_j \cdot \nabla \varphi_k \, dx \quad \text{und} \quad b_k \coloneqq \int_\Omega f \varphi_k \, dx \quad \text{für } k, j \in \{1, \ldots, N\}$$

definiert sind. Die Matrix A ist dabei nach Definition symmetrisch und wegen der linearen Unabhängigkeit der Funktionen $\varphi_1, \ldots, \varphi_N \in C_0^1(\Omega)$ auch positiv definit. Damit besitzt das Gleichungssysrem $A\lambda = b$ eine eindeutige Lösung, der Komponenten gerade die Koeffizienten der approximativen Lösung in der Darstellung (8.5) bilden. Ein wesentlicher Einfluss auf die Approximationsgüte der numerischen Lösung sowie die Effizienz ihrer Berechnung liegt bei der Finite-Elemente-Methode in der Wahl geeigneter Basisfunktionen. Um die Einträge der Matrix A, die als Integrale von Produkten der Ableitungen der Basisfunktionen auftreten, effizient berechnen zu können, werden häufig polynomiale oder stückweise polynomiale Funktionen verwendet. Zudem werden Basisfunktionen mit lokalem Träger gewählt, deren Träger also nur auf wenige "Elemente" der Zerlegung des Gebietes Ω beschränkt ist. Dadurch überlappen jeweils nur wenige Basisfunktionen, so dass die Matrix A dünn besetzt und das resultierende Gleichungssystem dann mit vergleichsweise geringem Aufwand lösbar ist.

Bemerkung 8.4. *Ein weiteres Verfahren für die numerische Lösung spezieller partieller Differentialgleichungen, die auf Erhaltungsgesetzen beruhen, ist die* Finite-Volumen-Methode*. Bei diesem Verfahren wird das Gebiet in kleine Kontrollvolumina (typischerweise von polygonialer oder polyedrischer Gestalt) zerlegt, auf denen die zugrundeliegenden Erhaltungsgrößen wie etwa Masse oder Energie (in integraler Form) auch in der Diskretisierung erhalten bleiben.*

Literaturverzeichnis

[1] P. Deuflhard, A. Hohmann, *Numerische Mathematik 1 – Eine algorithmisch orientierte Einführung*, 5. Auflage, de Gruyter Lehrbuch, Walter de Gruyter & Co., Berlin, 2008.

[2] R.W. Freund, R.H.W. Hoppe, *Stoer/Bulirsch: Numerische Mathematik 1*, 10. Auflage, Springer-Lehrbuch, Springer-Verlag, Berlin Heidelberg, 2007.

[3] C.F. Higham, D.J. Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Rev. 61 (4), 860–891, 2019.

[4] D. Peterseim, *Numerische Verfahren*, Vorlesungsskript Sommersemester 2024, Universität Augsburg, 2024.

[5] J. Stoer, R. Bulirsch, *Numerische Mathematik 2*, 5. Auflage, Springer-Lehrbuch, Springer-Verlag, Berlin Heidelberg, 2005.