Algorithmische Mathematik und Programmieren / Numerik I

Skript nach Vorlesungen von

Prof. Dr. Angela Kunoth

im Wintersemester 2015/16 und Sommersemester 2016

an der Universität zu Köln

Überarbeitet von Laslo Hunhold

Version vom 21. Oktober 2015

Die erste Fassung dieses Skripts wurde von Sandra Görke und Simon Jörres an der Universität Bonn ausgearbeitet, mit Abschnitten aus dem Skript zur Finanznumerik I erarbeitet von Kristina Rupp und Benjamin Letschert an der Universität Paderborn.

Prof. Dr. Angela Kunoth
Mathematisches Institut
Universität zu Köln
Weyertal 86-90
50931 Köln

kunoth@math.uni-koeln.de
http://www.mi.uni-koeln.de/AG-Kunoth/de/left/lehre/ss15/num1/

Inhaltsverzeichnis

Inhalte der Vorlesung

Die Ziele der *Numerischen Mathematik* oder *Numerik* sind die Konstruktion und das mathematische Verständnis von Algorithmen zur konkreten Auswertung mathematischer Formeln auf Computern. Wesentliche *Nebenbedingungen* hierbei sind, daß

- Rechenmaschinen nur *endlich viele Zahlen* zur Verfügung haben und folglich Rechenoperationen nur im Rahmen der *Maschinengenauigkeit* ausgeführt werden können. Dies führt zu sogenannten *Rundungsfehlern*;
- man nur *endlichen Speichervorrat* hat. Viele Funktionen können daher im Rechner nur *approximativ*, d.h. angenähert, dargestellt werden. Hierbei entstehen *Diskretisierungsfehler*;
- man mit *beschränkter Rechenzeit* auskommen muß. Dies kann in vielen Problemen dazu führen, daß nur *näherungsweise Lösungen* erreichbar sind. Die hierbei entstehenden Fehler werden unter dem Schlagwort *Verfahrensfehler* subsumiert.

In der Praxis errechnete Resultate können also mit einer Vielzahl verschiedener *Fehler* und *Fehlerquellen* behaftet sein. Daher ist es Aufgabe der modernen Numerik, für konkrete Problemstellungen zu entscheiden, welcher Algorithmus mit den zur Verfügung stehenden Ressourcen eine vorgegebene Genauigkeit mit dem geringsten Aufwand erreicht.

Typische *Anwendungsbeispiele*, in denen die Numerik oder numerische Methoden zum Zuge kommen, sind *numerische Simulationen* zur Berechnung von Dynamik. So ist es etwa in der *Automobilindustrie* heutzutage üblich, lange bevor der erste Prototyp eines neuen Fahrzeugs gebaut wird, *Crashtests* oder *Bremsmanöver* im Computer zu simulieren. Hierzu werden numerische Lösungen von *gewöhnlichen* oder *partiellen Differentialgleichungen* gebildet. Weiter ist es bei *Wettervorhersagen* nötig, enorm große *Datenmengen* und *komplizierte Strömungsvorgänge* auf sehr unterschiedlichen Längenskalen zu verarbeiten bzw. zu simulieren. Man sucht approximative Lösungen von gewöhnlichen oder partiellen Differentialgleichungen, was letztendlich auf die Lösung von (linearen oder nichtlinearen), in der Regel sehr großen Gleichungssystemen (mit Millionen oder Milliarden von Unbekannten) führt. Der in den vergangenen Jahren sehr populär gewordene Bereich der *Finanzmathematik* erfordert zur Bewertung von Optionen ebenfalls die Berechnung der Lösung partieller Differentialgleichungen (*verallgemeinerte Black-Scholes-Gleichungen*) oder die Berechnung sehr hochdimensionaler Integrale.

Ein ganz anderer Bereich, in dem die Numerik zum Tragen kommt, sind bildgebende Verfahren. Hier ist die Verarbeitung und Analyse großer Datenmengen erforderlich. In der *medizinischen Diagnostik*, vor allem in der *Computertomographie*, sind neben der Lösung von Integralgleichungen Signal– und Bildanalysen sowie Bildkompressionen von großer Wichtigkeit.

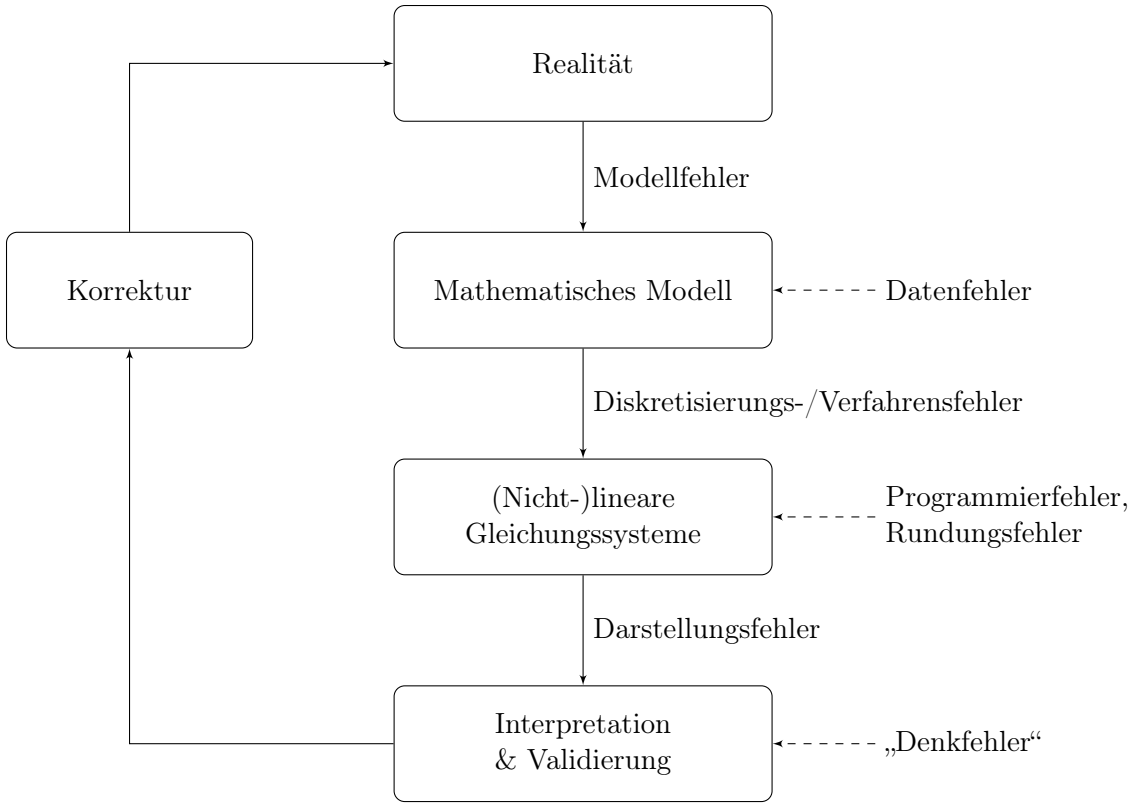
Die Numerische Mathematik versteht sich als ein Teil des *Wissenschaftlichen Rechnens (Scientific Computing)*, zu dem außerdem Disziplinen wie Computeralgebra, Computational Number Theory, Computational Chemistry o.ä. gehören können. Dieser interdisziplinäre Wissenschaftszweig hat in den letzten Jahren erheblich an Bedeutung gewonnen. Daher sollen die einführenden Beispiele von oben als ein erster Vorgeschmack für den stark anwendungsorientierten Charakter der Numerik dienen. Im Laufe der Vorlesung werden wir noch weitere Beispiele hierzu kennenlernen, die allesamt die immer größere Wichtigkeit des interdisziplinären Forschens verdeutlichen sollen.

Die Numerik umfaßt zur Hauptsache die Gebiete *Numerische Analysis*, bei der man etwa die Approximation von Funktionen und Integralen sowie die approximative Lösung von Differentialgleichungen diskutiert, und die *Numerische Lineare Algebra*, die vor allem die Lösung von linearen Gleichungssystemen oder die Berechnung von Eigenwerten behandelt.

Die Durchführung von numerischen Simulationen im Wissenschaftlichen Rechnen lässt sich in vier Schritte unterteilen:

1. MODELLIERUNG: Man benötigt eine Abbildung, die die Realität in einem Modell darstellt, sodaß man das Problem auf die wesentlichen Daten reduzieren und mit ihnen arbeiten kann. Aus geeigneten (Mess-)Daten erhält man dann z.B. ein mathematisch-physikalisches Modell für die "Dynamik" des Prozesses und muss nun Gleichungen lösen wie z.B. $F(y) = f$ mit gegebenem f und gesuchtem y. Oft führt dies auf gewöhnliche oder partielle Differentialgleichungen.
2. THEORIE: Um ein Problem erfolgreich lösen zu können, benötigt man theoretisch die Existenz und Eindeutigkeit einer Lösung, da dies in der Realität gegeben ist. Idealerweise ist das Problem wohlgestellt, und es liegt eine stetige Abhängigkeit der Lösung von den Daten vor.
3. NUMERIK: Nun erfolgt die Diskretisierung und numerische Lösung von $F(y) = f$:
   Man muss sich zunächst Gedanken über die sogenannte Konsistenz, die Stabilität und die Konvergenz von numerischen Verfahren machen, bevor man zur Implementierung des Algorithmus' kommen kann. Aufgrund der hohen Komplexität vieler Probleme benötigt man effiziente Lösungsverfahren, wie z.B. Multiskalenmethoden, parallele oder adaptive Algorithmen. Schließlich ist eine Visualisierung der Lösung nötig.
4. VALIDIERUNG: In diesem letzten Schritt erfolgt dann der Abgleich der approximativen Lösung mit den aus 1. gegebenen Daten und gegebenenfalls eine Korrektur des Modells, wobei dann wieder bei Punkt 1. begonnen werden muss.

Das Vorgehen soll im folgenden Diagramm skizziert werden, in dem in jedem Schritt zusätzlich die möglichen Fehlerquellen gekennzeichnet sind:



Im Rahmen der Algorithmischen Mathematik und der Numerik I/II werden folgende Inhalte behandelt:

- **Fehleranalyse:** Zu Anfang der Vorlesung ist es nötig, über mögliche Fehlerquellen in Algorithmen nachzudenken. Wir werden Verfahren kennenlernen, um die Fehlerfortpflanzung in einer Rechnung abzuschätzen. Dies wird unter anderem auf den Begriff der Kondition eines Problems führen. Weiter werden wir die Darstellung von Zahlen in Rechnern und die damit verbundenen Rundungsfehler besprechen. Ein weiteres Merkmal eines Algorithmus' ist seine Zuverlässigkeit. Dies wird in einer das Kapitel abschließenden Stabilitätsanalyse dargelegt werden.

- **Verfahren zur Lösung linearer Gleichungssysteme:** Hier behandeln wir zunächst die im Rahmen der Gauss–Elimination verwendete LR–Zerlegung von Matrizen. Weiter werden numerisch geeignetere Verfahren wie die QR–Zerlegung diskutiert. Darüber hinaus werden wir uns überlegen, wie man gewisse Strukturen von Matrizen ausnutzen kann, um diese Verfahren effizienter zu gestalten. Aus diesem Blickwinkel werden wir das Cholesky–Verfahren für symmetrisch positiv definite Matrizen sowie das Ausnutzen von Bandstrukturen von Matrizen diskutieren.

- **Ausgleichsrechnung (Least–Squares–Approximation):** In diesem Kapitel werden wir die Verfahren aus dem vorherigen Abschnitt der Vorlesung anwenden, um die in vielen Anwendungen wichtigen Ausgleichsprobleme über Normalengleichungen und direkt mit QR–Zerlegung zu berechnen. Bei dieser Art quadratischer Optimierung weiter ins Spiel kommende Techniken sind die Singulärwertzerlegung und die Pseudoinverse einer Matrix.

- **Berechnung von Eigenwerten und Eigenvektoren:** In der Linearen Algebra haben wir die Eigenwerte einer Matrix immer aus dem charakteristischen Polynom berechnet, was sich aus numerischer Sicht als ein schlecht konditioniertes Problem erweist. Nach der Diskussion theoretischer Grundlagen und Methoden zu Eigenwertabschätzungen lernen wir zwei Iterations– (Power–Iteration und Inverse Power–Iteration) sowie ein auf der QR–Zerlegung basierendes Verfahren kennen.

- **Iterative Verfahren zur Lösung nichtlinearer Gleichungen:** Dieser Abschnitt behandelt die Bestimmung von Nullstellen von nichtlinearen Gleichungssystemen. Dies führen wir zunächst für den skalaren und später für den vektorwertigen Fall durch. Die theoretische Grundlage für die Existenz und Eindeutigkeit von Lösungen von Iterationsverfahren, die meist aus Fixpunktgleichungen gewonnen werden, ist der Banach'sche Fixpunktsatz. Das wichtigste Verfahren zur Lösung dieser Problematik ist das Newton–Verfahren, welches wir in verschiedenen Varianten (Gauss–Newton–Verfahren, gedämpftes Newton–Verfahren) diskutieren werden.

- **Nichtlineare Ausgleichsprobleme:** Die für lineare Ausgleichsprobleme gewonnenen Erkenntnisse werden derart erweitert, daß wir sie auf nichtlineare Gleichungssysteme anwenden können.

- **Interpolation mit Polynomen:** Bei der Interpolation will man im Gegensatz zur Approximation wie bei den Ausgleichsproblemen gegebene (Mess-)Werte mit Funktionen nicht nur angenähert, sondern exakt darstellen. Die für uns wichtigsten Verfahren sind die Hermite–und die Lagrange–Interpolation mit Polynomen. Neben Existenz und Eindeutigkeit werden Darstellungsformen (bezüglich der Monom–, Lagrange– und Newton–Basis unter Verwendung dividierter Differenzen) sowie deren schnelle Auswertung (Algorithmus von Neville–Aitken) diskutiert.

- **Splinefunktionen:** Die Interpolation mit Polynomen gerät aus numerischer Sicht schnell an ihre Grenzen, sobald man viele Daten zu interpolieren hat. Ein grosser Bereich der Numerik und deren Anwendungen widmet sich daher zunehmend der Interpolation auf der Basis von Splines (stückweisen Polynomen). Zentral wird die Darstellung und hocheffiziente Auswertung von Splines über B–Spline–Basen sein.
- **Numerische Integration (Quadratur):** Selbst bei gegebenem Integranden kann die Integration einer Funktion theoretisch häufig nicht durchgeführt werden. Zur numerischen Berechnung greift man daher entweder auf Punktauswertungen des Integranden zu oder approximiert den Integranden durch einfacher zu integrierende Funktionen wie Splines (Newton-Cotes-Formeln). Eine weitere grosse Klasse von Quadraturverfahren liefern die auf Orthogonalpolynomen basierenden Gauss-Formeln, die sehr viel höhere Genauigkeit bei glatten Integranden liefern.

Die weiterführende Vorlesung Numerik II wird (neben den obigen, in der Numerik I nicht behandelten Themen) schwerpunktmäßig die Numerik gewöhnlicher Differentialgleichungen zum Inhalt haben. Diese Vorlesungsreihe wird anschliessend durch Themen der Numerik partieller Differentialgleichungen und des Wissenschaftlichen Rechnens fortgesetzt.

Da sich viele konstruktive Verfahren aus der Theorie nicht zur praktischen Durchführung auf Computern eignen, erfordert für schwierige Probleme die Entwicklung guter numerischer Verfahren umfangreiche Kenntnisse und große Erfahrung. Aus diesem Grunde scheint die These von Hardy,

> that Mathematicians make their best contributions, on the average, at age 38.8,

für Numeriker nicht unbedingt zuzutreffen. Vielmehr gilt die Faustregel aus [T], Seite 1093:

> Zu jedem noch so eleganten numerischen Verfahren gibt es Gegenbeispiele, bei denen das Verfahren völlig versagt.

Abschließend seien noch einige Bemerkungen zu vorlesungsbegleitender Literatur gegeben. Moderne Lehrbücher der Numerik sind z.B. [DH] und [H]. Besonders empfehlenswert aufgrund einer sehr präzisen mathematischen Beschreibung und wegen vielfältiger Beispiele aus den Ingenieurwissenschaften ist [DR]. Die Lehrbücher [S] und [HH] haben den Status von Klassikern der Numerischen Mathematik, gelten jedoch aufgrund der heutigen erheblich höheren Rechnerleistungen mittlerweile in einigen Belangen und in der Schwerpunktbildung als etwas überholt, aber dennoch lesenswert. Ein sehr umfassendes Buch zur Numerischen Linearen Algebra ist [GL]. Hierin befinden sich viele weitere Literaturhinweise und eine große Zahl von praktisch durchgerechneten Beispielen. Als Standardreferenz für aktive Wissenschaftler aller Disziplinen, die numerische Methoden verwenden, gilt [PTVF]. Ein Buch, welches hervorragend und dennoch auf knappem Raum die C–Programmiersprache behandelt, ist [KR]. Weitere, auf spezielle Themen bezogene Literaturhinweise, werden in den einzelnen Kapiteln erfolgen.

1 Einführung

Das Anforderungsprofil eines Mathematikers in der Numerik bildet heute die Schnittstelle zwischen der Mathematik und einigen anderen angewandten Wissenschaften.

Es ist keine Seltenheit, daß viele Anwendungen in Natur-, Ingenieurs-, Wirtschafts- und anderen Wissenschaften eine zunehmende „Mathematisierung“ verzeichnen. Beispiele hierfür sind die Berechnung von Gas- und Flüssigkeitsströmungen, die etwa im *Flugzeugbau* bei der Entwicklung neuer Modelltypen eine große Rolle spielen. Auch bei *Wettervorhersagen* sind solche Berechnungen wichtig. *Flugbahnberechnungen in der Raumfahrt* wären ohne die moderne Mathematik nicht denkbar.

In der zunehmenden Automatisierung vieler Lebensbereiche ist die *Robotersteuerung* nur ein Bereich. *Netzwerkberechnungen* und *Halbleiterdesign* sind in den *Computerwissenschaften* von Bedeutung. In der *Architektur* und *Baustatik* spielen die *Berechnung und Modellierung von Schwingungsvorgängen und Resonanz* eine ernstzunehmende Rolle.

Ein interessanter Fall für das Vorliegen eines derartigen Modellierungsfehlers ist die *Londoner Millennium-Bridge*, die kurz nach ihrer Eröffnung aufgrund von übermäßiger Resonanzwirkung vorübergehend wieder geschlossen werden mußte. In den *Werkstoffwissenschaften* sorgen *Verformungs- und Elastizitätsprobleme* für den Einsatz der Numerik. Auch die *Automobilindustrie* verfertigt *Karosserieentwürfe* ausschließlich mit Hilfe von numerischen Methoden. *Bildgebende Verfahren* spielen zum Beispiel in der *Medizin* im Bereich der *Computertomographie* eine wichtige Rolle. Auch in den klassischen Naturwissenschaften hat die Numerik als *Computational Biology* oder *Computational Chemistry* Einzug gehalten. Ein immer wichtigeres Feld in den *Wirtschaftswissenschaften* ist der Bereich *Computational Economics*. Vor allem in der *Finanzmathematik* sind numerische Methoden unter dem Schlagwort *Computational Finance* etwa bei der Diskretisierung von Differentialgleichungen zur Preisberechnung von *Optionsscheinen* oder anderen *Finanzderivaten* wichtig [GJ][Sey].

Dazu betrachten wir ein Beispiel, in dem die Schnittpunkte der Mathematik mit einer angewandten Wissenschaft dargestellt werden.

Beispiel 1.1 (Mortgage-Backed Securities (MBS))**.** *Bei Mortgage-Backed Securities (durch Hypotheken gesicherte Wertpapiere, häufig Collaterized Mortgage Obligations (CMOs, nach den Autoren von [CMO] genannt) handelt es sich um ein sogenanntes abgeleitetes Finanzprodukt, das seit einigen Jahren für den Finanzmarkt, vor allem in den USA, von großer Bedeutung ist.*

Um numerische Methoden zur Bewertung des Papiers durchführen zu können, benötigt man ein mathematisches Modell *der MBS; dies soll später vorgestellt werden.*

Letzlich führt dies auf die Aufgabe, ein hochdimensionales Integral

$$\int_{[0,1]^d} f(x)\,\mathrm{d}x, \quad f : [0,1]^d \to \mathbb{R} \text{ gegeben}, \tag{1.2}$$

mit einem sehr großen d zu berechnen. Typischerweise ist $d = 256$, da es 256 Handelstage im Jahr gibt, oder $d = 360$, da solche Verträge über 30 Jahre mit monatlichen Wertberechnungen laufen.

Man könnte versuchen, dies auf die Approximation eindimensionaler Integrale durch Punktauswertungen

$$\int_{[0,1]} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{N} a_i f(x_i) \tag{1.3}$$

an vorgegebenen Stellen x_i zurückzuführen. Konkret benötigt man Quadraturformeln zur approximativen Berechnung des Integrals.

Der nächste Schritt ist die Realisierung des Algorithmus *auf dem Rechner, verbunden mit dessen Programmierung. Das anschließende* Postprocessing *sollte der Visualisierung und der Validierung der Ergebnisse dienen.*

Dabei könnte es sich herausstellen, daß die Ergebnisse mit den Beobachtungen aus der Praxis nicht übereinstimmen. Im obigem Fall der Berechnung dieses hochdimensionalen Integrals stellt man im Übrigen sehr schnell fest, daß sich bekannte Quadraturformeln wie bei der eindimensionalen Quadratur aus Speicherplatzgründen nicht anwenden lassen (siehe Kapitel 8.8). Einen Ausweg bieten hier die Monte-Carlo-Methoden *und die* Quasi-Monte-Carlo-Methoden*, die zufällig oder auf eine raffinierte deterministische Art die Quadraturpunkte x_i in der hochdimensionalen Version von* (1.3) *auswählen.*

Bei diesem Vorgehen müssen verschiedene mögliche Fehlerquellen berücksichtigt werden. Eine sinnvolle Auswertung der Ergebnisse setzt ein Verständnis dieser voraus. Im ersten Schritt können *Daten–*, *Eingangs–* oder *Meßfehler* vorliegen. *Modellfehler* durch idealisierte Annahmen, z.B. um die Existenz und Eindeutigkeit der Lösung sicherzustellen, können im zweiten Schritt entstanden sein. Es kann im dritten Schritt zu *Abbruch–*, *Diskretisierungs–* und *Verfahrensfehlern* aufgrund der nur näherungsweise berechneten Lösung mittels numerischer Verfahren gekommen sein. Dies kann selbst bei exakter Lösung der Fall sein, denn das Ergebnis hängt z.B. vom Typ des Integranden und der Quadraturformel ab. *Rundungsfehler* können bei der Durchführung des Algorithmus im vierten Schritt hervorgerufen worden sein. Zusätzlich können natürlich auch nicht zu vernachlässigende *Denk–* und *Programmierfehler* vorliegen.

Daraus formulieren wir eine erste (offensichtliche) Forderung.

> Man sollte numerische Verfahren stets so konstruieren, daß Abbruch–, Diskretisierungs–, Modell–, Rundungs– und Verfahrensfehler möglichst minimiert werden.

Das Beispiel motivierte zwei wesentliche Kriterien für die Güte eines Verfahrens:

- *Genauigkeit*: Das Ergebnis soll innerhalb von Toleranzen *verlässlich* sein.
- *Effizienz*: Bei der Rechnung sollte eine möglichst geringe Anzahl von Rechenoperationen und möglichst wenig Speicherplatz benötigt werden.

Für deren Anwendung gilt aber leider auch die folgende Faustregel.

> Bessere Ergebnisse lassen sich in der Regel nur mit höherem Aufwand erzielen. Umgekehrt liefert höherer Aufwand aber nicht unbedingt auch bessere Ergebnisse.

Im Laufe der Vorlesung wird oft die Größenordnung eines Fehlers oder der Aufwand von Rechnungen abzuschätzen sein. Um hierbei die Notation zu vereinfachen, führen wir die Landau–Symbole ein:

Definition 1.4 (Asymptotische obere Schranke)**.** *Seien f und g zwei Funktionen. Dann heißt*

$$f(x) = \mathcal{O}(g(x)) \quad \text{für} \quad x \to x_0 \in \mathbb{R} \cup \{-\infty, +\infty\},$$

falls es eine Konstante $c \in \mathbb{R}$ gibt, sodaß

$$\limsup_{x \to x_0} \left| \frac{f(x)}{g(x)} \right| \leq c < \infty.$$

Also ist f nach oben durch g asymptotisch beschränkt. Das heißt, f wächst unwesentlich schneller als g.

Definition 1.5 (Asymptotisch gegenüber g vernachlässigbar)**.** *Seien f und g zwei Funktionen. Dann heißt*

$$f(x) = o(g(x)) \quad \text{für} \quad x \to x_0 \in \mathbb{R} \cup \{-\infty, +\infty\},$$

falls

$$\lim_{x \to x_0} \left| \frac{f(x)}{g(x)} \right| = 0.$$

Also ist f gegenüber g asymptotisch vernachlässigbar. Das heißt, f wächst wesentlich langsamer als g.

Beispiel 1.6 (Verwendung von „groß $\mathcal{O}$“ und „klein o“)**.** *Es gilt*

$$\sin(x) = \mathcal{O}(x) \quad \text{für} \quad x \to 0, \quad \text{da} \quad \lim_{x \to 0} \frac{\sin(x)}{x} = 1$$

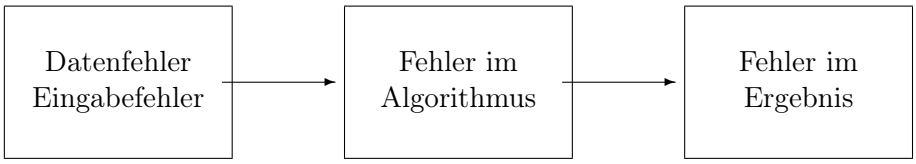
$$\sin(x) = o(\sqrt{x}) \quad \text{für} \quad x \to 0, \quad \text{da} \quad \lim_{x \to 0} \frac{\sin(x)}{\sqrt{x}} = 0$$

So lässt sich etwa das Restglied der Taylor–Entwicklung durch „groß $\mathcal{O}$“ ausdrücken:
Für die Taylor–Entwicklung an der Stelle $x + h$ einer Funktion $f \in \mathcal{C}^k(\mathbb{R})$ mit $h \in (0, 1)$ gilt:

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \mathcal{O}(h^3) \qquad \text{für} \quad h \to 0$$

2 Fehleranalyse: Kondition, Rundungsfehler, Stabilität

Beim Ausführen von Rechnungen gibt es verschiedene Fehlerquellen:



Wir gehen der Frage nach, wie man diese vermeiden oder minimieren kann, bzw. wie sich Fehlerverstärkungen nach oben abschätzen lassen. Die Vermeidung von Daten- und Eingabefehlern ist oft nicht möglich, sie gehören zum Problem. Dies wird uns auf den Begriff der *Kondition eines Problems* führen. Durch Algorithmen verursachte Fehler lassen sich bisweilen durch deren Modifikation beheben. Man untersucht dabei die *Stabilität eines Algorithmus*.

2.1 Kondition eines Problems

Die *Kondition* eines Problems gibt an, welche Genauigkeit man bei exakter Lösung bestenfalls erreichen kann:

Ein Problem heißt *gut konditioniert*, wenn kleine Störungen der Eingangsdaten kleine Änderungen im Ergebnis bewirken, sonst *schlecht konditioniert*.

Mathematisch beschreibt man dies folgendermaßen:

Man fasst das Problem als Funktionsauswertung einer Funktion $f : U \to \mathbb{R}$ auf einer offenen Menge $U \subset \mathbb{R}^n$ an einer Stelle $x \in U$ auf.

Beispiel 2.1.1. Multiplikation zweier reeller Zahlen: Werte $f(x_1, x_2) := x_1 \cdot x_2$ aus.

Beispiel 2.1.2. Addition zweier reeller Zahlen: Werte $f(x_1, x_2) := x_1 + x_2$ aus.

Beispiel 2.1.3. Bestimmung der kleineren Nullstelle von $u^2 - 2a_1u + a_2 = 0$ mit $a_1, a_2 \in \mathbb{R}_{>0}$ und $a_1^2 > a_2$. Die Lösung hiervon ist $u^* = f(a_1, a_2) := a_1 - \sqrt{a_1^2 - a_2}$.

Falls die Funktion f explizit bekannt ist, kann die *quantitative Beschreibung der Kondition* über die Taylor–Entwicklung erfolgen. Sei dazu $f : U \subset \mathbb{R}^n \to \mathbb{R}$ mit $f \in \mathcal{C}^2(U)$. Dann gilt

$$f(\tilde{x}) = f(x) + (\tilde{x} - x)^T \nabla f(x) + \mathcal{O}(\|\tilde{x} - x\|^2).$$

Wir nehmen im Folgenden an, daß $\|\tilde{x} - x\|$ „klein“ ist, sodaß wir die Terme 2. Ordnung vernachlässigen können. Für diese lineare Approximation schreiben wir

$$f(\tilde{x}) \doteq f(x) + (\tilde{x} - x)^T \nabla f(x). \tag{2.1.4}$$

Das Zeichen $\doteq$ steht also für „in 1. Näherung“. Dies zieht

$$f(\tilde{x}) - f(x) \doteq \sum_{i=1}^{n} (\tilde{x}_i - x_i) \frac{\partial f(x)}{\partial x_i}$$

nach sich, was unter der Annahme $f(x) \neq 0$ äquivalent ist zu

$$\frac{f(\tilde{x}) - f(x)}{f(x)} \doteq \sum_{i=1}^{n} \frac{\partial f(x)}{\partial x_i} \frac{x_i}{f(x)} \frac{\tilde{x}_i - x_i}{x_i}.$$

Nun folgt mit der Dreiecksungleichung

$$\underbrace{\left|\frac{f(\tilde{x}) - f(x)}{f(x)}\right|}_{\substack{\text{Relativer Fehler} \\ \text{in der Ausgabe}}} \dot{\leq} \sum_{i=1}^{n} \quad \underbrace{\left|\frac{\partial f(x)}{\partial x_i} \frac{x_i}{f(x)}\right|}_{\substack{=: |\varphi_i(x)| \\ \textit{Verstärkungsfaktor}}} \quad \cdot \quad \underbrace{\left|\frac{\tilde{x}_i - x_i}{x_i}\right|}_{\substack{\text{Relativer Fehler} \\ \text{in der Eingabe}}} \quad . \tag{2.1.5}$$

Damit definieren wir

$$\kappa_{\text{rel}} := \|\varphi(x)\|_\infty := \max_{i=1,\ldots,n} |\varphi_i(x)| \tag{2.1.6}$$

als *relative Kondition* von f und

$$\kappa_{\text{abs}} := \|\nabla f(x)\|_\infty \tag{2.1.6a}$$

als *absolute Kondition* von f, und sagen, f sei *schlecht konditioniert*, wenn $\kappa_{\text{rel}} \gg 1$.

Beispiel 2.1.7. Wir betrachten die Multiplikation aus Beispiel 2.1.1: Seien

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad f(x) = x_1 \cdot x_2, \qquad \text{d.h.} \quad \nabla f(x) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{2.1.8}$$

$\nabla f(x)$ bezeichnet den Gradienten von f definiert als $\left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right)^T$ für $f : \mathbb{R}^n \to \mathbb{R}$ differenzierbar.

Hier gilt also

$$\varphi_1(x) = \frac{\partial f(x)}{\partial x_1} \cdot \frac{x_1}{f(x)} = x_2 \cdot \frac{x_1}{x_1 \cdot x_2} = 1 = \varphi_2(x).$$

Also ist $\kappa_{\text{rel}} = 1$ unabhängig von x. Die Multiplikation zweier reeller Zahlen ist demnach *gut konditioniert*. ∎

Beispiel 2.1.9. Wir betrachten die Addition aus Beispiel 2.1.2. Seien nun

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad f(x) = x_1 + x_2, \qquad \text{d.h.} \quad \nabla f(x) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{2.1.10}$$

Da

$$\varphi_1(x) = 1 \cdot \frac{x_1}{x_1 + x_2} = \frac{x_1}{x_1 + x_2}; \qquad \varphi_2(x) = \frac{x_2}{x_1 + x_2}$$

folgt

$$\kappa_{\text{rel}} = \frac{\max\limits_{i=1,2} |x_i|}{|x_1 + x_2|}.$$

Das bedeutet

$$\kappa_{\text{rel}} \ = \ \begin{cases} \leq 1 & \text{falls } \operatorname{sign}(x_1) = \operatorname{sign}(x_2) \\ \gg 1 & \text{falls } x_1 \approx -x_2 \end{cases}$$

Im Fall $x_1 \approx -x_2$ spricht man von *Auslöschung führender Ziffern.* Hieran sieht man, daß in Algorithmen die Subtraktion gleichgroßer Zahlen unbedingt vermieden werden sollte. ∎

Nun verallgemeinern wir unsere Untersuchungen auf Fälle, in denen wir die Funktion f nicht explizit kennen und demnach auch die TAYLOR–Entwicklung nicht anwendbar ist. Seien $U \subset \mathbb{R}^n$ offen und $f : U \to \mathbb{R}^m$. Desweiteren bezeichne $\|\cdot\|$ eine Norm auf $\mathbb{R}^n$ bzw. $\mathbb{R}^m$. Wir definieren die Menge der Eingabedaten in einer Umgebung von x mit $\delta > 0$ als

$$E_\delta(x) := \{\tilde{x} : \|x - \tilde{x}\| \leq \delta \|x\|\} \tag{2.1.10a}$$

und sagen

Definition 2.1.11. Das Problem $[f, x, \delta]$ ist gut gestellt (well–posed), falls es ein $L_{\text{rel}}(\delta) \in \mathbb{R}^+$ gibt, sodaß für alle $\tilde{x} \in E_\delta(x)$ mit $x \neq 0$ und $f(x) \neq 0$

$$\frac{\|f(x) - f(\tilde{x})\|}{\|f(x)\|} \ \leq \ L_{\text{rel}}(\delta) \cdot \frac{\|\tilde{x} - x\|}{\|x\|} \tag{2.1.12}$$

gilt. Ist $[f, x, \delta]$ gut gestellt, so bezeichnet $L_{\text{rel}}(\delta)$ die *minimale* Konstante, sodaß (2.1.12) gilt. Gibt es keine solche Konstante, so heißt $[f, x, \delta]$ schlecht gestellt (ill-posed).

Die relative Kondition des Problems $[f, x]$ definieren wir dann durch

$$\kappa_{\text{rel}} \ := \ \lim_{\delta \to 0} L_{\text{rel}}(\delta) \tag{2.1.13}$$

Falls κ_{rel} klein ist, so heißt $[f, x]$ gut konditioniert. ∎

Man beachte, daß sich (2.1.12) als eine lokale LIPSCHITZ–Bedingung interpretieren lässt.

Bemerkung 2.1.14. Ist f differenzierbar, folgt aufgrund des Mittelwertsatzes für die relative Kondition

$$\kappa_{\text{rel}} \ = \ \frac{\|x\|}{\|f(x)\|} \cdot \|Df(x)\| \tag{2.1.15}$$

wobei $\|Df(x)\|$ die Norm der JACOBI–Matrix $Df(x) \in \mathbb{R}^{m \times n}$ in der zur Vektornorm $\|\cdot\|$ gehörigen Matrixnorm

$$\|A\| \ := \ \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} \tag{2.1.15a}$$

für $A \in \mathbb{R}^{m \times n}$ bezeichnet. ∎

Definition 2.1.15b. *Weitere wichtige Matrixnormen sind*

$$\|A\|_\infty := \max_{i=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|, \quad \|A\|_1 := \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|, \quad \|A\|_2 := \sqrt{\lambda(A^T A)}$$

$\lambda(A^T A)$ *bezeichnet den betragsmäßig größten Eigenwert der* symmetrisch positiv definiten *Matrix* $A^T A$ *(siehe 3.4.1).*

Mit dieser neuen Definition der relativen Kondition wollen wir die Kondition der Lösung eines linearen Gleichungssystems als Beispiel betrachten.

Beispiel 2.1.16. Seien $A \in \mathbb{R}^{n\times n}$ nichtsingulär und $b \in \mathbb{R}^n$ gegeben. Gesucht ist die Lösung $y \in \mathbb{R}^n$ von $Ay = b$. Zur Berechnung der relativen Kondition von y betrachten wir das Problem $f(b) = A^{-1}b$. Diese Gleichung ist offenbar linear bezüglich b und es gilt daher für die Ableitung $Df(y) = A^{-1}$. Setzen wir dies in die obige Formel (2.1.15) zur Berechnung der relativen Kondition ein, erhalten wir

$$\kappa_{\text{rel}} = \frac{\|b\|}{\|A^{-1}b\|} \cdot \|A^{-1}\| = \frac{\|Ay\|}{\|y\|} \cdot \|A^{-1}\| \le \|A\| \cdot \|A^{-1}\|. \tag{2.1.17}$$

Man nennt $\kappa(A) := \|A\| \cdot \|A^{-1}\|$ die *Kondition* der Matrix A. Wenn $\|\cdot\| = \|\cdot\|_2$ (Euklidische Vektornorm), spricht man von der *spektralen Kondition* von A und schreibt $\kappa_2(A)$. Falls A fast singulär ist, ist $\kappa(A)$ groß. Dies bedeutet, daß kleine Störungen der Daten in b große Auswirkungen auf die Lösung x haben und das Problem in diesem Fall schlecht konditioniert ist.

2.2 Maschinenzahlen und Rundungsfehler

Bisher haben wir uns lediglich mit der Konditionierung von Problemen beschäftigt. Da diese letzendlich in Rechenmaschinen gelöst werden, ist es jedoch elementar wichtig, daß man ein gutes Verständnis für deren Beschränkungen hat und auftretende *Rundungsfehler* abschätzen kann. Die im Rechner darstellbaren Zahlen heißen *Maschinenzahlen* und lassen sich in zwei Klassen unterscheiden: Die Integerzahlen und Gleitkommazahlen.

2.2.1 Integerzahlen

Betrachten wir eine beliebige ganze Zahl $a \in \mathbb{Z}$ und die Basis $b \in \mathbb{N}, b \ge 2$. Dann hat a eine Darstellung der Form

$$a = (-1)^s \sum_{i=0}^{\infty} d_i b^i$$

Für spezielle Basen b gibt es eigene Bezeichnungen, z.B. Binär oder Dual für $b = 2$, Oktal für $b = 8$, Dezimal für $b = 10$ und Hexadezimal für $b = 16$.
Ist die Basis b fest gewählt, bleiben lediglich die Freiheitsgrade $d_i \in \{0, \dots, b-1\}$ und $s \in \{0, 1\}$ übrig.

Rechner arbeiten bekanntlich intern mit $b = 2$, da sie pro Speichereinheit lediglich die Zustände 1 und 0 abbilden können. Die Speichereinheit ist *Bit*. 8 Bits entsprechen 1 *Byte*.

Betrachten wir die Ganzzahldarstellung, so folgt mit $b = 2$, daß $d_i, s \in \{0, 1\}$ und damit ganze Zahlen im Computer darstellbar sind. Dazu wählen wir ein festes $m \in \mathbb{N}$ und betrachten unter der Annahme, daß $|a| < b^m$

$$a = (-1)^s \sum_{i=0}^{\infty} d_i b^i = (-1)^s \left(\sum_{i=0}^{m-1} d_i b^i + \underbrace{\sum_{i=m}^{\infty} d_i b^i}_{=0, \text{ da } |a|<b^m} \right) = (-1)^s \sum_{i=0}^{m-1} d_i b^i \tag{2.2.1}$$

Zur Speicherung benötigt man also $m + 1$ Bits, nämlich m Bits für die d_i und 1 Bit für das Vorzeichenbit s. Ob das d_0 im Speicherblock als erstes oder als letztes kommt, hängt von der *endianness* der Architektur ab. Auf einer *Big Endian* (BE) Architektur kommt das d_0 zuletzt, auf einer *Little Endian* (LE) Architektur kommt das d_0 zuerst. Das Vorzeichenbit orientiert sich entsprechend an der Lage des d_0.

Da nicht immer ein Vorzeichen benötigt wird, gibt es einerseits *unsigned* und andererseits *signed* Integer. Bei ersterem setzt man $s = 0$ fest und verringert damit den benötigten Speicherplatz um 1 Bit auf m Bits, beschränkt sich aber auf positive Zahlen.

m	Speicher	Zahlbereich	42 (Big Endian)	42 (Little Endian)
8	1 Byte	$0, \ldots, 2^8 - 1 (= 255)$	0010 1010	0101 0100
16	2 Bytes	$0, \ldots, 2^{16} - 1 (= 65535)$	0000 0000 0010 1010	0101 0100 0000 0000
32	4 Bytes	$0, \ldots, 2^{32} - 1 (= 4294967295)$	...	
64	8 Bytes	$0, \ldots, 2^{64} - 1 (\approx 1.84 \cdot 10^{19})$	...	

Lassen wir das Vorzeichenbit los, speichert das höchste Bit die Vorzeicheninformation.
„0“ entspricht „+“, „1“ entspricht „−“. Beachte hier, daß 1 Byte nun nur noch eine ganze Zahl mit $m = 7$ speichern kann.
Damit die Monotonie der Zahldarstellung erhalten bleibt, speichert man nach der „b-Komplement“-Methode:

$$a \in \{-2^{m-1}, \ldots, 2^{m-1} - 1\} \to \begin{cases} a & \text{falls } a \geq 0 \\ a + 2^{m-1} & \text{falls } a < 0 \end{cases}$$

Für einen 1 Byte Integer gilt dann:

Darstellung 8 Bit (BE)	Darstellung 8 Bit (LE)	unsigned ($m = 8$)	signed ($m = 7$)
0000 0000	0000 0000	0	0
0000 0001	1000 0000	1	1
⋮	⋮	⋮	⋮
0111 1111	1111 1110	127	127
1000 0000	0000 0001	128	−128
⋮	⋮	⋮	⋮
1111 1111	1111 1111	255	−1

Hierbei ist zu beachten, daß alle Rechenoperationen bei unsigned Integern modulo 2^m ablaufen und folglich *Ganzzahlüberläufe* auftreten können (Beim gegebenen Beispiel ist bei signed $127+1 = -128$ und bei unsigned $255 + 1 = 0$).

2.2.2 Gleitkommazahlen

Nun kommen wir zur Darstellung der *Gleitkommazahlen* (floating–point numbers). Jedes $x \in \mathbb{R}$ besitzt eine Darstellung der Form

$$x = (-1)^s \sum_{i=0}^{\infty} d_i b^{-i+e}$$

wobei $s \in \{0,1\}$, $e \in \mathbb{Z}$ und $d_i < b$ sind. Diese Darstellung ist keineswegs eindeutig, was man sich leicht am Beispiel $1.\overline{9} = 2$ klarmacht.
Wegen der endlichen Zahlenmenge des Rechners sind der *Exponent* e und die *Mantisse* $\sum_{i=0}^{m} d_i b^{-i}$ natürlicherweise beschränkt. Also ist diese Darstellung von der Gestalt

$$x = \pm d_0 d_1 \dots d_e . d_{e+1} \dots d_m \tag{2.2.2}$$

mit $m+1$ Stellen, und die Position des Dezimalpunktes wird durch die Zahl e festgelegt. Auch in dieser Darstellungsform gibt es Mehrdeutigkeiten. So ist etwa $10_{\text{bin}} \cdot 2^0 = 1.0_{\text{bin}} \cdot 2^1$. Insbesondere unterscheidet man *normalisierte Gleitkommazahlen*, bei denen $d_0 = 1$ und folglich $x = (-1)^s 1.d_1 d_2 \dots d_m \cdot 2^e$ ist, und *denormalisierte Gleitkommazahlen*, bei denen $d_0 = 0$ und somit $x = (-1)^s 0.d_1 d_2 \dots d_m \cdot 2^e$ ist. Um hier eine Übereinkunft zu schaffen, sieht der 1985 herausgegebene IEEE–754–Standard vor (siehe etwa [PTVF]), dem wir uns anschließen, grundsätzlich normalisierte Gleitkommazahlen zu verwenden und diese nur durch wenige denormalisierte Gleitkommazahlen „am unteren Ende“ zu ergänzen. Insgesamt gilt also

Exponent	Bedeutung
$e = e_{\min} - 1$	denormalisierte Gleitkommazahl $x = (-1)^s \sum_{i=1}^{m} d_i 2^{-i+e_{\min}}$
$e_{\min} \le e \le e_{\max}$	normalisierte Gleitkommazahl $x = (-1)^s \left(1 + \sum_{i=1}^{m} d_i 2^{-i}\right) 2^e$
$e = e_{\max} + 1$	infinity, overflow, underflow, NaN (not a number)

Betrachtet man die normalisierten Gleitkommazahlen, fragt man sich schnell, wie die Darstellung der 0 möglich ist, wenn $d_0 = 1$ vorgegeben ist. Hier ist die Rolle des Exponenten wichtig.
Wenn b_e die Anzahl der Bits des Exponenten ist, dann ist mit diesem ein vorzeichenloser Zahlbereich von $0, \dots, 2^{b_e} - 1$ darstellbar. Da man aber insbesondere auch mit negativen Exponenten arbeiten möchte, wurde im IEEE–754–Standard der „Exponent Bias“ eingeführt.
Dieser besagt, daß die interne unsigned Darstellung um $2^{b_e-1} - 1$ ins Negative verschoben wird, d.h., der Zahlbereich liegt nun bei $-2^{b_e-1} + 1, \dots, 2^{b_e} - 2^{b_e-1}$.
Der kleinste und größte Exponent wird für die Zahl 0 und das Symbol ∞ reserviert. Auf Speicherebene entspricht das einem nur mit Nullen bzw. nur mit Einsen besetzten Exponenten.
Folglich liegt der vom Exponenten darstellbare Zahlenbereich bei $-2^{b_e-1} + 2, \dots, 2^{b_e} - 2^{b_e-1} - 1$.

Die Menge aller auf diese Weise im Rechner darstellbaren Gleitkommazahlen bezeichnen wir mit

$$\mathbb{M}(b, m, e_{\min}, e_{\max}). \tag{2.2.2a}$$

Die technischen Daten der standardisierten Datentypen sind in folgender Tabelle zusammengefasst. Die zusätzliche Mantissenstelle folgt aus der Tatsache, daß stets $d_0 = 1$ und die 0 eine spezielle Darstellung hat, die dies konsistent macht.

Eigenschaften	single precision/float	double precision/double
Speichergröße	32 bit = 4 byte	64 bit = 8 byte
Bits der Mantisse	23	52
Stellen der Mantisse	24	53
Bits des Exponenten	8	11
Zahlbereich des Exponenten	$-126, \ldots, 127$	$-1022, \ldots, 1023$
Zahlbereich normalisiert	$1.2 \cdot 10^{-38}, \ldots, 3.4 \cdot 10^{38}$	$2.2 \cdot 10^{-308}, \ldots, 1.8 \cdot 10^{308}$
kleinste positive denorm. Zahl	$1.4 \cdot 10^{-45}$	$4.9 \cdot 10^{-324}$
Genauigkeit für Dezimalzahlen	$7-8$ Stellen	$15-16$ Stellen

2.2.3 Rundung und Maschinengenauigkeit

Da $\mathbb{M}$ endlich ist, muß man im allgemeinen Eingabedaten durch Maschinenzahlen approximieren. Dazu benutzt man folgende *Reduktionsabbildung*

$$\mathrm{fl} : \mathbb{R} \to \mathbb{M}(b, m, e_{\min}, e_{\max}),$$

die jedem $x \in \pm[x_{\min}, x_{\max}]$ eine Maschinenzahl zuordnet, wobei $\pm[x_{\min}, x_{\max}]$ die Menge der minimal bzw. maximal darstellbaren Zahlen bezeichnet. Dieses Vorgehen nennt man *Rundung*. Da jedes $x \in \pm[x_{\min}, x_{\max}]$ eine Darstellung $x = \pm(\sum_{i=0}^{\infty} d_i b^{-i}) b^e$ besitzt, definiert man als *Standardrundung*

$$\mathrm{fl}(x) := \begin{cases} \pm\left[1 + \left(\sum_{i=1}^{m} d_i b^{-i}\right)\right] b^e & \text{falls } d_{m+1} < \frac{b}{2} \\ \pm\left[1 + \left(\sum_{i=1}^{m} d_i b^{-i}\right) + b^{-m}\right] b^e & \text{falls } d_{m+1} \geq \frac{b}{2}, \end{cases} \tag{2.2.3}$$

was bedeutet, daß die letzte Stelle der Mantisse um eins erhöht oder beibehalten wird, je nachdem ob die folgende Ziffer $\geq \frac{b}{2}$ oder $< \frac{b}{2}$ ist. Ferner setzt man meist $\mathrm{fl}(x) = 0$, falls $|x| < x_{\min}$ bzw. im Falle $|x| > x_{\max}$ gibt man in der Regel `OVERFLOW` aus und läßt den Algorithmus abbrechen. Manchmal setzt man hier aber auch $\mathrm{fl}(x) = \mathrm{sign}(x) x_{\max}$. Wir verwenden in dieser Vorlesung immer die Standardrundung. Für diese gilt gemäß der Definition (2.2.3) $|\mathrm{fl}(x) - x| \leq \frac{b^{-m}}{2} b^e$ woraus sich wegen des *relativen Rundungsfehlers*

$$\left|\frac{\mathrm{fl}(x) - x}{x}\right| \leq \frac{\frac{b^{-m}}{2} b^e}{b^{e-1}} = \frac{b^{1-m}}{2} =: \mathrm{eps} \tag{2.2.4}$$

ergibt. eps bezeichnet man daher auch als *relative Maschinengenauigkeit*. Sie charakterisiert das *Auflösungsvermögen* des Rechners. Da aus (2.2.4)

$$\mathrm{eps} = \min\{\delta > 0 \mid \mathrm{fl}(1+\delta) > 1\} \tag{2.2.5}$$

folgt, ist eps die untere Grenze der Zahlen, die zu eins addiert von der Rundung noch wahrgenommen wird. Also besagt (2.2.4), daß es ein $\varepsilon > 0$ gibt mit der Eigenschaft $|\varepsilon| \leq \mathrm{eps}$ und

$$\frac{\mathrm{fl}(x) - x}{x} = \varepsilon, \tag{2.2.6}$$

was äquivalent zu $\mathrm{fl}(x) = x(1+\varepsilon)$ ist.

Gleitpunktarithmetik und Fehlerverstärkung bei den elementaren Rechenoperationen

Ein Algorithmus besteht aus einer Folge arithmetischer Operationen $(+, -, \cdot, \div)$, mit denen Maschinenzahlen verknüpft werden. Ein Problem dabei ist, daß für $x, y \in \mathbb{M}(b, m, e_{\min}, e_{\max})$ und $* \in \{+, -, \cdot, \div\}$ im allgemeinen $x * y \notin \mathbb{M}(b, m, e_{\min}, e_{\max})$ ist. Also müssen die üblichen Operationen $*$ durch geeignete Gleitpunktoperationen $\circledast$ (floating point operations, *flops*) ersetzt werden. Dies nennt man *Pseudoarithmetik*. Die Realisation erfolgt im allgemeinen dadurch, daß man intern über die Mantissenlänge hinaus weitere Stellen mitführt, nach Exponentenausgleich mit diesen Stellen genau rechnet, dann normalisiert und schließlich rundet. So erfüllt man die Forderung

$$x \circledast y = \mathrm{fl}(x * y) \tag{2.2.7}$$

für $x, y \in \mathbb{M}(b, m, e_{\min}, e_{\max})$. Mit (2.2.6) heißt das

$$x \circledast y = (x * y)(1 + \varepsilon)$$

für $x, y \in \mathbb{M}(b, m, e_{\min}, e_{\max})$ und ein $\varepsilon > 0$ mit der Eigenschaft $|\varepsilon| \leq \mathrm{eps}$. Wir nehmen im folgenden stets an, daß (2.2.7) gilt. Man beachte aber, daß die Pseudoarithmetik zum Teil unliebsame Konsequenzen hat, z.B. geht die Assoziativität der Addition verloren: $(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$. Seien für ein Beispiel hierfür mit $m = 8$:

$$\begin{aligned} a &:= 0.23371258_{10} - 4 \\ b &:= 0.33678429_{10}2 \\ c &:= -0.33677811_{10}2. \end{aligned}$$

Dann gilt

$$\begin{aligned} a \oplus (b \oplus c) &= 0.23371258_{10} - 4 \oplus 0.61800000_{10} - 3 \\ &= 0.64137126_{10} - 3, \\ (a \oplus b) \oplus c &= 0.33678452_{10}2 \ominus 0.33677811_{10}2 \\ &= 0.64100000_{10} - 3. \end{aligned}$$

Das exakte Resultat jedoch ist

$$a + b + c = 0.641371258_{10} - 3.$$

Auch die Distributivität geht verloren: $(x \oplus y) \odot (x \oplus y) \neq (x \odot x) \oplus (x \odot y) \oplus (y \odot x) \oplus (y \odot y)$. Das heißt, insbesondere die Reihenfolge der Operationen spielt eine Rolle.

Forderung (2.2.7) besagt, daß jede einzelne Gleitpunktoperation im Rahmen der Maschinengenauigkeit bleibt. In einem Algorithmus oder Programm werden aber eine Vielzahl solcher Gleitpunktoperationen ausgeführt. Nun ergibt sich natürlich die Frage, inwieweit sich solche eingeschleppten Fehler bei einer nachfolgenden Operation *verstärken* oder *abschwächen*. Dazu betrachten wir zunächst den Fall, daß man anstelle von Gleitpunktoperationen $(\circledast)$ exakt rechnet $(*)$, aber mit *gestörten* Daten, um die *Verstärkungswirkung* von Operationen zu untersuchen.

Seien δ_x, δ_y relative Fehler von $\tilde{x}, \tilde{y}$ gegenüber exakten Daten x, y, also

$$\frac{\tilde{x} - x}{x} = \delta_x, \quad \frac{\tilde{y} - y}{y} = \delta_y,$$

was äquivalent zu $\tilde{x} = x(1 + \delta_x)$ bzw. $\tilde{y} = y(1 + \delta_y)$ mit $|\delta_x|, |\delta_y| \leq \varepsilon < 1$ ist.

Beispiel 2.2.8. Wir betrachten die Multiplikation $f(x,y) = xy$ mit $\kappa_{\text{rel}} = 1$ aus Beispiel 2.1.7. Es gilt

$$\left|\frac{f(\tilde{x},\tilde{y}) - f(x,y)}{f(x,y)}\right| = \left|\frac{\tilde{x}\tilde{y} - xy}{xy}\right| \dot{\leq} \kappa_{\text{rel}} \left(\left|\frac{\tilde{x}-x}{x}\right| + \left|\frac{\tilde{y}-y}{y}\right|\right) \leq 2\varepsilon.$$

Falls also $|\delta_x|, |\delta_y| \leq \text{eps}$ sind, dann gilt

$$\left|\frac{\tilde{x}\tilde{y} - xy}{xy}\right| \leq 2\text{eps},$$

was bedeutet, daß bei der Multiplikation der relative Fehler im Rahmen der Maschinengenauigkeit bleibt. Die Division hat ähnliche Eigenschaften. ∎

Beispiel 2.2.9. Wir betrachten nun die Addition $f(x,y) = x + y$, deren relative Kondition

$$\kappa_{\text{rel}} = \max\left\{\frac{|x|}{|x+y|}, \frac{|y|}{|x+y|}\right\}$$

im Falle $x \approx -y$ sehr groß sein kann, wie wir bereits in Beispiel 2.1.9 gesehen haben. Es gilt

$$\left|\frac{(\tilde{x}+\tilde{y}) - (x+y)}{x+y}\right| \dot{\leq} \kappa_{\text{rel}} \left(\left|\frac{\tilde{x}-x}{x}\right| + \left|\frac{\tilde{y}-y}{y}\right|\right) \leq 2\varepsilon\kappa_{\text{rel}}.$$

Das heißt, die Addition von Zahlen entgegengesetzter Vorzeichen ist problematisch, denn relative Fehler können sich enorm verstärken. ∎

Beispiel 2.2.10. Seien

$$\begin{aligned} x &= 0.1234\,67\star \\ y &= 0.1234\,56\star. \end{aligned}$$

Das Symbol $\star$ stehe für eine Störung, die hier offenbar jeweils an siebter Stelle vorliegt. Betrachte nun

$$x - y = 0.0000\,11\star.$$

Geschrieben in Gleitpunktarithmetik, ist dieser Wert bereits an der dritten Stelle gestört (*Auslöschung*). Hier ist also $\kappa_{\text{rel}} \approx 10^4$, d.h. kleine Störungen in x und y können zu großen Störungen in $x - y$ führen. ∎

Es ist zu beachten, daß man einem Ergebnis nicht ansehen kann, ob im Algorithmus Auslöschungen stattgefunden haben. Daher sollte man, wann immer es möglich ist, die Subtraktion gleich großer Zahlen vermeiden!

2.3 Stabilität eines Algorithmus

Die mathematische Auswertung eines Problems ist meistens auf verschiedene Arten und Weisen möglich. Dabei sollten Algorithmen bevorzugt werden, bei denen sich möglichst wenig Fehler akkumulieren können. Aus diesem Grunde heißt ein Algorithmus *stabil* oder *gutartig*, wenn die im Laufe der Rechnung erzeugten Fehler im Rahmen des durch die Kondition des Problems bedingten *unvermeidbaren* Fehlers bleiben. Liegt also ein gut–konditioniertes Problem vor, das mit einem stabilen Algorithmus bearbeitet wird, so bleiben die Fehler bei der Rechnung kontrolliert klein. In allen anderen Fällen lassen sich Fehler im allgemeinen nicht kontrollieren.

Beispiel 2.3.1. Wir setzen Beispiel 2.1.3 fort, in dem es darum ging, die kleinere der beiden Nullstellen von $u^2 - 2a_1u + a_2 = 0$ zu berechnen, wobei neben $a_1, a_2 > 0$ reelle Zahlen hier verstärkt $a_1^2 \gg a_2$ angenommen sei, sodaß es zu keiner Auslöschung kommen kann. Die Lösung ist durch

$$u^* = a_1 - \sqrt{a_1^2 - a_2} = \frac{a_2}{a_1 + \sqrt{a_1^2 - a_2}}$$

gegeben. Wir betrachten zwei verschiedene Algorithmen, um diese zu berechnen.

Algorithmus I	Algorithmus II
$y_1 = a_1 \cdot a_1$	$y_1 = a_1 \cdot a_1$
$y_2 = y_1 - a_2$	$y_2 = y_1 - a_2$
$y_3 = \sqrt{y_2}$	$y_3 = \sqrt{y_2}$
$u^* = y_4 = a_1 - y_3$	$y_4 = a_1 + y_3$
	$u^* = y_5 = \frac{a_2}{y_4}$

Offenbar sind beide Terme mathematisch äquivalent. Unsere Frage lautet nun aber, was der Unterschied zwischen beiden Algorithmen bei der Rechnung ist, bzw. ob diese numerisch stabil sind.

Für die Kondition von u^* bezüglich y_3 in Algorithmus I gilt gemäß (2.1.5)

$$\left|\frac{\partial u^*}{\partial y_3} \cdot \frac{y_3}{u^*}\right| = \left|\frac{-\sqrt{a_1^2 - a_2}}{a_1 - \sqrt{a_1^2 - a_2}}\right| = \left|\frac{(a_1 + \sqrt{a_1^2 - a_2})\sqrt{a_1^2 - a_2}}{a_2}\right| \gg 1.$$

Daher ist Algorithmus I sehr schlecht konditioniert. Betrachten wir in Algorithmus II die gleiche Stelle, so sehen wir:

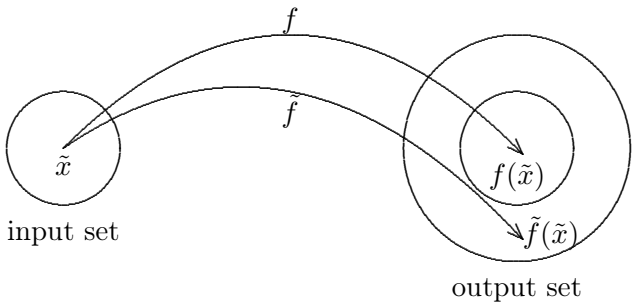
$$\left|\frac{\partial u^*}{\partial y_3} \cdot \frac{y_3}{u^*}\right| = \left|\frac{-a_2}{(a_1 + \sqrt{a_1^2 - a_2})^2} \cdot \frac{\sqrt{a_1^2 - a_2}(a_1 + \sqrt{a_1^2 - a_2})}{a_2}\right| = \left|\frac{\sqrt{a_1^2 - a_2}}{a_1 + \sqrt{a_1^2 - a_2}}\right| \leq 1.$$

Hier liegt also gute Kondition vor. Obwohl beide Algorithmen mathematisch äquivalent sind, ist der erste numerisch instabil, der zweite aber stabil. In Algorithmus I tritt in Schritt 4 Auslöschung auf. In unserer Situation ist Algorithmus II dort numerisch stabil. Dies kann sich aber ändern, denn für $a_1^2 \approx a_2$ tritt in beiden Algorithmen im 2. Schritt Auslöschung auf. ∎

Zwei wesentliche Möglichkeiten, die Akkumulation von Fehlern im Laufe eines Algorithmus einzuschätzen, bilden die *Vorwärts–* und *Rückwärtsfehleranalyse*. Bei der Vorwärtsanalyse (forward analysis) wird die Menge $\tilde{f}(\tilde{x})$ aller durch Eingabefehler und Fehler im Algorithmus gestörten Resultate betrachtet. Diese sind ein Teil der eigentlichen Resultatmenge $f(\tilde{x})$. Wir nennen einen Algorithmus stabil im Sinne der Vorwärtsanalyse, wenn die Menge $\tilde{f}(\tilde{x})$ von derselben Größenordnung wie $f(\tilde{x})$ ist:

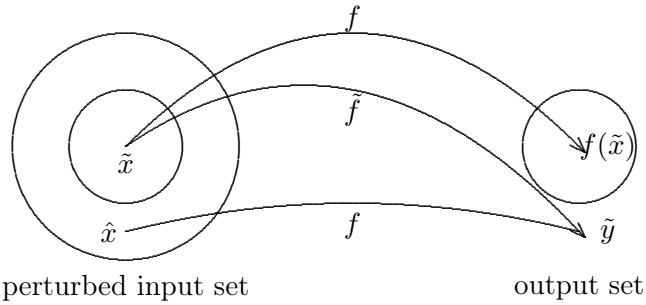
$$\frac{\|\tilde{f}(\tilde{x}) - f(\tilde{x})\|}{\|f(\tilde{x})\|} \leq \sigma \cdot \kappa_{\text{rel}} \cdot \text{eps},$$

σ also klein genug ist.



Die Rückwärtsanalyse (backward analysis) die auf [W] zurückgeht, beruht auf dem Prinzip, daß man sämtliche im Laufe der Rechnung auftretenden Fehler als Ergebnis *exakter* Rechnungen zu

geeignet gestörten Daten ansieht. Daraufhin schätzt man den Strörungsgrad der Daten und die Kondition des Problems ab, um so eine Abschätzung für den Gesamtfehler zu erhalten.



Dazu folgendes

Beispiel 2.3.2. Seien x_1, x_2, x_3 Maschinenzahlen einer Rechenmaschine mit Genauigkeit eps. Berechne die Summe $f(x_1, x_2, x_3) = (x_1 + x_2) + x_3$. Also gilt

$$\tilde{f}(x_1, x_2, x_3) = \big((x_1 + x_2)(1 + \varepsilon_1) + x_3\big)(1 + \varepsilon_2),$$

wobei $|\varepsilon_i| \leq$ eps sei. In erster Näherung ist dies gleich

$$x_1(1 + \varepsilon_1 + \varepsilon_2) + x_2(1 + \varepsilon_1 + \varepsilon_2) + x_3(1 + \varepsilon_2) =: \hat{x}_1 + \hat{x}_2 + \hat{x}_3 \tag{2.3.3}$$

Nun kann man das fehlerbehaftete Resultat $\tilde{f}$ als exaktes Ergebnis zu gestörten Eingabedaten der Form $\hat{x}_i = x_i(1 + \delta_i)$ mit $|\delta_i| \leq 2$eps für $i = 1, 2$ und $|\delta_i| \leq$ eps für $i = 3$ auffassen. Der unvermeidbare Fehler in den Daten ist nach (2.1.5) durch

$$F_{\text{Daten}}(x) = \left|\frac{f(\tilde{x}) - f(x)}{f(x)}\right| \dot{\leq} \kappa_{\text{rel}} \sum_{i=1}^{3} \left|\frac{\tilde{x}_i - x_i}{x_i}\right| \leq 3\kappa_{\text{rel}}\text{eps}$$

beschränkt. Der durch die Rechnung bedingte Fehler läßt sich durch

$$F_{\text{Rechnung}}(x) = \left|\frac{f(\hat{x}) - f(x)}{f(x)}\right| \dot{\leq} \kappa_{\text{rel}} \sum_{i=1}^{3} \left|\frac{\hat{x}_i - x_i}{x_i}\right| \leq \kappa_{\text{rel}} \sum_{i=1}^{3} |\delta_i| \leq 5\kappa_{\text{rel}}\text{eps}$$

abschätzen. Also sind die Fehler F_{Rechnung} und F_{Daten} in der gleichen Größenordnung und f damit ein stabiler Algorithmus. Man beachte allerdings, daß κ_{rel} für bestimmte Werte x_i wieder sehr groß sein kann, was bedeutet, daß $F_{\text{Rechnung}} \gg$ eps werden kann. ∎

Das Ziel der in diesem zweiten Kapitel geführten Diskussion soll Grundlage für die vernünftige Einschätzung von Verfahrenskomponenten sein. Es sollte eine gewisse Sensibilität dafür entwickelt werden, wann und unter welchen Umständen es in Algorithmen zu fehlerhaften Resultaten kommen kann und in welcher Größenordnung diese liegen können. Hierfür haben wir einige Beispiele gesehen. Man kann häufig trotzdem nicht jeden einzelnen Schritt eines komplexeren Algorithmus' sezieren. Faustregeln sind

- Kenntnisse über die Kondition eines Problems sind oft für dessen Beurteilung und Interpretation von entscheidender Bedeutung;

- im Rahmen der Gleitpunktarithmetik sollte man Abfragen, ob eine Größe gleich Null oder ob zwei Zahlen gleich sind, vermeiden. Anstelle von `IF(x==y)` sollte man besser `IF(FABS(x-y)>0)` prüfen;
- soweit es geht, sollten Auslöschungseffekte vermieden werden;
- bei der Bildung von Summen können gewisse Reihenfolgen vorteilhafter als andere sein.

3 Direkte Verfahren zur Lösung linearer Gleichungssysteme

3.1 Vorbemerkungen

In diesem Kapitel behandeln wir Lösungsmethoden für lineare Gleichungssysteme. Eine Vielzahl von Problemen in den Naturwissenschaften wird auf das Lösen von linearen Gleichungsystemen zurückgeführt, so z.B. die Diskretisierung von Differentialgleichungen. Zunächst werden wir ein paar Begriffe und Bezeichnungen einführen. Mit $\mathbb{K}$ seien stets die reellen oder komplexen Zahlen gemeint; $\mathbb{K}^{m\times n}$ bezeichne die Menge aller Matrizen der Form

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

mit Einträgen in $\mathbb{K}$. Im folgenden lautet unsere grundsätzliche Aufgabe: Gegeben seien $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$. Man finde ein $x \in \mathbb{R}^n$, das die Gleichung $Ax = b$ erfüllt. Aus der Linearen Algebra wissen wir bereits, daß

$$Ax = b \quad \Leftrightarrow \quad \sum_{i=1}^{n} a_{ij}x_j = b_i, \quad i = 1, ..., n \tag{3.1.1}$$

ist. Grundlegend für die weitere Diskussion ist der folgende

Satz 3.1.2. Seien $A \in \mathbb{R}^{n\times n}$ und $b, x \in \mathbb{R}^n$. Dann sind äquivalent:

(i) $Ax = b$ besitzt genau eine Lösung $x \in \mathbb{R}^n$;

(ii) $\det(A) \neq 0$;

(iii) A ist regulär;

(iv) $\operatorname{rang}(A) = n$;

(v) $Ax = 0$ hat nur die triviale Lösung $x \equiv 0$. ∎

Im folgenden sei stets $\det(A) \neq 0$ angenommen. Prinzipiell könnte man die Lösung des linearen Gleichungssystems $Ax = b$ durch Anwendung der CRAMER'schen Regel ermitteln. Diese besagt, daß der Lösungsvektor x des linearen Gleichungssystems mit regulärer Koeffizientenmatrix ist durch

$$x_j = \frac{\det(A_j)}{\det(A)}$$

eindeutig bestimmt, wobei A_j die Matrix bezeichne, die man aus der Koeffizientenmatrix A erhält, indem man die j-te Spalte durch den Vektor b ersetzt. Die Lösung eines linearen Gleichungssystems mit dieser Methode erfordert die Berechnung von $n + 1$ Determinanten $\det(A_1), \ldots, \det(A_n), \det(A)$. Um diese zu berechnen, könnte man etwa die LEIBNIZ–Regel

$$\det(A) = \sum_{\sigma\in S_n} \operatorname{sign}\sigma \prod_{i=1}^{n} a_{i,\sigma(i)}$$

verwenden, die $n!$ Rechenoperationen erfordert. Daß die CRAMER'sche Regel zur Lösung eines Systems von mehr als drei Gleichungen für die Praxis ungeeignet ist, zeigt die folgende Tabelle. Dort stehen die Zeiten, die ein 40 Gflop–Rechner ($\hat{=} 4 \cdot 10^{10}$ flops/sec) für die Berechnung von x mittels CRAMERscher Regel für $n = 10, 12, 14, 16, 18, 20$ bräuchte.

n	10	12	14	16	18	20
Zeit	$9.9 \cdot 10^{-4}$ s	0.16 s	0.54 min	2.47 h	5.1 w	40.5 a

Im folgenden werden wir verschiedene Ansätze zur Lösung von linearen Gleichungssystemen $Ax = b$ kennenlernen, die ausnutzen, daß die Lösung von (3.1.1) identisch zur Lösung von $CAx = Cb$ mit $C \in \mathbb{R}^{n\times n}$ und $\det(C) \neq 0$ ist.

3.2 Dreiecksmatrizen und Rückwärtseinsetzen

Eine Matrix $R = (r_{ij})_{i,j=1,\ldots,n} \in \mathbb{R}^{n\times n}$ heißt *obere Dreiecksmatrix*, falls $r_{ij} = 0$ für $i > j$ gilt,

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{pmatrix}.$$

Analog gilt für eine untere Dreiecksmatrix

$$L = \begin{pmatrix} \ell_{11} & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ \ell_{n1} & \cdots & \cdots & \ell_{nn} \end{pmatrix}.$$

Insbesondere ist die Transponierte einer oberen Dreiecksmatrix eine untere Dreiecksmatrix und umgekehrt, also $R^T = L$ und $L^T = R$. Die einzigen Matrizen, die sowohl obere als auch untere Dreiecksgestalt haben, sind Diagonalmatrizen.

Wir stellen zunächst die Frage, für welchen Typ von A das Gleichungssystem $Ax = b$ besonders leicht lösbar ist. Falls A diagonal ist, ist der Fall trivial. Hat A obere Dreiecksgestalt, so ist (3.1.1) von der Form

$$\begin{aligned} r_{11}x_1 + \ldots + r_{1n}x_n &= b_1 \\ \ddots \quad \vdots &\quad \vdots \\ r_{nn}x_n &= b_n \end{aligned} \tag{3.2.1}$$

Da

$$\det(R) = \prod_{i=1}^{n} r_{ii}$$

gilt, ist (3.2.1) genau dann eindeutig lösbar, wenn $r_{ii} \neq 0$ für alle $i = 1, \ldots, n$ gilt. Dies läßt folgende Lösungsstrategie zu: Stelle die letzte Gleichung von (3.2.1) zu

$$x_n = \frac{b_n}{r_{nn}}$$

um. Durch sukzessives Einsetzen der Werte $x_n, x_{n-1}, \ldots$ von unten nach oben erhält man den Lösungsvektor des Systems. Hiermit läßt sich folgender Algorithmus herleiten.

Algorithmus 3.2.2 (Rückwärtseinsetzen)**.** Seien $R \in \mathbb{R}^{n\times n}$ eine obere Dreiecksmatrix mit $r_{ii} \neq 0$ für alle $i = 1, \ldots, n$ und $b \in \mathbb{R}^n$.

`FOR` $j = n, \ldots, 1$

$$x_j = \frac{b_j - \sum_{k=j+1}^{n} r_{jk} x_k}{r_{jj}}$$

`END.` ∎

Der *Rechenaufwand* für diesen Algorithmus beträgt für jedes $j = n, \ldots, 1$ je $n - j$ Multiplikationen/Additionen und eine Division. Insgesamt ergibt sich als Rechenaufwand

$$\sum_{j=1}^{n-1} (n - j) = \frac{n(n-1)}{2}$$

für die Multiplikationen bzw. Additionen und n und für die Divisionen. üblicherweise ist eine Addition viel schneller berechenbar als eine Punktoperation. Daher zählen wir nur die Multiplikationen und Divisionen als wesentliche Operationen. Weiterhin berücksichtigen wir nur die Terme von höchster Ordnung, d.h.

$$\frac{n(n-1)}{2} \doteq \frac{n^2}{2}.$$

Demnach beträgt der Rechenaufwand für das Rückwärtseinsetzen

$$\mathcal{O}\left(\frac{1}{2} n^2\right), \tag{3.2.3}$$

wobei n die Anzahl der Unbekannten bezeichnet. Das Verfahren für das Vorwärtseinsetzen läuft komplett analog. Im folgenden erarbeiten wir eine Strategie zur Lösung von $Ax = b$ für Matrizen A mit beliebiger Struktur und $\det(A) \neq 0$. Dazu transformieren wir A auf obere Dreiecksgestalt. Hierzu noch folgendes

Lemma 3.2.4. a) Seien R_1 und R_2 obere Dreiecksmatrizen, dann ist auch $R_1 R_2$ eine obere Dreiecksmatrix.

b) Ist R eine obere Dreiecksmatrix, so ist auch R^{-1} eine obere Dreiecksmatrix.

Analoge Aussagen gelten für untere Dreiecksmatrizen.

3.3 GAUSS–Elimination und LR–Zerlegung

Wir beginnen unsere Diskussion mit Methoden ohne *Pivotisierung*. Die bekannteste Methode, das System

$$Ax = b \tag{3.3.1}$$

mit $\det(A) \neq 0$ auf obere Dreiecksgestalt zu bringen, ist die GAUSS–Elimination. Die Idee hierbei ist, daß man die Lösung von (3.3.1) nicht verändert, wenn man ein Vielfaches einer Gleichung von einer anderen subtrahiert. Sei $a_{11} \neq 0$. Man subtrahiere nun geeignete Vielfache der ersten Zeile von A von den übrigen Zeilen, sodaß die resultierenden Einträge der ersten Spalte der Matrix A verschwinden, d.h.

$$A = A^{(1)} = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} \to \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix} = A^{(2)}.$$

Ist nun der (veränderte) Eintrag a_{22} von $A^{(2)}$ ungleich null, so wiederhole man das Verfahren für die zweite Spalte. Mit diesem Verfahren kann man A sukzessive auf obere Dreiecksgestalt bringen, ohne daß man die Lösung von (3.3.1) verändert. Damit dies gilt, muß man entsprechend auch die rechte Seite b umformen. Dazu kann man das System auch in eine einzige Matrix, eine *erweiterte Koeffizientenmatrix* (A, b) schreiben. Hieraus ergibt sich folgender Algorithmus.

Algorithmus 3.3.2 (Reduktion auf obere Dreiecksgestalt)**.** Seien $A \in \mathbb{R}^{n\times n}$ mit $\det(A) \neq 0$ und $b \in \mathbb{R}^n$.

```
FOR j = 1, ..., n − 1
    IF a_jj ≠ 0
        FOR i = j + 1, ..., n
```

$$z_i \leftarrow z_i - \frac{a_{ij}}{a_{jj}} z_j =: \ell_{ij} \tag{3.3.3}$$

```
        END
    END
END
```

■

Hierbei steht z_i für die i-te Zeile der jeweiligen Untermatrix $\tilde{A}^{(j)}$. Es empfiehlt sich, zur Prüfung, ob $a_{jj} \neq 0$ ist, nicht `IF(a[j][j]!=0)` abzufragen, sondern besser `IF((FABS(a[j][j])>0)` abzuprüfen. Das Resultat dieses Algorithmus hat die Form (R, c), wobei R eine obere Dreiecksmatrix ist. Dieser Algorithmus versagt allerdings, wenn $a_{jj} = 0$ für ein j ist. Diesen Fall behandeln wir später. Um ein Gefühl für das praktische Vorgehen des Algorithmus zu bekommen, betrachten wir folgendes

Beispiel 3.3.4. Gegeben seien

$$A = \begin{pmatrix} 2 & -1 & -3 & 3 \\ 4 & 0 & -3 & 1 \\ 6 & 1 & -1 & 6 \\ -2 & -5 & 4 & 1 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 1 \\ -8 \\ -16 \\ -12 \end{pmatrix}.$$

Die erweiterte Koeffizientenmatrix (A, b) lautet

$$(A, b) = \begin{pmatrix} 2 & -1 & -3 & 3 & 1 \\ 4 & 0 & -3 & 1 & -8 \\ 6 & 1 & -1 & 6 & -16 \\ -2 & -5 & 4 & 1 & -12 \end{pmatrix}.$$

Wir gehen den obigen Algorithmus schrittweise durch: Sei $j = 1$. Der Algorithmus berechnet

$$\begin{array}{c|c} i & \ell_{i1} \\ \hline 2 & 2 \\ 3 & 3 \\ 4 & -1 \end{array} \quad \rightarrow \quad \begin{pmatrix} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 4 & 8 & -3 & -19 \\ 0 & -6 & 1 & 4 & -11 \end{pmatrix}.$$

Nun folgt $j = 2$:

$$\begin{array}{c|c} i & \ell_{i2} \\ \hline 3 & 2 \\ 4 & -3 \end{array} \quad \rightarrow \quad \begin{pmatrix} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 10 & -11 & -41 \end{pmatrix}.$$

Und schließlich noch $j = 3$:

$$\begin{array}{c|c} i & \ell_{i3} \\ \hline 4 & 5 \end{array} \quad \rightarrow \quad \begin{pmatrix} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 0 & -46 & -46 \end{pmatrix} = (R, c).$$

Nun erhält man die Lösung

$$x = \begin{pmatrix} -\frac{9}{2} \\ 2 \\ -3 \\ 1 \end{pmatrix}$$

durch Rückwärtseinsetzen. In der Praxis ist es weiterhin üblich, die Elemente unter der Diagonalen, die wir hier als Null ausgegeben haben, mit den jeweiligen ℓ_{ij} Werten der nebenstehenden Tabellen zu überspeichern. ∎

Für die Gauss–Elimination ohne Pivotisierung ergibt sich folgender Algorithmus.

Algorithmus 3.3.5 (Gauss–Elimination ohne Pivotisierung)**.**

1.) Bestimme $(A, b) \rightarrow (R, c)$ gemäß Algorithmus 3.3.2.

2.) Löse $Rx = c$ gemäß Algorithmus 3.2.2.

∎

Bei der Implementierung braucht man durch geeignetes überspeichern der Einträge von A und Ablegen der ℓ_{ij} unterhalb der Diagonalen *keinen* zusätzlichen Speicherplatz.

Programmentwurf 3.3.6 (für Algorithmus 3.3.2)**.** Seien $A \in \mathbb{R}^{n\times n}$ mit $\det(A) \neq 0$ und $b \in \mathbb{R}^n$.
`FOR` $j = 1, \ldots, n-1$

`IF` $a_{jj} = 0$ `STOP`

`ELSE`

`FOR` $i = j+1, ..., n$

$$a_{ij} \leftarrow \frac{a_{ij}}{a_{jj}}; \qquad b_j \leftarrow b_i - a_{ij} b_j$$

`FOR` $k = j+1, ..., n$

$$a_{ik} \leftarrow a_{ik} - a_{ij} a_{jk}$$

`END`

`END`

```
    END

END
```

■

Der Rechenaufwand für diesen Algorithmus ist

$$\mathcal{O}\left(\frac{n^3}{3}\right), \tag{3.3.7}$$

da bei den drei Rechenschritten für Rückwärtseinsetzen und GAUSS–Elimination jeweils $n - j$ Operationen ausgeführt werden müssen.

Satz 3.3.8. Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär und der GAUSS–Algorithmus 3.3.6 durchführbar (d.h. es gilt $a_{jj}^{(j)} \neq 0$ für jedes j). Dann liefert Algorithmus 3.3.6 eine LR–Zerlegung von A, d.h.

$$A = LR \tag{3.3.9}$$

mit

$$L := \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 \\ \ell_{n1} & \cdots & \ell_{n,n-1} & 1 \end{pmatrix}$$

und

$$R := \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

sind. Diese Zerlegung ist eindeutig.

Beweis: Einen Beweis findet man z.B. in [DH]. ■

Da die Matrix L auf der Diagonalen nur Einsen enthält, also normiert ist, muß man diese bei der Implementierung nicht abspeichern. Wir geben die Matrizen L und R für obiges Beispiel noch einmal explizit an:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ -1 & -3 & 5 & 1 \end{pmatrix} \quad \text{und} \quad R = \begin{pmatrix} 2 & -1 & -3 & 3 \\ 0 & 2 & 3 & 5 \\ 0 & 0 & 2 & 7 \\ 0 & 0 & 0 & -46 \end{pmatrix}.$$

Mit der Faktorisierung (3.3.9) von A läßt sich Algorithmus 3.3.5 folgendermaßen schreiben:

Algorithmus 3.3.10.

1.) Bestimme $A = LR$ mit $Ax = LRx = Ly = b$.

2.) Löse $Ly = b$ durch Vorwärtseinsetzen.

3.) Löse $Rx = y$ durch Rückwärtseinsetzen.

■

Der Rechenaufwand für Algorithmus 3.3.10 beträgt für das Vorwärts– und Rückwärtseinsetzen je $\mathcal{O}\left(\frac{1}{2}n^2\right)$ und für die LR–Zerlegung $\mathcal{O}\left(\frac{1}{3}n^3\right)$, insgesamt also $\mathcal{O}\left(\frac{1}{3}n^3\right)$.

Wir kommen nun zu Methoden mit Pivotisierung. Satz 3.3.8 gilt nur unter der Voraussetzung, daß $a_{jj}^{(j)} \neq 0$ für alle j gilt. Schon das Beispiel

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

zeigt, daß dies nicht immer gelten muß. Weiter kann es problematisch sein, wenn einige Matrixeinträge sehr klein im Vergleich zu anderen sind. Ein eindrucksvolles Beispiel hierfür findet man in [DH], p. 10. Andererseits ändert sich die Lösung des Gleichungssystems $Ax = b$ nicht, wenn man die Gleichungen in $Ax = b$, d.h. die Zeilen von A und die entsprechenden Komponenten von b vertauscht. Im obigen Beispiel liefert bereits die Vertauschung beider Zeilen eine obere Dreiecksmatrix. Ist allgemeiner $\det(A) \neq 0$, so enthält die erste Spalte von A mindestens ein Element $a_{i1} \neq 0$. Falls also $a_{11} = 0$ ist, vertausche man die erste Zeile mit der i-ten Zeile, damit eine Anwendung des obigen Prinzips ermöglicht wird. Dies legt die folgende Vertauschungsstrategie nahe. Man bestimme in der zu eliminierenden Spalte zuerst das betragsmäßig größte Element und vertausche die Zeilen entsprechend. Dies nennt man *Spaltenpivotisierung.*

Programmentwurf 3.3.11 (LR–Zerlegung mit Pivotisierung)**.** Für $j = 1, \ldots, n-1$ berechne p mit $j \leq p \leq n$, sodaß

$$|a_{pj}| = \max_{j \leq p \leq n} |a_{ij}|$$

ist. Setze dann

$$r_j = p,$$

um die Pivotzeile zu merken. Vertausche dann Zeile j mit Zeile p. Für $i = j+1, \ldots, n$ berechne

$$a_{ij} \leftarrow \frac{a_{ij}}{a_{jj}}.$$

Für $k = j+1, \ldots, n$ berechne

$$a_{ik} \leftarrow a_{ik} - a_{ij}a_{jk}.$$

■

Die Vertauschung wird mittels elementarer Permutationsmatrizen dargestellt. Sie entstehen aus der Einheitsmatrix durch das Vertauschen der Zeile i mit der Zeile k. Es ist

$$P_{ik} = \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 0 & \cdots & \cdots & \cdots & 1 & & \\ & & & 1 & & & & & \\ & & \vdots & & \ddots & & \vdots & & \\ & & & & & 1 & & & \\ & & 1 & \cdots & \cdots & \cdots & 0 & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{pmatrix} \tag{3.3.12}$$

Damit entspricht in Programmentwurf 3.3.11 r_j den Permutationsmatrizen P_{j,r_j}, und es folgt

$$P := \prod_{j=1}^{n-1} P_{j,r_j}. \tag{3.3.13}$$

Beachte

$$\det(P_{ik}) = \begin{cases} +1 & \text{falls } i = k \\ -1 & \text{falls } i \neq k \end{cases} \tag{3.3.14}$$

Nun haben wir folgenden

Satz 3.3.15. Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär. Dann existiert eine Permutationsmatrix P, eine dazu eindeutige normierte untere Dreiecksmatrix L und eine obere Dreiecksmatrix R mit

$$PA = LR. \tag{3.3.16}$$

Beweis: Einen Beweis findet man in [DH] und in Standard–Numerik–Büchern. ∎

Also haben wir folgenden

Algorithmus 3.3.17 (GAUSS–Elimination mit Pivotisierung)**.**

1.) Bestimme $PA = LR$ mit $PAx = LRx = Pb$.

2.) Löse $Ly = Pb$ durch Vorwärtseinsetzen.

3.) Löse $Rx = y$ durch Rückwärtseinsetzen.

∎

Der Rechenaufwand ist immer noch $\mathcal{O}\left(\frac{1}{3}n^3\right)$. Anstelle der Spaltenpivotisierung kann man auch eine Zeilen– oder sogar eine Totalpivotisierung (d.h. Zeilen– und Spaltenpivotisierung) durchführen, wobei letzteres meist nicht angewendet wird, da es in der Regel zu aufwendig ist.

Wir kommen nun zu *äquilibrierung* und *Skalierung*. Eine Multiplikationen der Zeilen des Gleichungssystems $Ax = b$ mit Skalaren ändert die Lösung nicht, wohl aber die Reihenfolge der Vertauschung. Anstelle von $Ax = b$ löst man dann ein System der Form $DAx = Db$, wobei $D = \operatorname{diag}(d_1, \ldots, d_n)$ eine Diagonalmatrix und $d_i = \left(\sum_{j=1}^{n} |a_{ij}|\right)^{-1}$ für $i = 1, \ldots, n$ ist. Dies nennt man *Zeilenäquilibrierung*. Entsprechend könnte man eine Spaltenäquilibrierung oder beides durchführen. Hierfür gibt es aber kein festes Verfahren. Daher sind solche Skalierungsfragen in Paketen wie LAPACK meist dem User überlassen.

Einige Konsequenzen und Anwendungen von Satz 3.3.15 sind etwa die vereinfachte Berechnung von Determinanten. Aus $PA = LR$ folgt

$$\det(P) \cdot \det(A) = \det(L) \cdot \det(R) = \det(R).$$

Mit (3.3.14) ist $\sigma := \det(P) = (-1)^s$, wobei s die Gesamtanzahl der Zeilenvertauschungen bezeichne. Hieraus folgt nun

$$\det(A) = \sigma \det(R) = \sigma \prod_{j=1}^{n} r_{jj}. \tag{3.3.18}$$

Hierbei beträgt der Aufwand nur $\mathcal{O}\left(\frac{1}{3}n^3\right)$ Operationen im Vergleich zu $\mathcal{O}(n!)$ Operationen bei der Anwendung der LEIBNIZ–Regel. (Für $n = 20$ bedeutet dies auf einem 40 Gflop–Rechner 0.1 µs gegenüber 40.5 a.) Für den Fall mehrerer rechter Seiten b muß man das Gleichungssystem $Ax = b$ nicht mehrfach mit verschiedenen rechten Seiten lösen, sondern benötigt nur einmal die

LR–Zerlegung von A mit dem Aufwand $\mathcal{O}\left(\frac{1}{3}n^3\right)$. Der Rest erfolgt über Vorwärts– und Rückwärtseinsetzen mit einem Aufwand von $\mathcal{O}\left(\frac{1}{2}n^2\right)$. Auch die Berechnung der Inversen ist durch das Lösen von Gleichungssystemen möglich. Es gilt, daß

$$PA = LR \quad \Leftrightarrow \quad A^{-1} = R^{-1}L^{-1}P$$

ist. Algorithmisch bedeutet dies die Lösung von n Systemen $Ax^i = e^i$ für $i = 1, \ldots, n$. Dann sind die x^i die Spalten von A^{-1}. Der Aufwand hierbei ist $\mathcal{O}\left(\frac{n^3}{3}\right)$ für die LR–Zerlegung, sowie n–mal Vor– und Rückwärtseinsetzen mit je $\mathcal{O}\left(\frac{n^2}{2}\right)$, also insgesamt $\mathcal{O}\left(\frac{4n^3}{3}\right)$. In der Numerik wird in der Regel die Inverse einer Matrix *nie explizit* benötigt. Der Ausdruck $x = A^{-1}b$ ist stets prozedural so zu interpretieren, daß x die Lösung von $Ax = b$ ohne explizite Berechnung von A^{-1} ist.

Bemerkung 3.3.19.

a) Die LR–Zerlegung aus (3.3.18) läßt sich auch blockweise anwenden, d.h. für $a_{ij} \in \mathbb{R}^{r\times r}$ mit $r < n$.

b) Insbesondere läßt sich die LR-Zerlegung verwenden, wenn A eine Block– oder Tridiagonalmatrix ist.

■

3.4 Cholesky–Verfahren

Obige LR–Zerlegung ist prinzipiell für beliebige Gleichungssysteme mit nichtsingulären Matrizen A anwendbar. Viele Anwendungsbeispiele liefern jedoch Matrizen, die bestimmte Struktureigenschaften haben. Oft liegt zum Beispiel der Fall vor, daß A symmetrisch und positiv definit ist.

Definition 3.4.1. Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt *symmetrisch positiv definit*, falls

$$A = A^T \qquad \text{und} \qquad x^T Ax = \langle x, Ax\rangle > 0$$

für alle $x \in \mathbb{R}^n$, $x \not\equiv 0$ ist. Der Begriff *symmetrisch negativ definit* ist analog definiert. ■

Beispiele für symmetrisch positive Matrizen sind etwa $A = I$, $A := B^T B$ mit $B \in \mathbb{R}^{m\times n}$ und $m > n$ sowie $\operatorname{rang}(B) = n$. Diese Problematik werden wir in Kapitel 4 bei den linearen Ausgleichsproblemen wieder aufgreifen. Wir gehen weiter mit einigen

Eigenschaften 3.4.2. Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit. Dann gilt:

i) Ist A invertierbar, so ist auch A^{-1} symmetrisch positiv definit;

ii) $a_{ii} > 0$ für alle $i = 1, \ldots, n$;

iii) jede Hauptuntermatrix

$$\tilde{A} = \begin{pmatrix} a_{i1,i1} & \cdots & a_{i1,i\ell} \\ \vdots & & \vdots \\ a_{i\ell,i1} & \cdots & a_{i\ell,i\ell} \end{pmatrix}$$

für $1 \le i_1 \le i_\ell \le n$ ist wieder symmetrisch positiv definit;

iv) es gilt $\max\limits_{i=1,\ldots,n}(a_{ii}) \geq |a_{p,q}|$ für $p, q = 1, \ldots, n$.

v) A hat nur positive reelle Eigenwerte;

vi) bei der LR–Zerlegung ohne Pivotisierung ist die Restmatrix wieder symmetrisch positiv definit. ∎

Wesentlich ist das folgende Resultat.

Satz 3.4.3. Jede symmetrisch positiv definite Matrix $A \in \mathbb{R}^{n\times n}$ besitzt eine eindeutige Zerlegung

$$A = LDL^T, \tag{3.4.4}$$

wobei

$$L = \begin{pmatrix} 1 & & \\ * & \ddots & \\ * & * & 1 \end{pmatrix} \quad \text{und} \quad D = \operatorname{diag}(d_{11}, \ldots, d_{nn}) \tag{3.4.5}$$

mit $d_{ii} > 0$ für alle $i = 1, \ldots, n$ ist. Umgekehrt ist jede Matrix der Form LDL^T mit den Eigenschaften 3.4.5 wieder symmetrisch positiv definit.

Beweis: Wir beginnen mit der Notwendigkeit der Bedingung. Sei A symmetrisch positiv definit. Nach 3.4.2 (i) ist A invertierbar. Weiter existiert nach 3.4.2 (ii) und 3.4.2 (iv) eine eindeutige LR–Zerlegung von A (ohne Pivotisierung) der Form $A = LR$. Sei nun $D := \operatorname{diag}(r_{11}, \ldots, r_{nn})$ und $M := D^{-1}R$. Dann gilt $A = LR = LDM$ und $A^T = M^T DL^T$. Wegen $A = A^T$ und der Eindeutigkeit der Zerlegung ist $L = M^T$, also $A = LDL^T$. Weiter ist $r_{ii} > 0$ für alle $i = 1, \ldots, n$, denn

$$0 < x^T Ax = (x, LDL^T x) = (L^T x, DL^T x) = (y, Dy) = \sum_{i=1}^{n} d_{ii} y_i^2 \tag{3.4.6}$$

womit die Behauptung folgt. Die Bedingung ist hinreichend. Die Symmetrie ist klar. Der Rest folgt mit (3.4.6). ∎

Bemerkung 3.4.7.

i) Es gilt

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

mit $\tilde{L} := LD^{\frac{1}{2}}$ und $D^{\frac{1}{2}} := \operatorname{diag}(d_{11}^{\frac{1}{2}}, \ldots, d_{nn}^{\frac{1}{2}})$. Die Matrix L heißt CHOLESKY–Zerlegung von A und ist eine untere Dreiecksmatrix.

ii) Beachte, daß wegen 3.4.2 (ii) und 3.4.2 (iv) keine Pivotisierung notwendig ist. Dies ist auch nicht sinnvoll, da eine Vertauschung von Zeilen die Symmetriestruktur zerstören würde.

iii) Da $A = A^T$, ist A bereits durch $\frac{n(n+1)}{2}$ Einträge a_{ij} mit $i \leq j$ vollständig bestimmt.

Eine mögliche Anwendung der CHOLESKY–Zerlegung bietet die Matrix, die durch Diskretisierung einer elliptischen Differentialgleichungen entsteht.

Die Konstruktion der CHOLESKY–Zerlegung von A beruht auf einer geschickten Reihenfolge der Berechnung der Einträge von L durch elementweise Auswertung von $A = LDL^T$ der Form

$$\begin{pmatrix} \ell_{11} & & 0 \\ \vdots & \ddots & \\ \ell_{n1} & \cdots & \ell_{nn} \end{pmatrix} \begin{pmatrix} d_{11} & & 0 \\ & \ddots & \\ 0 & & d_{nn} \end{pmatrix} \begin{pmatrix} \ell_{11} & \ldots & \ell_{n1} \\ & \ddots & \vdots \\ 0 & & \ell_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}. \tag{3.4.8}$$

Für das untere Dreieck $i \geq k$ gilt

$$a_{ik} = \sum_{j=1}^{n} \ell_{ij} d_{jj} \ell_{kj} = \sum_{j=1}^{k-1} \ell_{ij} d_{jj} \ell_{kj} + \ell_{ik} d_{kk} \ell_{kk}.$$

Dies ist äquivalent zu

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

Für $k = 1$ gilt

$$\ell_{i1} d_{11} = a_{i1},$$

womit für $i = 1$ $d_{11} = a_{11}$ folgt, da $\ell_{11} = 1$, und für $i > 1$ $\ell_{i1} = \frac{a_{i1}}{d_{11}}$. Damit haben wir die erste Spalte von L berechnet. Setzen wir nun voraus, daß alle Einträge von L bis Spalte $k-1$, d.h. alle ℓ_{ij} für $j > 1$ und $j < k-1$, sowie d_{ii} für $i \leq k-1$ bekannt sind. Damit berechnet sich die k-te Spalte von L im Fall $i = k$ als

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

und im Fall $i > k$ als

$$\ell_{ik} = (a_{ik} - \sum_{j=1}^{k-1} \ell_{ij} d_{jj} \ell_{kj}) / d_{kk}.$$

Das Verfahren ist immer durchführbar, da nach (3.4.5) $d_{kk} > 0$ ist. Somit haben wir einen

Programmentwurf 3.4.9 (CHOLESKY–Verfahren)**.**

FOR $k = 1, \ldots, n$

$$\text{diag} \leftarrow a_{kk} - \sum_{j<k} a_{kj}^2 a_{jj}$$

END

IF diag $< 10^{-5}\, a_{kk}$ STOP

ELSE

$$a_{kk} \leftarrow \text{diag}$$

FOR $i = k+1, \ldots, n$

$$a_{ik} \leftarrow (a_{ik} - \sum_{j<k} a_{ij} a_{jj} a_{kj}) / a_{kk}$$

END

END ∎

Bemerkung 3.4.10.

i) Die Lösung von $Ax = b \Leftrightarrow LDL^T x = b$ reduziert sich wieder auf $Ly = b$ mit $y = DL^T x$ (vorwärtseinsetzen) und $L^T x = D^{-1} y$.

ii) Obiger Programmentwurf enthält einen numerischen Test auf positiv–Definitheit.

iii) Der Aufwand ist etwa die Hälfte der LR–Zerlegung, also $\mathcal{O}\left(\frac{1}{6} n^3\right)$.

∎

3.5 Bandmatrizen

In den Anwendungen gibt es oft Matrizen $A \in \mathbb{R}^{n\times n}$, die dünn besetzt (sparse) sind, d.h. die Anzahl der Nicht–Null Einträge ist $\mathcal{O}(n)$ im Unterschied zu $\mathcal{O}(n^2)$ einer vollbesetzen Matrix. Eine spezielle Form sind Bandmatrizen. Bandmatrizen sind quadratische Matrizen, bei der alle Matrixelemente gleich Null sein müssen, außer denen, die auf der Hauptdiagonalen und p darüber und q darunter liegenden Diagonalen a_{ij} mit $j \leq i + p$ bzw. a_{ij} mit $i \leq j + q$ liegen. Die Matrixelemente ungleich Null sind alle auf einem diagonal verlaufenden Band angeordnet, was auch den Namen erklärt. Eine Bandmatrix hat also die folgende Gestalt:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1p} & & & 0 \\ \vdots & \ddots & & \ddots & & \\ a_{q1} & & \ddots & & \ddots & \\ & \ddots & & \ddots & & a_{n-p+1,n} \\ & & \ddots & & & \vdots \\ 0 & & & a_{n,n-q+1} & \cdots & a_{nn} \end{pmatrix}. \tag{3.5.1}$$

Man sagt, die Matrix hat die *Bandbreite* $p + q - 1$.

Bemerkung 3.5.2. Bei der LR–Zerlegung ohne Pivotisierung bleibt die Bandbreite erhalten. Sei $A = LR$ mit Bandbreite $p + q - 1$. Dies ist äquivalent zu L hat die Bandbreite q und R hat die Bandbreite p. ∎

Für den Rechenaufwand gilt: Die LR—Zerlegung kostet $\mathcal{O}(pqn)$ und das Vorwärts– und Rückwärtseinsetzen $\mathcal{O}((p + q)n)$. Insbesondere gilt, falls p und q klein sind relativ zu n, daß der Aufwand zu $\mathcal{O}(n)$ proportional zur Anzahl der Unbekannten schrumpft. Weiter kann man eine kompakte Speicherung von Bandmatrizen durchführen. Bandmatrizen werden im Allgemeinen nicht als Feld mit n^2 Einträgen gespeichert, sondern man benutzt *Skyline– oder SKS–Formate*. Hierzu codiert man die Skyline der Matrix. Die Idee dabei ist, daß man Matrixeinträge hintereinander weg speichert und einen Vektor mit Positionen der Einträge initialisiert.

3.6 Fehleranalyse bei linearen Gleichungssystemen

Wir fragen uns nun, wie die exakte Lösung x von

$$Ax = b \tag{3.6.1}$$

mit nichtsymmetrischem $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$ von Störungen in A und b abhängt. Dies soll im Sinne der Rückwärtsanalyse geschehen. Wir interpretieren also Rundungsfehler bei Rechnungen als Eingabefehler. Sei dabei $\|\cdot\|$ eine beliebige, aber fest gewählte Vektornorm auf $\mathbb{R}^n$ bzw. die durch

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

induzierte Operatornorm. Als Operatornorm ist $\|\cdot\|$ submultiplikativ, d.h.

$$\|AB\| \leq \|A\| \cdot \|B\| \tag{3.6.2}$$

für $A, B \in \mathbb{R}^{n\times n}$.

Sei $\tilde{x} = x + \Delta x$ die Lösung des gestörten Systems

$$(A + \Delta A)(x + \Delta x) = (b + \Delta b). \tag{3.6.3}$$

Satz 3.6.4 (Störungssatz)**.** Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär und $b \in \mathbb{R}^n$. Es gelte

$$\|A^{-1}\| \cdot \|\Delta A\| < 1. \tag{3.6.5}$$

Dann gilt für Δx gemäß (3.6.3)

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa_{\|\cdot\|}(A)}{1 - \kappa_{\|\cdot\|}(A) \cdot \frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right). \tag{3.6.6}$$

mit der Kondition von A bezüglich $\|\cdot\|$,

$$\kappa_{\|\cdot\|}(A) = \|A\| \cdot \|A^{-1}\|. \tag{3.6.7}$$

■

Beachte noch, daß wegen

$$\kappa_{\|\cdot\|}(A) \cdot \frac{\|\Delta A\|}{\|A\|} = \|A\| \cdot \|A^{-1}\| \cdot \|\Delta A\| < 1$$

der Faktor auf der rechten Seite in (3.6.6) positiv ist. Weiter besagt (3.6.6), daß sich der relative Fehler in x durch den relativen Fehler der Eingabedaten $\frac{\|\Delta A\|}{\|A\|}$ und $\frac{\|\Delta b\|}{\|b\|}$ abschätzen läßt. Satz 3.6.4 gilt auch, wenn $\Delta A = 0$ oder $\Delta b = 0$ ist, also keine Störung weder in A noch in b vorhanden ist. Falls $\Delta A = 0$ ist, reduziert sich (3.6.6) auf

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa_{\|\cdot\|}(A) \frac{\|\Delta b\|}{\|b\|}. \tag{3.6.8}$$

Beweis: Es gilt

$$\begin{aligned}
& (A + \Delta A)(x + \Delta x) = b + \Delta b \\
\iff \quad & Ax + (\Delta A)x + A(\Delta x) + (\Delta A)(\Delta x) = b + (\Delta b) \\
\iff \quad & A(\Delta x) = \Delta b - (\Delta A)x - (\Delta A)(\Delta x) \\
\iff \quad & \Delta x = A^{-1}(\Delta b) - A^{-1}(\Delta A)x - A^{-1}(\Delta A)(\Delta x),
\end{aligned}$$

womit

$$\|\Delta x\| \leq \|A^{-1}\| \cdot \|\Delta A\| \cdot \|x\| + \|A^{-1}\| \cdot \|\Delta A\| \cdot \|\Delta x\| + \|A^{-1}\| \cdot \|\Delta b\|$$

folgt. Dies ist äquivalent zu

$$\left(1 - \kappa_{\|\cdot\|}(A) \frac{\|\Delta A\|}{\|A\|}\right) \cdot \|\Delta x\| \leq \|A^{-1}\| \cdot \frac{\|A\|}{\|A\|} \cdot \|\Delta A\| \|x\| + \frac{\|A^{-1}\|}{\|A\|} \cdot \|\Delta b\| \|A\|.$$

Beachte nun

$$\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|,$$

woraus

$$\frac{1}{\|A\|} \leq \frac{\|x\|}{\|b\|}$$

folgt, womit wir weiter abschätzen

$$\left(1 - \kappa_{\|\cdot\|}(A) \frac{\|\Delta A\|}{\|A\|}\right) \|\Delta x\|$$

$$\begin{aligned} &\leq \kappa_{\|\cdot\|}(A)\left(\frac{\|\Delta A\|}{\|A\|})\|x\| + \frac{\|\Delta b\|}{\|b\|}\right) \\ &\leq \kappa_{\|\cdot\|}(A)\left[\frac{\|\Delta A\|}{\|A\|}\|x\| + \frac{\|\Delta b\|}{\|b\|}\|x\|\right]. \end{aligned}$$

Und somit gilt

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa_{\|\cdot\|}(A)}{1 - \kappa_{\|\cdot\|}(A) \cdot \frac{\|\Delta A\|}{\|A\|}} \cdot \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}\right).$$

∎

Bemerkung 3.6.9. In einer Maschine mit Maschinengenauigkeit eps sind die Daten A, b mit dem Fehler eps behaftet. Daher gilt nach Satz 3.6.4, daß es für die Bestimmung von x einen unvermeidlichen Fehler der Größenordnung $\kappa_{\|\cdot\|}(A)\,\text{eps}$ gibt.

Bemerkung 3.6.10. Allgemein läßt sich natürlich $\|A^{-1}\|$ nicht explizit ausrechnen, da die Berechnung von A^{-1} mehr Aufwand als die Lösung von $Ax = b$ erfordert. Daher schätzt man $\kappa_{\|\cdot\|}(A)$ über Kenntnisse der Eigenwerte ab. Es gilt $\frac{|\lambda_{\max}(A)|}{|\lambda_{\min}(A)|} \geq \kappa_{\|\cdot\|}(A)$. ∎

Wir fassen die Ergebnisse dieses Abschnitts noch einmal zusammen: Ist A symmetrisch positiv definit, so kann man das Cholesky–Verfahren anwenden. Die Rückwärtsanalyse liefert, daß $\tilde{x}$ die exakte Lösung des gestörten Systems

$$(A + \Delta A)\tilde{x} = b$$

ist, wobei sich die Störung durch

$$\frac{\|\Delta A|\|_\infty}{\|A\|_\infty} \leq c_n \cdot \text{eps}$$

mit einer von n abhängigen Konstante c_n abschätzen läßt. Somit ist die Störung in der Größenordnung der Maschinengenauigkeit. Also ist das Cholesky–Verfahren stabil. ähnliches gilt für die LR–Zerlegung mit Pivotisierung. Hier ist das Verfahren auch stabil, d.h. der Fehler bleibt in der Größenordnung des durch die Kondition bedingten Fehlers. Das Verfahren für die LR–Zerlegung für beliebiges A ohne Pivotisierung hingegen ist nicht stabil.

Beispiel 3.6.11. Löse das System $Ax = b$, wobei A die Hilbert–Matrix

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+1} \\ \vdots & & & & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & & \cdots & & \frac{1}{2n-1} \end{pmatrix} \in \mathbb{R}^{n\times n}$$

ist. Wähle $b = (\frac{1}{n}, \frac{1}{n+1}, \ldots, \frac{1}{2n-1})^T$. Dann ist die Lösung $x = (0, \ldots, 0, 1)^T$. A ist symmetrisch positiv definit. Für $n = 12$ und $\text{eps} = 10^{-16}$ gilt für die Cholesky–Zerlegung, daß das Resultat mit einem Fehler der Größenordnung

$$\frac{\|\tilde{x} - x\|_\infty}{\|x\|_\infty} \approx 1.6 \cdot 10^{-2}$$

behaftet ist, da $\kappa_\infty(A) \approx 10^{16}$ für $n = 12$ ist. ∎

3.7 QR–Zerlegung

Bisher hatten wir als Verfahren zur Lösung des Gleichungssystems $Ax = b$ die Zerlegung $A = LR$ betrachtet, wobei L eine untere und R eine obere Dreiecksmatrix waren. Jetzt diskutieren wir eine Zerlegung

$$A = QR \tag{3.7.1}$$

wobei Q eine orthogonale und R eine obere Dreiecksmatrix ist. Dazu definieren wir

Definition 3.7.2. Eine Matrix $Q \in \mathbb{R}^{n\times n}$ heißt orthogonal, falls $Q^TQ = I$ ist. ∎

Aufgrund der Orthogonalität ist $Q^{-1} = Q^T$, was

$$Ax = b \iff QRx = b \iff Rx = Q^Tb \tag{3.7.3}$$

impliziert, d.h. die Berechnung der Lösung reduziert sich wieder auf Rückwärtseinsetzen, wenn die QR–Zerlegung von A bekannt ist (für nichtsinguläres A). Die QR–Zerlegung ist ein sehr stabiles Verfahren zur Lösung linearer Gleichungsysteme. Sie wird auch zur Lösung von Ausgleichsproblemen und der Berechnung von Eigenwerten eingesetzt. Insbesondere muß die Matrix weder quadratisch noch nichtsingulär sein. Die Menge der orthogonalen Matrizen bezeichnen wir mit

$$\mathbb{O}_n(\mathbb{R}) := \{Q \in \mathbb{R}^{n\times n} : \ Q^T = Q^{-1}\}. \tag{3.7.3a}$$

Satz 3.7.4. Für orthogonale Matrizen $Q, P \in \mathbb{O}_n(\mathbb{R})$ gilt

i) $|\det(Q)| = 1$.

ii) $Q \cdot P \in \mathbb{O}_n(\mathbb{R})$.

iii) $\|Qx\|_2 = \|x\|_2$ für alle $x \in \mathbb{R}^n$.
Geometrisch beschreiben orthogonale Matrizen also Drehungen oder Spiegelungen.

iv) $\|A\|_2 = \|QA\|_2 = \|AQ\|_2$ für alle $A \in \mathbb{R}^{n\times n}$

v) $\kappa_2(QA) = \kappa_2(AQ) = \kappa_2(A)$ für alle $A \in \mathbb{R}^{n\times n}$

vi) $\kappa_2(Q) = 1$.

∎

Wir kommen nun zur Bestimmung der Zerlegung (3.7.1). Die Matrix A wird durch sukzessive Multiplikation mit geeigneten Orthogonalmatrizen $Q_i \in \mathbb{O}_n$ von links auf obere Dreiecksgestalt gebracht. Man setzt

$$\prod Q_i =: Q.$$

Die Konstruktion der Q_i erfolgt nach zwei unterschiedlichen Prinzipien. Entweder

- nach den Householder–Reflexionen, oder
- nach den Givens–Rotationen, die gezielt ein vorhandenes Nullenmuster der Matrix ausnutzen können.

3.7.1 HOUSEHOLDER–Spiegelungen

Das Prinzip der HOUSEHOLDER–Spiegelungen ist, die Spalten von A sukzessive auf ein Vielfaches des ersten Einheitsvektors zu transformieren. Definiere dazu für $v = (v_1, \ldots, v_n)^T$ die *Dyade*

$$vv^T = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} (v_1, \ldots, v_n) = \begin{pmatrix} v_1v_1 & \cdots & v_1v_n \\ \vdots & & \vdots \\ v_nv_1 & \cdots & v_nv_n \end{pmatrix}.$$

Man beachte, daß $vv^T \in \mathbb{R}^{n\times n}$ und $v^Tv \in \mathbb{R}$ sind. Eine HOUSEHOLDER–Transformation ist von der Form

$$Q_v := I - \frac{2}{v^Tv} vv^T. \tag{3.7.5}$$

Eigenschaften 3.7.6. Für die Matrix Q_v gilt:

i) $Q_v = Q_v^T$.

ii) $Q_v^2 = I$, da

$$Q^2v = Q(Qv) = \left(I - \frac{2}{v^Tv}vv^T\right)\left(I - \frac{2}{v^Tv}vv^T\right) = I - \frac{4}{v^Tv}vv^T + \frac{4}{(v^Tv)^2}v(v^Tv)v^T = I.$$

iii) $Q_v \in \mathbb{O}_n(\mathbb{R})$, da $Q_v^TQ_v \overset{\text{i)}}{=} Q_v^2 \overset{\text{ii)}}{=} I$.

iv) $Q_{\alpha v} = Q_v$ für alle $\alpha \in \mathbb{R}_{\neq 0}$.

v) $Q_vy = y \iff y^Tv = 0$.

vi) $Q_vv = -v$.

∎

Im Folgenden ist es unsere

Grundaufgabe 3.7.7. Zu gegebenem $y \in \mathbb{R}^n$ finde ein $v \in \mathbb{R}^n$, sodaß

$$Q_vy = \pm\|y\|_2 e^1 \tag{3.7.8}$$

gilt. Der Faktor $\|y\|_2$ auf der rechten Seite stammt von $\|Q_vy\|_2 = \|y\|_2$, da $Q_v \in \mathbb{O}_n$ ist; weiter rührt $\pm$ daher, daß man zwei Möglichkeiten zum Spiegeln hat. Aus der Forderung

$$Q_vy = \left(I - \frac{2}{v^Tv}vv^T\right)y = y - \frac{2}{v^Tv}\left(v^Ty\right)v = ce^1$$

folgt, daß v eine Linearkombination von y und e^1 ist. Weiterhin kann wegen 3.7.6 (iv) die Skalierung von v frei gewählt werden. Setze daher

$$v := y + \alpha e^1. \tag{3.7.9}$$

Die Bestimmung von α steht noch aus:

$$\begin{aligned}
Q_v y &= \left(I - \frac{2}{v^T v} v v^T\right) y \\
&= y - \frac{2}{(y+\alpha e^1)^T (y+\alpha e^1)} \left(y+\alpha e^1\right)\left(y+\alpha e^1\right)^T y \\
&= y - \frac{2}{(y+\alpha e^1)^T (y+\alpha e^1)} \left(yy^T + y\alpha\left(e^1\right)^T + \alpha e^1 y^T + \alpha e^1 \alpha\left(e^1\right)^T\right) y \\
&= y - \frac{2}{(y+\alpha e^1)^T (y+\alpha e^1)} \left(yy^T y + y\alpha y_1 + \alpha e^1 y^T y + \alpha e^1 \alpha y_1\right) \\
&= y - \frac{2}{(y+\alpha e^1)^T (y+\alpha e^1)} \left(y^T y y + \alpha y_1 y + \alpha e^1 \left(y^T y + \alpha y_1\right)\right) \\
&= \left(1 - \frac{2}{(y+\alpha e^1)^T (y+\alpha e^1)} (y^T y + \alpha y_1)\right) y - \frac{2\alpha e^1 (y^T y + \alpha y_1)}{(y+\alpha e^1)^T (y+\alpha e^1)}
\end{aligned}$$

Da $Q_v y$ ein Vielfaches von e^1 sein soll, muß der Koeffizient von y verschwinden. Also soll gelten

$$\begin{aligned}
& 1 && = && \frac{2}{(y+\alpha e^1)^T (y+\alpha e^1)} (y^T y + \alpha y_1) \\
\Leftrightarrow\ & \left(y+\alpha e^1\right)^T (y+\alpha e^1) && = && 2y^T y + 2\alpha y_1 \\
\Leftrightarrow\ & y^T y + y^T \alpha e^1 + \alpha (e^1)^T y + \alpha (e^1)^T \alpha e^1 && = && 2y^T y + 2\alpha y_1 \\
\Leftrightarrow\ & -y^T y + 2\alpha y_1 + \alpha^2 && = && 2\alpha y_1 \\
\Leftrightarrow\ & \alpha^2 && = && y^T y = \|y\|_2^2 \\
\Leftrightarrow\ & \alpha && = && \pm \|y\|_2 .
\end{aligned}$$

Eine sinnvolle Wahl für α ist daher

$$\alpha := \begin{cases} \operatorname{sign}(y_1)\|y\|_2 & \text{falls } y_1 \neq 0 \\ \|y\|_2 & \text{falls } y_1 = 0 \end{cases} \tag{3.7.10}$$

Der Grund für die Verwendung von sign ist die Vermeidung von Auslöschung

$$v = y + \alpha e^1 = y + \operatorname{sign}(y_1)\|y\|_2 e^1 = \begin{pmatrix} y_1 + \operatorname{sign}(y_1)\|y\|_2 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

■

Damit haben wir

Algorithmus 3.7.11 (zur Lösung der Grundaufgabe (3.7.7))**.** Gegeben sei $y \in \mathbb{R}^n$, setze

$$\alpha \leftarrow \operatorname{sign}(y_1)\|y\|_2 \qquad \text{und} \qquad v \leftarrow y + \alpha e^1.$$

■

Das Ergebnis dieses Algorithmus ist $Q_v y = -\alpha e^1$. Man beachte, daß Q_v *nicht* explizit aufgestellt werden muß, wodurch an Speicherplatz gespart wird. Die alleinige Anwendung von Q_v auf y ist ausreichend. Dazu folgendes

Beispiel 3.7.12. Finde zu

$$y = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

ein $v \in \mathbb{R}^3$ mit

$$Q_v y = \pm \|y\|_2 e^1 = \pm 3 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Nach Algorithmus 3.7.11 ist $\alpha = +3$ sowie

$$v = \begin{pmatrix} 2+3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}$$

und es gilt

$$Q_v y = \begin{pmatrix} -3 \\ 0 \\ 0 \end{pmatrix}.$$

Um das Beispiel testen zu können, geben wir die Lösung Q_v zu diesem an,

$$Q_v = \frac{1}{15} \begin{pmatrix} -10 & -10 & -5 \\ -10 & 11 & -2 \\ -5 & -2 & 14 \end{pmatrix}.$$

■

Nun kommen wir zur Reduktion von A auf eine obere Dreiecksmatrix mittels HOUSEHOLDER–Matrizen. Die Idee hierbei ist die sukzessive Anwendung der Grundaufgabe auf die (ggf. verkürzten) Spalten von A. Sei $A \in \mathbb{R}^{m \times n}$ und $m \geq n$. Wende Algorithmus 3.7.11 auf die erste Spalte a^1 von A an, d.h.

$$\begin{array}{rcl} v^1 & := & a^1 + \operatorname{sign}(a_{11}) \|a^1\|_2 e^1 \\ Q_1 & := & Q_{v^1} \in \mathbb{O}_m(\mathbb{R}). \end{array} \tag{3.7.13}$$

Die Matrix $Q_1 A$ hat die Form

$$Q_1 A = \begin{pmatrix} * & * & * & * \\ 0 & & & \\ \vdots & & \tilde{A}^2 & \\ 0 & & & \end{pmatrix} =: A^{(2)},$$

mit $\tilde{A}^2 \in \mathbb{R}^{(n-1)\times(n-1)}$. Man beachte, daß $Q_1 a^1$ schon bekannt ist, aber man noch

$$Q_1 a^j = a_j - \frac{2v^1((v^1)^T a^j)}{(v^1)^T v^1}$$

berechnen muß. Daraus folgt dann explizit die Matrix $A^{(2)}$. Bestimme analog ein $\tilde{Q}_2 = Q_{v^2} \in \mathbb{R}^{(m-1)\times(m-1)}$ für die erste Spalte von $\tilde{A}^{(2)}$ und setze

$$Q_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \vdots & & & \\ \vdots & & \tilde{Q}_2 & \\ 0 & & & \end{pmatrix} \in \mathbb{R}^{m \times n}.$$

Damit erhält man

$$Q_2Q_1A = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & 0 & & \\ \vdots & \vdots & \tilde{A}^{(3)} & \\ 0 & 0 & & \end{pmatrix}$$

usw. Auf freiwerdenden Stellen unterhalb der $\tilde{a}_{ii}$ speichert man $v^i \in \mathbb{R}^{m-i+1}$. Da allerdings nur $p-i$ Plätze frei werden, speichert man die Diagonalelemente gesondert in einem Vektor $d \in \mathbb{R}^n$ ab.

Beispiel 3.7.14. Seien

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 0 \\ 2 & 0 \end{pmatrix}, \quad \text{und damit} \quad v^1 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} + 3e^1 = \begin{pmatrix} 4 \\ 2 \\ 2 \end{pmatrix}.$$

Mit $Q_1 = Q_{v_1}$ ist

$$Q_1A = \begin{pmatrix} -3 & -\frac{1}{3} \\ 0 & -\frac{2}{3} \\ 0 & -\frac{2}{3} \end{pmatrix}$$

Für v_2 erhalten wir

$$v_2 = \begin{pmatrix} -\frac{2}{3}(1+\sqrt{2}) \\ -\frac{2}{3} \end{pmatrix}.$$

Damit ergibt sich mit $\tilde{Q}_2 = \tilde{Q}v_2$, daß

$$\tilde{Q}_2 \begin{pmatrix} -\frac{2}{3} \\ -\frac{2}{3} \end{pmatrix} = \begin{pmatrix} \frac{2\sqrt{2}}{3} \\ 0 \end{pmatrix},$$

womit folgt

$$Q_2Q_1A = \begin{pmatrix} -3 & -\frac{1}{3} \\ 0 & \frac{2\sqrt{2}}{3} \\ 0 & 0 \end{pmatrix}.$$

Das Resultat wird als

$$\begin{pmatrix} 4 & -\frac{1}{3} \\ 2 & -\frac{2}{3}(1+\sqrt{2}) \\ 2 & -\frac{2}{3} \end{pmatrix}$$

und $d = (-3, \frac{2\sqrt{2}}{3})^T$ abgespeichert. ∎

Damit gelten folgende Eigenschaften der HOUSEHOLDER–Transformationen:

- Das Verfahren ist sehr stabil, d.h. eine Pivotisierung ist nicht erforderlich. Für Einzelheiten schaue man in [GL].
- Der Aufwand für eine vollbesetze Matrix $A \in \mathbb{R}^{m\times n}$ ist für $m \approx n$ von der Ordnung $\mathcal{O}\left(\frac{2}{3}n^3\right)$ und für $m \gg n$ bei $\mathcal{O}\left(mn^2\right)$.

3.7.2 GIVENS–Rotationen

Das Prinzip der GIVENS–Rotationen ist, die Spalten von A durch ebene Drehungen sukzessive in Achsenrichtung zu bringen. Dazu stellen wir uns wieder eine

Grundaufgabe 3.7.15. Gegeben sei $(a,b)^T \in \mathbb{R}^2$. Finde $c,s \in \mathbb{R}$ mit der Eigenschaft

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix} \tag{3.7.16}$$

und

$$c^2 + s^2 = 1. \tag{3.7.17}$$

■

Bemerkung 3.7.18. Wegen (3.7.17) kann man $c = \cos\varphi$ und $s = \sin\varphi$ für ein $\varphi \in [0, 2\pi)$ setzen, d.h. (3.7.16) stellt eine Drehung im $\mathbb{R}^2$ um den Winkel φ dar. Für die Rechnung wird φ jedoch nicht explizit benötigt. ■

Man beachte, daß die Matrix

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

nur bis auf die Vorzeichen bestimmt ist. Die Wahl des Vorzeichens hat aber keine Auswirkungen. Weiterhin läßt die Drehung die EUKLIDISCHE Länge unverändert,

$$\left\| \begin{pmatrix} r \\ 0 \end{pmatrix} \right\|_2 = |r| = \sqrt{a^2+b^2} = \left\| \begin{pmatrix} a \\ b \end{pmatrix} \right\|_2. \tag{3.7.19}$$

Zur Bestimmung von c und s gilt nach (3.3.16)

$$-sa + cb = 0 \iff cb = sa \iff c = \frac{sa}{b}.$$

Dies in die erste Gleichung eingesetzt, ergibt

$$\frac{sa}{b}a + sb = r \iff s\left(\frac{a^2+b^2}{b}\right) \iff s = \frac{rb}{a^2+b^2} = \frac{b}{r}.$$

Damit ergibt sich weiter

$$c = \frac{sa}{b} = \frac{b}{r}\frac{a}{b} = \frac{a}{r}.$$

Also erhalten wir für s und c

$$c := \frac{a}{\sqrt{a^2+b^2}} \quad \text{und} \quad s = \frac{b}{\sqrt{a^2+b^2}}. \tag{3.7.20}$$

Diese Lösung ist eine Möglichkeit, die Grundaufgabe zu lösen. Um Rundungsfehler gering zu halten, implementiert man anstelle (3.7.20) das folgende Schema.

Algorithmus 3.7.21.

`IF` $b = a$,

$c \leftarrow 1$

$s \leftarrow 0$

`ELSE IF` $|b| > |a|$,

$\tau \leftarrow \frac{a}{b}$
$s \leftarrow \frac{1}{\sqrt{1+\tau^2}}$
$c \leftarrow s \cdot \tau$

`ELSE`

$\tau \leftarrow \frac{b}{a}$
$c \leftarrow \frac{1}{\sqrt{1+\tau^2}}$
$s \leftarrow c \cdot \tau$ ∎

Dieses Vorgehen ist zur Vermeidung von Multiplikationen großer Zahlen. Jetzt erweitern wir die obige Strategie auf $n \times n$ Matrizen durch Einbettung.

Definition 3.7.22. Wir definieren die Matrix $G_{i,k} \in \mathbb{O}_n$ durch

$$G_{i,k} = \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & c & \cdots & \cdots & \cdots & s & & \\ & & & 1 & & & & & \\ & & \vdots & & \ddots & & \vdots & & \\ & & & & & 1 & & & \\ & & -s & \cdots & \cdots & \cdots & c & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{pmatrix},$$

wobei c, s wieder (3.7.7) erfüllen. $G_{i,k}$ bewirkt eine Rotation in der durch e^i und e^k aufgespannten Ebene. ∎

Insbesondere gilt

$$G_{i,k} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_{i-1} \\ r \\ x_{i+1} \\ \vdots \\ x_{k-1} \\ 0 \\ x_{k+1} \\ x_n \end{pmatrix} \tag{3.7.23}$$

für $r = \pm\sqrt{x_i^2 + x_k^2}$ und

$$c = \frac{x_i}{r} \quad \text{und} \quad s = \frac{x_k}{r}. \tag{3.7.24}$$

Wir fahren fort mit einem

Beispiel 3.7.25. Unter Anwendung des obigen Prinzips gilt

$$x = \begin{pmatrix} 4 \\ -3 \\ 1 \end{pmatrix} \overset{G_{1,2}\cdot}{\to} \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} \overset{G_{1,3}\cdot}{\to} \begin{pmatrix} \sqrt{26} \\ 0 \\ 0 \end{pmatrix}$$

mit

$$G_{1,2} = \begin{pmatrix} \frac{4}{5} & -\frac{3}{5} & 0 \\ \frac{3}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{und} \quad G_{1,3} = \begin{pmatrix} \frac{5}{\sqrt{26}} & 0 & \frac{1}{\sqrt{26}} \\ 0 & 1 & 0 \\ -\frac{1}{\sqrt{26}} & 0 & \frac{5}{\sqrt{26}} \end{pmatrix}.$$

■

Nun die Anwendung auf Matrizen: Zur Reduktion von A auf obere Dreiecksgestalt geht man folgendermaßen vor. Man wende nacheinander GIVENS–Rotationen an, um die Einträge unterhalb der Diagonalen zu Null zu machen. Bei diesem Verfahren nutzt man die vorhandenen Nullen aus. Man erhält

$$G_{i_N,k_N} \cdots G_{i_1,k_1} A = R. \tag{3.7.26}$$

Daraus folgt mit (3.7.2) und (3.7.4) (ii)

$$A = G_{i_1,k_1}^T \cdots G_{i_n,k_N}^T \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} =: QR, \tag{3.7.27}$$

wobei $Q \in \mathbb{O}_m(\mathbb{R})$ und $R \in \mathbb{R}^{m\times n}$ sind. Wir bemerken noch für den Fall $p := \operatorname{rang}(A) < n$, daß $\tilde{R}$ dann von der Form

$$\tilde{R} = \begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ & \ddots & & & \vdots \\ & & * & \cdots & * \\ & & & & \\ & 0 & & & \end{pmatrix} \tag{3.7.28}$$

ist. Falls $m < n$, so hat R die Form

$$R = \begin{pmatrix} 0 & * & \cdots & * \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & * \end{pmatrix} \in \mathbb{R}^{m\times n}. \tag{3.7.29}$$

Beachte noch, daß die Anwendung von $G_{i,k}$ die i–te und k–te Zeile verändert. Hierbei muß die Reihenfolge beachtet werden.

Beispiel 3.7.30. Schematisch schauen wir uns zunächst einmal die Wirkung des Verfahrens an:

$$\begin{pmatrix} * & * \\ * & * \\ * & * \end{pmatrix} \overset{G_{1,2}\cdot}{\to} \begin{pmatrix} * & * \\ 0 & * \\ * & * \end{pmatrix} \overset{G_{1,3}\cdot}{\to} \begin{pmatrix} * & * \\ 0 & * \\ 0 & * \end{pmatrix} \overset{G_{2,3}\cdot}{\to} \begin{pmatrix} * & * \\ 0 & * \\ 0 & 0 \end{pmatrix}.$$

Ein konkretes Zahlenbeispiel ist

$$A = \begin{pmatrix} 3 & 5 \\ 0 & 2 \\ 0 & 0 \\ 4 & 5 \end{pmatrix} \overset{G_{1,4}\cdot}{\to} \begin{pmatrix} 5 & 7 \\ 0 & 2 \\ 0 & 0 \\ 0 & -1 \end{pmatrix} \overset{G_{2,4}\cdot}{\to} \begin{pmatrix} 5 & 7 \\ 0 & \sqrt{5} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

■

Hinweise zur Implementierung 3.7.31. A wird mit $G_{i,k}A$ wie folgt überschrieben: Berechne c, s gemäß Algorithmus 3.7.21.

Für $j = 1, \ldots, n$

$a \leftarrow a_{ij}$

$b \leftarrow a_{kj}$

$a_{ij} \leftarrow ca + sb$

$a_{kj} \leftarrow -sa + cb$ ∎

Die Codierung der $G_{i,k}$ ist mittels einer Zahl

$$\varphi := \varphi_{ik} := \begin{cases} 1 & \text{falls } c = 0 \\ \text{sign}(c) \cdot \frac{s}{2} & \text{falls } |c| > |s| \\ \text{sign}(s) \cdot \frac{2}{c} & \text{falls } |s| > |c| \end{cases} \tag{3.7.32}$$

möglich, die auf den freiwerdenden Stellen $a_{i,k}$ abgespeichert werden kann. Zur Dekodierung verwende dann

Algorithmus 3.7.33.

`IF` $\varphi = 1$

$c \leftarrow 0$

$s \leftarrow 1$

`ELSE IF` $|\varphi| < 1$

$s \leftarrow 2 \cdot \varphi$

$c \leftarrow \sqrt{1 - s^2}$

`ELSE`

$c \leftarrow \frac{2}{\varphi}$

$s \leftarrow \sqrt{1 - c^2}$ ∎

Beachte, daß die Transformation $G_{i,k}$ ohne explizite Aufstellung der Matrix erfolgt.

Eigenschaften der Givens–Rotationen sind

- die QR–Zerlegung mit Givens–Rotationen ist sehr stabil, d.h. eine Pivotisierung ist nicht erforderlich.
- Eine flexible Anpassung bei strukturierten Matrizen, etwa bei vorhandenen Nulleinträgen ist sehr gut möglich. Ein Beispiel hierfür sind Hessenberg–Matrizen, bei denen man nur $n - 1$ Givens–Rotationen benötigt um eine obere Dreiecksgestalt zu erreichen.
- Der Aufwand für die QR–Zerlegung mit Givens–Rotationen bei vollbesetzter Matrix $A \in \mathbb{R}^{n \times n}$ ist falls $m \approx n$ von der Ordnung $\mathcal{O}\left(\frac{4}{3}n^3\right)$, und falls $m \gg n$ von der Ordnung $\mathcal{O}\left(2mn^2\right)$. Der Aufwand ist bei dünn besetzen Matrizen ist wesentlich niedriger, jedoch höher als bei der LR–Zerlegung; aber dafür wesentlich stabiler. Bei sog. *schnellen* Givens–*Rotationen* ist der Aufwand falls $m \approx n$ von der Ordnung $\mathcal{O}\left(\frac{2}{3}n^3\right)$ und falls $m \gg n$ von der Ordnung $\mathcal{O}(m \cdot n^2)$.

HOUSEHOLDER–Spiegelungen und GIVENS–Rotationen liefern den konstruktiven Beweis für den folgenden

Satz 3.7.34. Sei $A \in \mathbb{R}^{m\times n}$. Dann existiert stets ein $Q \in \mathbb{O}_m(\mathbb{R})$ und eine obere Dreiecksmatrix R mit

$$A = QR,$$

wobei R die Form (3.7.28) oder (3.7.29) hat. ∎

Wir fassen den Inhalt der Kapitel 3.7.1 und 3.7.2 noch einmal kurz zusammen. Die Lösung von $Ax = b$ mit $A \in \mathbb{R}^{n\times n}$ über die QR–Zerlegung mittels $A = QR$ und Rückwärtseinsetzen ist sehr stabil, da $\kappa_2(A) = \kappa_2(R)$ und $\kappa_2(Q) = 1$ ist. Die Bestimmung von Q und R erfolgt über GIVENS– oder HOUSEHOLDER–Transformationen. Beide Methoden sind sehr stabil. Der Aufwand der GIVENS–Methode ist etwa doppelt so hoch wie der der HOUSEHOLDER–Methode. Allerdings kann bei der GIVENS–Transformation gezielt die Nullstruktur der Matrix A ausgenutzt werden.

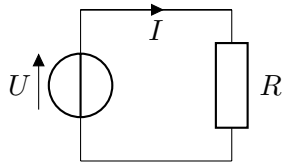
4 Lineare Ausgleichsprobleme

4.1 Einleitung

Zur Motivation der in diesem Kapitel beleuchteten Problematik seien zwei Beispiele vorangestellt.

Beispiel 4.1.1. Betrachte einen Gleichstromkreis mit Stromstärke I, Spannung U und Widerstand R. Nach dem OHMschen Gesetz gilt (zumindest für gewisse Temperaturbereiche)

$$U = RI. \tag{4.1.2}$$



Nun sei uns eine Messreihe von Daten (U_i, I_i) für $i = 1, \ldots, m$ gegeben. Die Aufgabe lautet: bestimme aus den Messdaten den Widerstand R im Stromkreis. Theoretisch müßte R

$$U_i = RI_i \tag{4.1.3}$$

für alle $i = 1, \ldots, m$ erfüllen. Da die Messdaten aber mit Fehlern behaftet sind, existieren $i \neq j$ mit

$$R = \frac{U_i}{I_i} \neq \frac{U_j}{I_j}.$$

Nun stellt sich die Frage, welcher für R errechnete Wert der beste ist, oder welches Paar/welche Paare von Messdaten am geeignetesten sind. Um dies zu beantworten, versuchen wir den Meßfehler *auszugleichen*, z.B. indem wir das R bestimmen, das den Ausdruck

$$f(R) := \sum_{i=1}^{m} (RI_i - U_i)^2 \tag{4.1.4}$$

minimiert. Das Minimum von f ist gegeben durch

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

Das bedeutet

$$R^* = \frac{\sum\limits_{i=1}^{m} U_i I_i}{\sum\limits_{i=1}^{m} I_i^2} \tag{4.1.5}$$

ist extremal. Da $f''(R) = 2\sum\limits_{i=1}^{m} I_i^2 > 0$ für alle R ist, ist R^* auch minimal und damit ein in diesem Sinne geeigneter Wert für den Widerstand im Stromkreis. ∎

Beispiel 4.1.6 (FOURIER–Analyse)**.** Gegeben sei ein Vorgang, der sich durch eine stetige Funktion $f(t)$ mit $t \geq 0$ beschreiben lasse und periodisch mit der Periode T sei. Diesen modellieren wir mittels der FOURIER–Polynome für $N \in \mathbb{N}$

$$g_N(t) = \frac{1}{2}a_0 + \sum_{k=1}^{N} \left[a_k \cos\left(\frac{2\pi k}{T}t\right) + b_k \sin\left(\frac{2\pi k}{T}t\right) \right]. \tag{4.1.7}$$

Dabei soll

$$\|g_N - f\|_{\mathrm{L}_2}^2 := \int_0^T (g_N(t) - f(t))^2 dt = \min \tag{4.1.8}$$

sein. Daraus ergeben sich als FOURIER–Koeffizienten für (4.1.7)

$$a_k = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi k}{T}\right) dt \tag{4.1.9a}$$

und

$$b_k = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi k}{T}\right) dt. \tag{4.1.9b}$$

Wir nehmen nun an, daß anstelle von f nur eine Reihe von Messdaten

$$f_i \approx f(t_i)$$

mit $0 \le t_1 < \ldots < t_m \le T$ vorliege, wobei $m > 2N + 1$ sei. Statt (4.1.8) fordern wir nun

$$\sum_{i=1}^{m} (g_N(t_i) - f_i)^2 = \min \tag{4.1.10}$$

zur Bestimmung von g_N, d.h. zur Bestimmung von a_k, b_k. ∎

Beispiel 4.1.11 (Zeitreihenanalyse des DAX (aus [Sch]))**.** Gegeben sei ein Datensatz, der sich aus einer endlichen Menge von Zahlentupeln (t_k, f_k) zusammensetzt. Falls

$$0 \le t_0 < \ldots < t_N - 1 =: T, \quad f_k \in \mathbb{R} \quad \text{für jedes} \quad k \in \{0, \ldots, N-1\} \quad \text{gilt,}$$

nennt man diesen eine *Zeitreihe*. Die Entwicklung effizienter Verfahren zur Trendbestimmung großer Datensätze, wie beispielsweise dem DAX (Deutscher Aktienindex), und zur Signalverarbeitung ist in den letzten Jahrzehnten immer relevanter geworden.

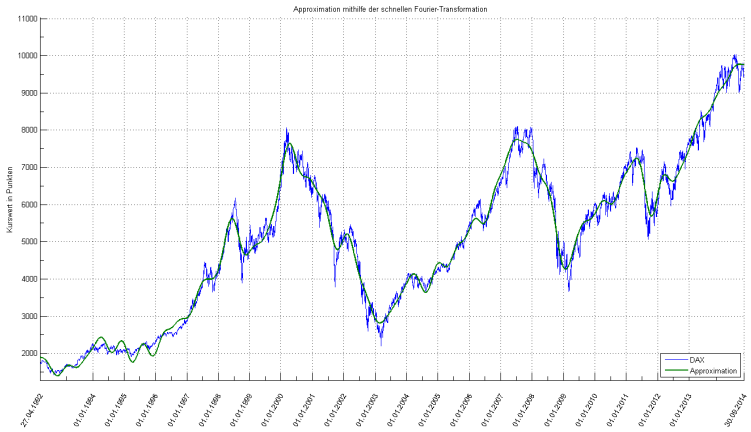


Abbildung 1: *Approximation (grün) des deutschen Aktienindex (blau) mithilfe der schnellen* FOURIER*-Transformation und den 64 betragsmäßig größten Koeffizienten*

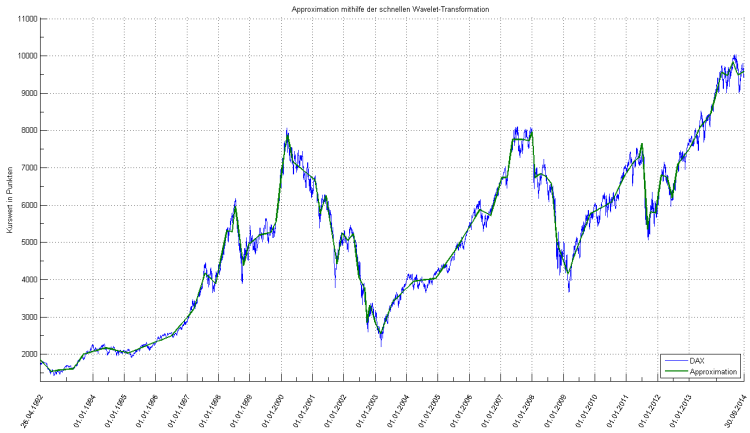


Abbildung 2: *Approximation (grün) des deutschen Aktienindex (blau) mithilfe der schnellen Wavelet-Transformation und den 64 betragsmäßig größten Koeffizienten*

Zwei bekannte Verfahren, die schnelle FOURIER-Transformation (FFT), die schnelle Wavelet-Transformation (FWT) und die neuere Empirical Mode Decomposition (EMD), werden in den Abbildungen 1, 2 und 3 am Beispiel des DAX miteinander verglichen.

Im Vergleich zu den anderen beiden Verfahren, kommt die EMD ohne Benutzereingabe aus, d.h., der Datensatz muss nicht nachträglich von Hand gewichtet werden, um eine gute Approximation zu erhalten.

Abbildung 3: *Approximation (grün) des deutschen Aktienindex (blau) bestimmt durch die Empirical Mode Decomposition mit dem Residuum und acht Intrinsic Mode Functions*

Andererseits muss neben dieser Eigenschaft vom Anwender die Effizienz und Effektivität mehrerer Kompressionsverfahren abgewogen werden.
Mit wachsenden Datenmengen ist man bei einer groben Trendbestimmung mit dem schnellsten Algorithmus viel besser bedient als mit einem langsameren, der zwar hochwertigere Daten liefert, dies aber möglicherweise zu spät vermag und damit den Sinn einer Prognose zunichtemacht.

Für weiterführende Informationen sei die Bachelorarbeit [Sch] zu diesem Thema empfohlen, der auch die obigen Abbildungen entnommen sind.

4.2 Lineare Ausgleichsprobleme — GAUSSsche Fehlerquadratmethode

Im allgemeinen liegt folgende Situation vor:

Aufgabe 4.2.1. Gegeben seien m Messwerte

$$(t_i, b_i)\,, \tag{4.2.2}$$

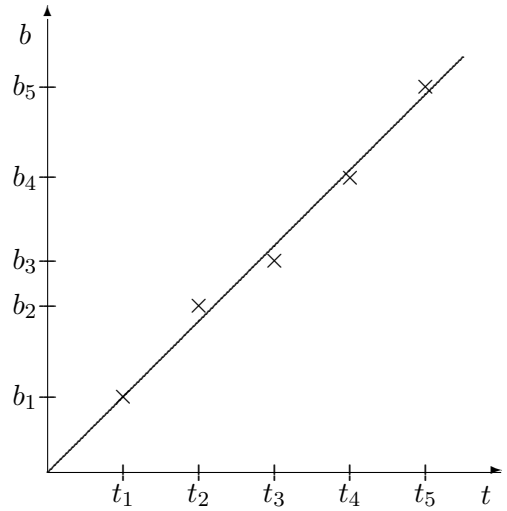
wobei $t_i, b_i \in \mathbb{R}$ und $i = 1, \ldots, m$ seien, die die Zustände eines Objektes $b(t)$ zur Zeit t_i beschreiben mögen.
Also gelte

$$b_i \approx b(t_i) \tag{4.2.3}$$

an allen betrachteten m Stellen. Wir nehmen weiter an, daß diesem Prozeß eine gewisse Gesetzmäßigkeit zugrunde liege, sodaß es eine *Modellfunktion*

$$b(t) = \varphi(t; x_1, \ldots, x_n) \tag{4.2.4}$$

mit den n unbekannten Parametern $x_1, \ldots, x_n$ gebe. Nun lautet die Aufgabe, diese Parameter aus den Messungen (4.2.2) so zu bestimmen, daß der betrachtete Prozeß mit Ausnahme gewisser Toleranzen angemessen modelliert wird.



In aller Regel wird der Fall $m \geq n$ vorliegen, was bedeutet, daß man (viel) mehr Messdaten als Parameter hat. Eine weitere Erschwernis ist, daß diese Daten mit Fehlern behaftet sind, sodaß die x_i so zu bestimmen sind, daß

$$\sum_{i=1}^{m} \omega_i (b_i - \varphi(t_i; x_1, \ldots, x_n))^2 = \min \tag{4.2.5}$$

mit *Gewichten* $\omega_i > 0$ wird. Diese Lösungsmethode nennt man GAUSSsche Fehlerquadratmethode. ∎

Wenn φ linear von $x_1, \ldots, x_n$ abhängt, also für jedes $i = 1, \ldots, m$

$$\varphi(t_i; x_1, \ldots, x_n) = \sum_{j=1}^{n} a_{ij} x_j \tag{4.2.6}$$

gilt, so heißt die Aufgabe 4.2.1 ein *lineares Ausgleichsproblem*. In der Statistik heißt dies auch *lineare Regression*. Für den Rest dieses Kapitels nehmen wir an, daß die Beziehung (4.2.6) gilt, was uns die Lösung von (4.2.5) leichter macht. In Matrix-Vektorform lautet (4.2.5)

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2 = \|Ax - b\|_2^2 = \min \tag{4.2.7}$$

mit $A := (a_{ij})_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}} \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$, was für unsere Aufgabe bedeutet: Finde das $x^* \in \mathbb{R}^n$, für das

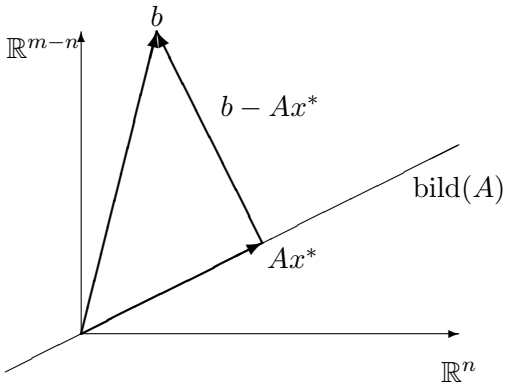
$$\|Ax^* - b\|_2 = \min_{x \in \mathbb{R}^n} \|Ax - b\|_2. \tag{4.2.8}$$

Dieses x^* heißt dann *Lösung* dieses i.a. überbestimmten linearen Gleichungssystems $Ax = b$. Man beachte, daß diese „Lösung“ die Gleichung $Ax = b$ im allgemeinen nicht erfüllt.

Bemerkung 4.2.9. Prinzipiell ist die in (4.2.8) durchgeführte Normierung in jeder beliebigen Norm möglich. Man unterscheidet folgende Begrifflichkeiten:

- $\|\cdot\|_1$: lineare Optimierung (wird häufig in den Wirtschaftswissenschaften verwendet)
- $\|\cdot\|_\infty$: Tschebyscheff'sche Ausgleichsrechnung
- $\|\cdot\|_2$: am häufigsten in den Anwendungen verwendete Norm. Dieses Vorgehen läßt sich auch statistisch bzw. wahrscheinlichkeitstheoretisch interpretieren und macht (4.2.8) leicht lösbar. Es gibt eine geometrische Anschauung, da $\|\cdot\|_2 = \sqrt{(\cdot,\cdot)}$ mit dem Euklidischen Skalarprodukt $(\cdot,\cdot)$ gilt. Wir werden in diesem Kapitel daher stets die Euklidische Norm verenden.

■



Geometrische Interpretation zum dritten Punkt von Bemerkung 4.2.9: finde zu gegebenem $b \in \mathbb{R}^m$ das $x^* \in \mathbb{R}^n$, für das $b - Ax^*$ senkrecht auf der Menge bild$(A) = \{x \in \mathbb{R}^n \mid Ax \in \mathbb{R}^m\}$ steht. Aus dieser geometrischen Interpretation leiten wir nun die *Normalengleichungen* ab: Wir haben gesehen, daß genau dann

$$\|Ax - b\|_2 = \min_{x \in \mathbb{R}^n} \tag{4.2.10}$$

ist, wenn $Ax - b \perp$ bild(A). Unter Ausnutzung der Eigenschaft $\sqrt{(\cdot,\cdot)} = \|\cdot\|_2$ übersetzt sich dies zu

$$(w, Ax - b) = 0$$

für alle $w \in$ bild(A).

Ersetzen von w durch ein $y \in \mathbb{R}^n$ liefert

$$(Ay, Ax - b) = 0.$$

Mit Verwendung des Matrixproduktes anstelle des Euklidischen Skalarprodukts gilt nun

$$(Ay)^T(Ax - b) = 0,$$

was gleichwertig mit

$$y^T A^T (Ax - b) = 0$$

für jedes $y \in \mathbb{R}^m$ ist. Dies schreiben wir wieder als Skalarprodukt durch

$$(y, A^T Ax - A^T b) = 0.$$

womit wir schließlich

$$A^T Ax = A^T b \tag{4.2.11}$$

als gesuchte *Normalengleichungen* erhalten. Damit können wir als erstes Ergebnis festhalten:

Satz 4.2.12. $x^* \in \mathbb{R}^n$ ist Lösung des linearen Ausgleichsproblems (4.2.8) genau dann, wenn x^* die Normalengleichungen (4.2.11) löst. Insbesondere ist x^* eindeutig, wenn A vollen Rang hat, denn wir haben in 3.1.2 gesehen, daß $A^T A$ in diesem Fall invertierbar ist. ■

Zum Schluß noch dieses Abschnitts noch eine

Bemerkung 4.2.13. Falls A vollen Rang hat, ist $A^T A$ symmetrisch positiv definit. Dies haben wir am Anfang Kapitel 3.4 bereits gesehen. ■

4.3 Orthogonale Projektionen auf einen Unterraum

Die oben abgeleitete Beziehung $\|Ax-b\|_2 = \min \Leftrightarrow Ax-b \perp \text{bild}(A)$, läßt sich auch allgemeiner fassen, was wir später noch benötigen werden.

Aufgabe 4.3.1. Gegeben sei ein (nicht notwendigerweise vollständiger) Vektorraum V mit Skalarprodukt $\langle\cdot,\cdot\rangle : V \times V \to \mathbb{R}$ und der dadurch induzierten Norm $\|\cdot\| = \sqrt{\langle\cdot,\cdot\rangle}$. Sei $U_n \subset V$ ein n-dimensionaler Teilraum von V. Zu vorgegebenem $v \in V$ bestimme man ein $u_n \in U_n$ mit der Eigenschaft

$$\|u_n - v\| = \min_{u\in U_n} \|u - v\|. \tag{4.3.2}$$

Dieses $u_n \in U_n$ existiert, da U_n endlichdimensional ist. ∎

Unter der vorhin angekündigten Verallgemeinerung bekommen wir nun

Satz 4.3.3. Unter den Voraussetzungen von Aufgabe 4.3.1 gilt, daß

$$\|u_n - v\| = \min_{u\in U_n} \|u - v\| \tag{4.3.4}$$

ist äquivalent zu

$$\langle u_n - v, u\rangle = 0 \tag{4.3.5}$$

für alle $u \in U_n$ ist, d.h. $u_n - v \perp_{\langle\cdot,\cdot\rangle} U_n$.

Beweis: Wir zeigen zunächst, daß die Bedingung hinreichend ist: Sei also $u_n \in U_n$ so gewählt, daß (4.3.5) gilt, und sei $u \in U_n$ beliebig, also ist $u - u_n \in U_n$. Für ein beliebig vorgegebenes $v \in V$ gilt nun

$$\begin{aligned} \|u-v\|^2 &= \|(u-u_n)+(u_n-v)\|^2 \\ &= \|u-u_n\|^2 + 2\langle u-u_n, u_n-v\rangle + \|u_n-v\|^2. \end{aligned}$$

Da nun das Skalarprodukt in der rechten Gleichung aufgrund von (4.3.5) verschwindet, ist dies gleich

$$\|u-v\|^2 = \|u-u_n\|^2 + \|u_n-v\|^2.$$

Da nun der erste Summand nichtnegativ ist, folgt

$$\|u-v\| \geq \|u_n-v\|,$$

woraus die Behauptung folgt. Die Gleichheit gilt im Fall $u_n = u$. Nun zur Notwendigkeit der Bedingung: Seien $v \in V$ gegeben und $u_n \in U_n$ Lösung von (4.3.4). Weiter sei $\hat{u} \in U_n$ so gewählt, daß $\alpha := \langle u_n - v, \hat{u}\rangle$ nicht verschwindet, was wir zum Widerspruch mit (4.3.4) führen wollen. Mit $\tilde{u} := u_n - \frac{\alpha}{\|\hat{u}\|^2}\hat{u} \in U_n$ gilt

$$\begin{aligned} \|\tilde{u}-v\|^2 &= \|u_n - v - \frac{\alpha}{\|\hat{u}\|^2}\hat{u}\|^2 \\ &= \|u_n-v\|^2 - 2\frac{\alpha^2}{\|\hat{u}\|^2}\langle u_n - v, \hat{u}\rangle + \frac{\alpha^2}{\|\hat{u}\|^4}\|\hat{u}\|^2 \\ &= \|u_n-v\|^2 - \frac{\alpha^2}{\|\hat{u}\|^2}. \end{aligned}$$

Also ist

$$\|\tilde{u}-v\| < \|u_n-v\|$$

im Widerspruch zu (4.3.4). ∎

Die Lösung von Aufgabe 4.3.1 ist also die *orthogonale Projektion* von v bezüglich $\langle\cdot,\cdot\rangle$ auf U_n. Dies macht die Lösung von Aufgabe 4.3.1 natürlich besonders leicht, wenn eine orthogonale Basis von U_n bezüglich $\langle\cdot,\cdot\rangle$ bekannt ist. Diese ist über das Verfahren von GRAM–SCHMIDT stets konstruierbar. Sei $\{\varphi_1,\ldots,\varphi_n\}$ eine Orthonormalbasis von U_n bezüglich $\langle\cdot,\cdot\rangle$, d.h.

$$\langle\varphi_i,\varphi_j\rangle = \delta_{ij} \tag{4.3.6}$$

für alle $i,j=1,\ldots,n$. Dazu noch

Bemerkung 4.3.7. Falls $\{\varphi_1,\ldots,\varphi_n\}$ eine Orthonormalbasis von U_n ist, so hat jedes $u\in U_n$ eine eindeutige Darstellung

$$u=\sum_{j=1}^{n}\langle u,\varphi_j\rangle\varphi_j. \tag{4.3.8}$$

In dieser Gleichung bezeichnet man $\langle u,\varphi_j\rangle$ als verallgemeinerte FOURIER–Koeffizienten.

Beweis: Sei $v_n := u-\sum_{j=1}^{n}\langle u,\varphi_j\rangle\varphi_j$. Wir müssen $v_n=0$ zeigen. Nach Konstruktion von v_n gilt

$$\begin{aligned}\langle v_n,\varphi_i\rangle &= \left\langle u-\sum_{j=1}^{n}\langle u,\varphi_j\rangle\varphi_j,\varphi_i\right\rangle\\ &= \langle u,\varphi_i\rangle-\sum_{j=1}^{n}\langle u,\varphi_j\rangle\langle\varphi_j,\varphi_i\rangle\\ &= \langle u,\varphi_i\rangle-\langle u,\varphi_i\rangle \quad = \ 0\end{aligned}$$

für jedes $i=1,\ldots,n$. Also $v_n\perp U_n$. Da aber $v_n\in U_n$, muß $v_n=0$ sein.
Die Eindeutigkeit der Darstellung (4.3.8) folgt für zwei Darstellungen $u=\sum_{j=1}^{n}u_j\varphi_j=\sum_{j=1}^{n}\tilde{u}_j\varphi_j$ mit Koeffizienten $u_j,\tilde{u}_j$ wegen der Orthonormalität der φ_i aus $\langle u,\varphi_i\rangle=\sum_{j=1}^{n}u_j\langle\varphi_j,\varphi_i\rangle=u_i$ und andererseits $\langle u,\varphi_i\rangle=\sum_{j=1}^{n}\tilde{u}_j\langle\varphi_j,\varphi_i\rangle=\tilde{u}_i$, also $u_i=\tilde{u}_i$ für jedes $i=1\ldots,n$. ∎

Wir fassen die Ergebnisse dieses Abschnitts zusammen als

Satz 4.3.9. Sei $\{\varphi_1,\ldots,\varphi_n\}$ eine Orthonormalbasis von $U_n\subset V$. Für jedes $v\in V$ löst dann

$$u_n := \sum_{j=1}^{n}\langle v,\varphi_j\rangle\varphi_j \tag{4.3.10}$$

die Aufgabe 4.3.1.

Beweis: Mit (4.3.6) und (4.3.10) gilt

$$\begin{aligned}\langle u_n-v,\varphi_i\rangle &= \left\langle\sum_{j=1}^{n}\langle v,\varphi_j\rangle\varphi_j-v,\varphi_i\right\rangle\\ &= \sum_{j=1}^{n}\langle v,\varphi_j\rangle\langle\varphi_j,\varphi_i\rangle-\langle v,\varphi_i\rangle \quad = \ 0\end{aligned}$$

für alle $i=1,\ldots,n$. Da $\{\varphi_1,\ldots,\varphi_n\}$ eine Orthonormalbasis von U_n ist, folgt $\langle u_n-v,u\rangle=0$ für alle $u\in U_n$ und mit Satz 4.3.3 dann die Behauptung. ∎

4.4 Singulärwertzerlegung und Pseudoinverse

In Satz 4.2.12 haben wir gesehen, daß ein x^* genau dann das lineare Ausgleichsproblem (4.2.8) löst, wenn x^* die Normalengleichungen (4.2.11), nämlich

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

mit $A \in \mathbb{R}^{m\times n}$ wobei $m \geq n$ ist, löst. Insbesondere zieht der Fall $\text{rang}(A) = n$ nach sich, daß $A^T A$ invertierbar ist, woraus für x^* folgt

$$x^* = (A^T A)^{-1} A^T b.$$

In dieser Gleichung nennt man die Matrix $(A^T A)^{-1} A^T$ *Pseudoinverse* von A und bezeichnet sie mit A^+. Der Name Pseudoinverse kommt daher, daß hier $A^+ A = I$ ist (da hier $m \geq n$ und rang $A = n$ vorausgesetzt wurde), jedoch im allgemeinen $AA^+ \neq I$ ist. Allgemein definiert man die Pseudoinverse aber über die *Singulärwertzerlegung*. Dazu folgender

Satz 4.4.1. Sei $A \in \mathbb{R}^{m\times n}$ beliebig mit $\text{rang}(A) = p \leq \min(m, n)$. Dann gibt es orthogonale Matrizen $U \in \mathbb{R}^{m\times m}$ und $V \in \mathbb{R}^{n\times n}$, so daß

$$A = U\Sigma V^T \tag{4.4.2}$$

mit $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p, 0, \ldots, 0) \in \mathbb{R}^{m\times n}$, wobei man $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$ als die *Singulärwerte* von A bezeichnet.

Beweis: Unsere Fassung findet man in [DH], p. 147f.. Wer sich für den komplexen Fall interessiert, möge z.B. in [SB], p. 21f. nachsehen. ∎

Gleichung (4.4.2) heißt entsprechend *Singulärwertzerlegung* (singular–value decomposition). Mit den Eigenwerten von A hängen die Singulärwerte folgendermaßen zusammen:
Es gilt $\sigma_i = \sqrt{\lambda_i(A^T A)}$, wobei $\lambda_i(A^T A)$ der i-te Eigenwert von $A^T A$ ist. Hieran sieht man, daß alle Singulärwerte offenbar reell sind. Ein Verfahren zur praktischen Berechnung der Singulärwertzerlegung werden wir in Abschnitt 5.8 besprechen. Um unsere Betrachtungen fortzusetzen, definieren wir nun passend zur Matrix Σ in (4.4.2)

$$\Sigma^+ := \text{diag}(\sigma_1^{-1}, \ldots, \sigma_p^{-1}, 0, \ldots, 0) \in \mathbb{R}^{n\times m} \tag{4.4.3}$$

und erhalten die am Anfang angekündigte

Definition 4.4.4. Sei $A \in \mathbb{R}^{m\times n}$ und $A = U\Sigma V^T$ die zugehörige Singulärwertzerlegung. Dann ist die *Pseudoinverse* von A gegeben durch

$$A^+ := V\Sigma^+ U^T \in \mathbb{R}^{n\times m}. \tag{4.4.5}$$

∎

Wichtige Eigenschaften der Pseudoinversen sind

Eigenschaften 4.4.6. Für A und A^+ aus (4.4.5) gelten die Moore–Penrose–Axiome

$$\begin{aligned} (A^+A)^T &= A^+A \\ (AA^+)^T &= AA^+ \\ A^+AA^+ &= A^+ \\ AA^+A &= A. \end{aligned}$$

∎

Damit haben wir

Satz 4.4.7. Seien $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ beliebig und $b \in \mathbb{R}^m$, dann ist

$$x^* = A^+ b \tag{4.4.8}$$

diejenige Lösung des Ausgleichproblems (4.2.8), die die kleinste $\|\cdot\|_2$–Norm hat. D.h. aber, daß $\|Ax^* - b\|_2$ ist minimal und nicht $\|x^*\|_2$. ∎

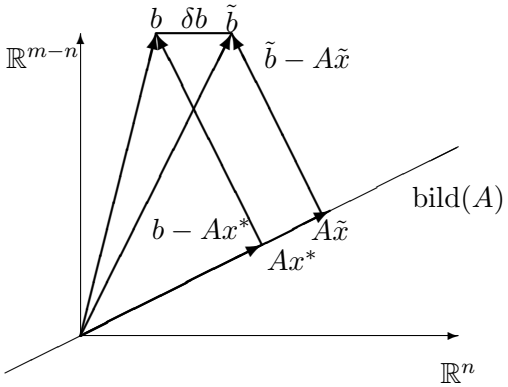
Einen Algorithmus zur praktischen Berechnung werden wir in Abschnitt 4.6 konstruieren. Wir wollen uns zunächst noch ein paar Gedanken zur Kondition dieses Problems machen.

4.5 Kondition des linearen Ausgleichsproblems

In diesem Abschnitt sei die Matrix $A \in \mathbb{R}^{m\times n}$ stets mit $\operatorname{rang}(A) = n$. Bisher haben wir für quadratische, nicht singuläre Matrizen B die Kondition immer als $\kappa_2(B) = \|B\|_2 \cdot \|B^{-1}\|_2$ berechnet. Da nun aber $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ nicht quadratisch ist, brauchen wir eine Verallgemeinerung des Konditionsbegiffs. Daher definieren wir

$$\kappa_2(A) := \frac{\max\limits_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}}{\min\limits_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}} \tag{4.5.1}$$

Um nun der Frage der Kondition von (4.2.8) nachzugehen, gehen wir der Frage nach, wie sich Störungen in A und b auf die Lösung x^* auswirken. Wir beginnen mit Störungen in b. Sei also x^* die exakte Lösung von (4.2.8) und $\tilde{x}$ Lösung des gestörten Problems $\|A\tilde{x} - \tilde{b}\|_2 = \min$.



Es bezeichne θ den Winkel zwischen b und $\operatorname{bild}(A)$. Wegen $Ax^* - b \perp \operatorname{bild}(A)$ läßt sich θ mit den trigonometrischen Funktionen ausdrücken durch

$$\cos(\theta) = \frac{\|Ax^*\|_2}{\|b\|_2} \tag{4.5.2a}$$

und

$$\sin(\theta) = \frac{\|b - Ax^*\|_2}{\|b\|_2} \tag{4.5.2b}$$

Damit können wir für die Kondition folgende Aussage treffen.

Satz 4.5.3. Für die Kondition des linearen Ausgleichsproblems bezüglich Störungen in b gilt

$$\frac{\|\tilde{x} - x^*\|_2}{\|x^*\|_2} \leq \frac{\kappa_2(A)}{\cos(\theta)} \cdot \frac{\|\tilde{b} - b\|_2}{\|b\|_2}$$

Beweis: Man findet diesen Satz nebst anderen interessanten Aussagen in [S], Kapitel 4.8.3, sowie in [DH], Kapitel 3.1.3. ∎

Offenbar hängt die Kondition von (4.2.8) nicht nur von $\kappa_2(A)$ (wie bei linearen Gleichungssystemen) ab, sondern auch vom *Winkel* θ zwischen b und $\operatorname{bild}(A)$. Wir gehen über zu Störungen in der Matrix A. Sei $\tilde{A}$ die gestörte Matrix, womit sich das gestörte Problem $\|\tilde{A}\tilde{x} - b\|_2 = \min$ ergibt.

Satz 4.5.4. Für die Kondition des linearen Ausgleichsproblems (4.2.8) bezüglich Störungen in A gilt

$$\frac{\|\tilde{x} - x^*\|_2}{\|x^*\|} \leq \left(\kappa_2(A) + \kappa_2(A)^2 \tan(\theta)\right) \frac{\|\tilde{A} - A\|_2}{\|A\|_2}$$

Beweis: Beweise findet man wieder in [DH], Kapitel 3.1.3 und [S], Kapitel 4.8.3. ∎

Man beachte noch, daß mit

$$\tan(\theta) = \frac{\sin(\theta)}{\cos(\theta)} = \frac{\frac{\|b - Ax^*\|_2}{\|b\|_2}}{\frac{\|Ax^*\|_2}{\|b\|_2}} = \frac{\|b - Ax^*\|_2}{\|Ax^*\|_2}$$

folgende Aussagen über die Kondition getroffen werden können: Ist $\|b - Ax^*\|_2 \ll \|b\|_2$, das *Residuum* also klein, dann ist mit $\cos(\theta) \approx 1$ und $\tan(\theta) \approx 0$ also θ klein, und das Ausgleichsproblem verhält sich konditionell wie ein lineares Gleichungssystem (in den Sätzen 4.5.3 und 4.5.4 spielt nur der Faktor $\kappa_2(A)$ eine Rolle). Wenn aber $\|b - Ax^*\|_2 \approx \|b\|_2$ ist, dann $\cos(\theta) \ll 1$ und damit $\tan(\theta) \gg 1$. Hier treten somit wesentlich andere Effekte auf, denn das lineare Ausgleichsproblem ist hier schlecht konditioniert. Ist also die Lösung x^* nicht bekannt, sollte man besser ein stabiles Verfahren zur Berechnung anwenden. Dieses wollen wir nun konstruieren.

4.6 Numerische Lösung von linearen Ausgleichsproblemen

Unser Vorgehen beginnt wieder bei den Normalengleichungen. Wir haben in Sektion 4.2 mit Satz 4.2.12 gesehen, daß x^* genau dann (4.2.8) $\min\limits_{x \in \mathbb{R}^n} \|Ax - b\|_2$ löst, wenn es die Normalengleichungen (4.2.11) $A^T Ax = A^T b$ löst. Weiter gab es eine eindeutige Lösung, wenn $\operatorname{rang}(A) = n$ war. Diesen Fall wollen zunächst betrachten. Wir wissen aus Bemerkung 4.2.13, daß $A^T A$ in dieser Situation positiv definit ist. Also können wir die Cholesky–Zerlegung aus Abschnitt 3.4 anwenden und erhalten einen ersten Algorithmus:

Algorithmus 4.6.1. Berechne

1.) $A^T A$ und $A^T b$.

2.) Cholesky–Zerlegung $LDL^T = A^T A$.

3.) Löse $Ly = A^T b$ und $L^T x = D^{-1} y$ durch Vorwärts- und Rückwärtseinsetzen.

∎

Dieser Algorithmus hat aber Nachteile. Für große $m \gg n$ ist die explizite Berechnung von $A^T A$ aufwendig. Dabei besteht zusätzlich die Gefahr von Auslöschungseffekten. Nach Satz 4.5.4 erfolgt die Fehlerverstärkung bei einer Störung in A mit dem Fehler $\kappa_2(A^T A) = \kappa_2(A)^2$, also mit quadrierter Kondition. Da aber bei linearen Ausgleichsproblemen oft der Fall $\kappa_2(A) \gg 1$ vorliegt, ist Algorithmus 4.6.1 nicht sehr stabil. Also sollte man diesen Ansatz nur bei gut konditioniertem A verwenden.

Besser ist dagegen eine Lösung über die QR-Zerlegung. Wir bleiben zunächst aber weiter beim Fall $\operatorname{rang}(A) = n$. Nach Satz 3.7.4 gilt mit orthogonalem $Q \in \mathbb{R}^{m \times m}$

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2 \quad = \quad \min_{x \in \mathbb{R}^n} \|Q(Ax - b)\|_2$$

$$= \min_{x\in\mathbb{R}^n} \|QAx - Qb\|_2$$

Dies nutzen wir aus, indem wir ein $Q \in \mathbb{O}_m(\mathbb{R})$ berechnen mit

$$QA = R =: \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}$$

wobei $\hat{R} \in \mathbb{R}^{n\times n}$ in oberer Dreiecksgestalt und $0 \in \mathbb{R}^{(m-n)\times n}$ sind. Weiter schreiben wir

$$Qb = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

mit $b_1 \in \mathbb{R}^n$ und $b_2 \in \mathbb{R}^{m-n}$. Daraus folgt

$$\begin{aligned} \|QAx - Qb\|_2^2 &= \|Rx - Qb\|_2^2 \\ &= \|\hat{R}x - b_1\|_2^2 + \|b_2\|_2^2 \end{aligned}$$

Der zweite Term hängt nicht von x ab. Also ist $\|QAx - Qb\|_2$ minimal, wenn der erste Term verschwindet, d.h. wenn

$$x = \hat{R}^{-1}b_1.$$

Man beachte noch, daß für das *Residuum* $r := Ax - b \neq 0$ damit

$$\|r\|_2 = \|b_2\|_2 \tag{4.6.2}$$

folgt. Damit haben wir den ersten Satz bewiesen:

Satz 4.6.3. Sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und $\operatorname{rang}(A) = n$. Weiter seien $b \in \mathbb{R}^m$ und $Q \in \mathbb{R}^{m\times m}$ orthogonal mit

$$QA = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} = R \quad \text{und} \quad Qb = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{4.6.4}$$

wobei $\hat{R}$ eine obere Dreiecksmatrix und $b_1 \in \mathbb{R}^n$ sowie $b_2 \in \mathbb{R}^{m-n}$ seien. Dann ist $x^* := \hat{R}^{-1}b_1$ die Lösung des linearen Ausgleichsproblems (4.2.8) $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2$, und $\|Ax^* - b\|_2 = \|b_2\|_2$. ∎

Jetzt betrachten wir den schwierigeren Fall $p = rang(A) < n$. Die praktische Berechnung der „kleinsten“ Lösung (4.4.8) des Ausgleichsproblems (4.2.8) funktioniert wie folgt: Seien $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $p = \operatorname{rang}(A) \leq \min(m, n)$. Nach Satz 3.7.34 existiert nun ein orthogonales $Q \in \mathbb{R}^{m\times m}$ mit

$$QA = \begin{pmatrix} \hat{R} & \hat{S} \\ 0 & 0 \end{pmatrix} \tag{4.6.5}$$

wobei $\hat{R} \in \mathbb{R}^{p\times p}$ eine invertierbare obere Dreiecksmatrix und $\hat{S} \in \mathbb{R}^{p\times(n-p)}$ sind. Nun zerlegen wir x und Qb analog in

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{und} \quad Qb = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

mit $x_1, b_1 \in \mathbb{R}^p$ und $x_2, b_2 \in \mathbb{R}^{n-p}$. Nun haben wir

Lemma 4.6.6. x ist Lösung von (4.2.8) genau dann, wenn

$$x_1 = \hat{R}^{-1}b_1 - \hat{R}^{-1}\hat{S}x_2 \tag{4.6.7}$$

gilt.

Beweis: Der Term

$$\begin{aligned} \|Ax-b\|_2^2 &= \|QAx - Qb\|_2^2 \\ &= \left\| \begin{pmatrix} \hat{R}x_1 + \hat{S}x_2 - b_1 \\ b_2 \end{pmatrix} \right\|_2^2 \\ &= \|\hat{R}x_1 + \hat{S}x_2 - b_1\|_2^2 + \|b_2\|_2^2 \end{aligned}$$

wird genau dann minimal, wenn $\hat{R}x_1 = b_1 - \hat{S}x_2$. ∎

Hieraus leiten wir weiter ab:

Lemma 4.6.8. Sei $p = \operatorname{rang}(A) < n$. Definiere $W := \hat{R}^{-1}\hat{S} \in \mathbb{R}^{p\times(n-p)}$ und $v := \hat{R}^{-1}b_1$. Dann ist die „kleinste“ Lösung von (4.2.8) gegeben durch $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ mit $x_1 \in \mathbb{R}^p$ und $x_2 \in \mathbb{R}^{(n-p)}$ als Lösung von

$$(I + W^TW)x_2 = W^Tv \quad \text{und} \quad x_1 = v - Wx_2 \tag{4.6.9}$$

Beweis: Aus Lemma 4.6.6 wissen wir $x_1 = \hat{R}^{-1}b_1 - \hat{R}^{-1}\hat{S}x_2 =: v - Wx_2$. Wenn wir dies nun in $\|x\|_2$ einsetzen, erhalten wir:

$$\begin{aligned} \|x\|_2^2 &= \|x_1\|_2^2 + \|x_2\|_2^2 = \|v - Wx_2\|_2^2 + \|x_2\|_2^2 \\ &= \|v\|_2^2 - 2\langle v, Wx_2\rangle + \langle Wx_2, Wx_2\rangle + \langle x_2, x_2\rangle \\ &= \|v\|_2^2 - 2\langle W^Tv, x_2\rangle + \langle W^TWx_2, x_2\rangle + \langle x_2, x_2\rangle \\ &= \|v\|_2^2 + \langle x_2 + W^TWx_2 - 2W^Tv, x_2\rangle \\ &= \|v\|_2^2 + \langle (I + W^TW)x_2 - 2W^Tv, x_2\rangle =: \varphi(x_2) \end{aligned}$$

Nun berechnen wir das Minimum von φ:

$$\begin{aligned} \varphi'(x_2) &= -2W^Tv + 2(I + W^TW)x_2 \\ \varphi''(x_2) &= 2(I + W^TW) \end{aligned}$$

Da φ quadratisch in x_2 ist, ist $\varphi'(x_2) = 0$ genau dann, wenn $(I + W^TW)x_2 = W^Tv$ ist. Da $2(I + W^TW)$ symmetrisch positiv definit ist, liegt ein Minimum vor. ∎

Wegen der symmetrisch positiv definiten Struktur von $I + W^TW$ errechnet man x_2 mit Cholesky–Zerlegung. Nun haben wir alles, was wir brauchen, um den in diesem Kapitel gesuchten Algorithmus zur Singulärwertzerlegung und zur Lösung des linearen Ausgleichsproblems erstellen zu können:

Algorithmus 4.6.10. Wir berechnen $x^* = A^+b$ als Lösung von (4.2.8) über QR–Zerlegung von A. Seien dazu $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$. Berechne nun

1.) QR-Zerlegung (4.6.5) von A mit $p = \operatorname{rang}(A)$, $Q \in \mathbb{R}^{m\times n}$ orthogonal, $\hat{R} \in \mathbb{R}^{p\times p}$ invertierbare obere Dreiecksmatrix und $\hat{S} \in \mathbb{R}^{p\times(n-p)}$.

2.) $W \in \mathbb{R}^{p\times(n-p)}$ aus $\hat{R}W = \hat{S}$

3.) CHOLESKY-Zerlegung von $I + W^TW$ als $(I + W^TW) = LL^T$, wobei $L \in \mathbb{R}^{(n-p)\times(n-p)}$ eine untere Dreiecksmatrix ist

4.) $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} := Qb$ mit $b_1 \in \mathbb{R}^p, b_2 \in \mathbb{R}^{m-p}$

5.) $v \in \mathbb{R}^p$ aus $\hat{R}v = b_1$

6.) $x_2 \in \mathbb{R}^{n-p}$ aus $LL^Tx_2 = W^Tv$

7.) setze noch $x_1 := v - Wx_2$

■

Dann ist $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A^+b$. Hierbei muß man für mehrere Seiten b die Schritte 1.)–3.) natürlich nur einmal ausführen. Da sich in der Praxis nicht immer $\text{rang}(A) = \min(m, n)$ garantieren lässt, sollte man im allgemeinen Algorithmus 4.6.10 verwenden. Außerdem ist in diesem Algorithmus der Fehler bedingt durch die Stabilität der QR–Zerlegung stets unter Kontrolle, denn der Fehler verstärkt sich nur mit $\kappa_2(A) = \kappa_2(\hat{R})$ aufgrund der Orthogonalität von Q.

5 Berechnung von Eigenwerten und Eigenvektoren

5.1 Einleitung

Sei $A \in \mathbb{R}^{n\times n}$. In diesem Kapitel behandeln wir die folgende Aufgabe: Bestimme $\lambda \in \mathbb{C}$ und $v \in \mathbb{C}^n, v \not\equiv 0$, die die Eigenwertgleichung

$$Av = \lambda v \tag{5.1.1}$$

erfüllen. Hierbei heißt λ Eigenwert und v zugehöriger Eigenvektor zum Eigenwert λ.

Beispiele, in denen Eigenwerte eine große Rolle spielen, sind

- Berechnung von $\|A\|_2$ und $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$, wobei hier nur die extremen Eigenwerte gesucht sind, denn in Aufgabe 5 haben wir für symmetrisches A gezeigt, daß $\|A\|_2 = |\lambda_{max}(A)|$ bzw. $\|A^{-1}\|_2 = |\lambda_{min}(A)|^{-1}$ gilt. Da in den Anwendungen häufig symmetrische Matrizen vorliegen, ist die Wichtigkeit der Eigenwerte also evident. Weiter wissen wir, das symmetrische Matrizen nur reelle Eigenwerte besitzen.
- für $A = A^T$ gilt $\rho(A) = \|A\|_2$, wobei man $\rho(A)$ den *Spektralradius* nennt. Für allgemeines A gilt: $\|A\|_2 = \sqrt{\rho(A^TA)}$.
- Systeme gekoppelter linearer gewöhnlicher Differentialgleichungen können mittels Basen von Eigenvektoren entkoppelt werden und sind dann einfacher zu lösen in der Form $\dot{x}_i = dx_i$ mit $d \in \mathbb{C}$ für alle $i = 1, \ldots, n$.
- Modellierung von Schwingungs- und Resonanzabläufen, die wieder mit gewöhnlichen Differentialgleichungen beschrieben werden können.

Passend zum letzten Punkt wollen wir uns nun ein wichtiges Beispiel aus den Anwendungen ansehen, bei dessen Lösung die Berechnung von Eigenwerten eine große Rolle spielt:

Beispiel 5.1.2 (Sturm–Liouville–Problem)**.** Das Sturm–Liouville–Problem handelt davon, die Zahlen λ und die Lösung $u(x)$ der Differentialgleichung

$$u''(x) + \lambda r(x)u(x) = 0 \tag{5.1.3}$$

mit $x \in [0, 1]$ und den Randbedingungen $u(0) = u(1) = 0$ zu finden. Diese Gleichung modelliert die gedämpfte Schwingung einer Feder. Es sei $r \in \mathcal{C}^0([0, 1])$ mit $r(x) > 0$ bereits bekannt. Dieses r stellt die Dichte der Feder im Punkt x dar. Falls $r(x) \equiv 1$ ist, so sind die Zahlen $\lambda = (k\pi)^2$ und $u(x) = \sin(k\pi x)$ für $k = 0, 1, 2, \ldots$ eine Lösung von (5.1.3), was man sich durch eine einfache Rechnung klar macht. Falls r aber nicht konstant ist, gibt es im allgemeinen keine geschlossene Formel, um alle Lösungen anzugeben. Also muß (5.1.3) numerisch über ein *Diskretisierungsverfahren* gelöst werden. Dazu betrachten wir $n+1$ auf $[0, 1]$ äquidistant verteilte Gitterpunkte $x_j = jh$ mit $j = 0, \ldots, n$ und $h = \frac{1}{n}$.

Nun berechnen wir die Lösung von u an den *Gitterpunkten* x_j folgendermaßen: werte $r(x), u(x)$ und $u''(x)$ an den Punkten x_j aus und ersetze $u''(x_j)$ durch den Differentialquotienten

$$u''(x) \approx \frac{u(x_j + h) - 2u(x_j) + u(x_j - h)}{h^2}. \tag{5.1.4}$$

Wir bezeichnen nun $u_j \approx u(x_j)$ und $u_{j+1} = u(x_{j+1})$. Später wird sich über die TAYLOR–Entwicklung zeigen, daß die Approximation (5.1.4) genau bis auf Terme 2. Ordnung ist, wenn u glatt genug ist. Damit erhält unsere Differentialgleichung (5.1.3) die *diskretisierte Form*

$$\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + \lambda r(x_j) u_j = 0 \tag{5.1.5}$$

für die *inneren Gitterpunkte* $j = 1, \ldots, n-1$. Zusammen mit den Randbedingungen erhalten wir damit ein System von $n-1$ Gleichungen in $n-1$ Unbekannten u_j:

$$-Bu + \lambda Du = 0, \tag{5.1.6}$$

wobei $u = (u_1, \ldots, u_n)^T$ und

$$B := \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad D := \begin{pmatrix} r(x_1) & & & & \\ & r(x_2) & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & r(x_{n-1}) \end{pmatrix}$$

mit $r(x_i) > 0$ seien. Mit $D^{\frac{1}{2}} = \operatorname{diag}(\sqrt{r(x_1)}, \ldots, \sqrt{r(x_n)})$ folgt

$$\begin{aligned} Bu = \lambda Du &\Leftrightarrow Bu = \lambda D^{\frac{1}{2}} D^{\frac{1}{2}} u \\ &\Leftrightarrow D^{\frac{-1}{2}} Bu = \lambda D^{\frac{1}{2}} u \end{aligned}$$

Setze $D^{\frac{1}{2}} u =: v$ und $A := D^{\frac{-1}{2}} B D^{\frac{-1}{2}}$. Das ist gerade eine Eigenwertgleichung (5.1.1)

$$Av = \lambda v.$$

Da A symmetrisch positiv definit ist, existiert eine Basis aus Eigenvektoren, und alle Eigenwerte sind reell und positiv. ∎

5.2 Theoretische Grundlagen

Seien $A \in \mathbb{R}^{n \times n}$ und $\lambda \in \mathbb{C}$ Eigenwert von A und $v \in \mathbb{C}^n, v \not\equiv 0$ zugehöriger Eigenvektor. Dann gilt

$$Av = \lambda v. \tag{5.2.1}$$

In diesem Unterabschnitt stellen wir einige Grundlagenbegriffe über Eigenwerte und Eigenvektoren zusammen, die wir zu einem großen Teil bereits in der Linearen Algebra kennengelernt haben.

Lemma 5.2.2. λ ist ein Eigenwert von A genau dann, wenn $\det(A - \lambda I) = 0$ ist. ∎

Man bezeichnet $p(\lambda) := \det(A - \lambda I) \in \mathcal{P}_n$ als *charakteristisches Polynom* der Matrix A. Dieses ist offenbar aus der Menge der Polynome vom Grad n und die Nullstellen dieses sind die Eigenwerte von A. Theoretisch könnte man also die Eigenwerte als Nullstellen dieses Polynoms berechnen. Dies hat aber für die praktische Nutzung einige entscheidende Nachteile, denn die Koeffizienten

von p müssen zunächst einmal aus A berechnet werden. Weiter hängen die Nullstellen oft sensibel von den Koeffizienten von p ab. Das Problem ist also schlecht konditioniert. Weiter müssen die Nullstellen für $n > 3$ alle mit iterativen Verfahren (z.B. mit dem NEWTON–Verfahren) bestimmt werden. Insgesamt erweist sich diese Methode also im allgemeinen als untauglich, um die Eigenwerte zu berechnen. Höchstens für kleine n könnte man dies noch akzeptieren. Um aber die schlechte Kondition der Berechnung der Eigenwerte aus dem charakteristischen Polynom in der Relation zur Kondition des eigentlichen Eigenwertproblems noch etwas besser zu herauszustellen, wollen wir in einem Beispiel doch einmal die Eigenwerte einer Matrix aus den Nullstellen des charakteristischen Polynoms berechnen.

Beispiel 5.2.3. Sei $A = I \in \mathbb{R}^{n\times n}$ mit den Eigenwerten $\lambda_i = 1$ und den Eigenvektoren e^i für $i = 1, \ldots, n$. Später wird sich zeigen, daß das Eigenwertproblem gut konditioniert in dem Sinne ist, daß, wenn μ Eigenwert von $A+\Delta A$ ist, wobei ΔA eine Störung der Matrix A mit $\|\Delta A\|_2 \leq \varepsilon$ bezeichne, dann aufgrund der Störungssätze der numerischen Linearen Algebra folgt

$$|1 - \mu| \leq \varepsilon. \tag{5.2.4}$$

Hier gilt nun unter Verwendung des binomischen Satzes

$$\begin{aligned} p(\lambda) &= \begin{vmatrix} 1-\lambda & & & & \\ & \ddots & & 0 & \\ & & \ddots & & \\ & 0 & & \ddots & \\ & & & & 1-\lambda \end{vmatrix} = (1-\lambda)^n \\ &= \sum_{k=0}^{n} \binom{n}{k} (-\lambda)^k = \binom{n}{0} - \binom{n}{1}\lambda + \binom{n}{2}\lambda^2 - \ldots + (-\lambda)^n \binom{n}{n} \end{aligned}$$

Bei einer kleinen Störung $\tilde{\varepsilon} > 0$ (z.B.) des ersten Koeffizienten $\binom{n}{0} = 1$ ergibt sich das gestörte Polynom

$$\begin{aligned} p_{\tilde{\varepsilon}}(\lambda) &= 1 - \tilde{\varepsilon} - \binom{n}{1}\lambda + \binom{n}{2}\lambda^2 + \ldots + (-\lambda)^n \binom{n}{n} \\ &= p(\lambda) - \tilde{\varepsilon} = (1-\lambda)^n - \tilde{\varepsilon} \end{aligned}$$

mit den Nullstellen

$$\begin{aligned} p_{\tilde{\varepsilon}}(\lambda) = 0 &\Leftrightarrow (1-\lambda)^n = \tilde{\varepsilon} \\ &\Leftrightarrow \lambda = \left\{ \begin{array}{ll} 1 + \varepsilon^{\frac{1}{n}}: & \text{falls } n \text{ ungerade} \\ 1 \pm \varepsilon^{\frac{1}{n}}: & \text{falls } n \text{ gerade} \end{array} \right\}. \end{aligned}$$

Mit $|1 - \lambda| = \varepsilon^{\frac{1}{n}} \gg \varepsilon$ für $n > 1$ und $0 < \varepsilon \ll 1$ ist dieser Rechenweg also unakzeptabel, da das Problem (5.2.4) gut konditioniert ist. ∎

Wir erinnern uns noch an einige weitere Tatsachen aus der Linearen Algebra.

- Nach dem *Fundamentalsatz der Algebra* hat das charakteristische Polynom

  $$p(\lambda) = \det(A - \lambda I)$$

  genau n (gezählt mit der entsprechenden algebraischen Vielfachheit) reelle oder komplexe Nullstellen $\lambda_1, \ldots, \lambda_n$.

- λ_i heißt *einfacher* Eigenwert, wenn die entsprechende Nullstelle des charakteristischen Polynoms einfach ist.
- Die Menge aller Eigenwerte von A, bezeichnet durch

  $$\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$$

  heißt *Spektrum* von A.
- Zwei Matrizen $A, B \in \mathbb{R}^{n\times n}$ heißen ähnlich, wenn es eine invertierbare Matrix $T \in \mathbb{R}^{n\times n}$ gibt, mit der Eigenschaft

  $$B = T^{-1}AT.$$

- ähnliche Matrizen haben das gleiche Spektrum, d.h. es gilt

  $$\sigma(A) = \sigma(T^{-1}AT)$$

  für eine beliebige invertierbare Matrix $T \in \mathbb{R}^{n\times n}$.

  Beweis: ähnliche Matrizen haben dasselbe charakteristische Polynom:

  $$\begin{aligned} \det(T^{-1}AT - \lambda I) &= \det(T^{-1}(A - \lambda I)T) \\ &= \det(T^{-1})\det(A - \lambda I)\det(T) \\ &= \det(A - \lambda I) \qquad \blacksquare \end{aligned}$$

- A heißt *diagonalisierbar*, wenn A ähnlich zu einer Diagonalmatrix ist.
- A ist genau dann diagonalisierbar, wenn A n verschiedene linear unabhängige Eigenvektoren hat.
- Hat A n verschiedene Eigenwerte, so ist A diagonalisierbar. $\blacksquare$

Wir werden in Zukunft zwei Sätze über die SCHUR'sche Normalform oder SCHUR–Faktorisierung benötigen. In der linearen Algebra haben wir gesehen, daß nicht jede Matrix diagonalisierbar ist. Aber eine Transformation auf obere Dreiecksgestalt ist immer möglich. Dies liefern uns die besagten SCHURschen Normalformen.

Satz 5.2.5 (Komplexe SCHUR–Faktorisierung)**.** Zu jeder Matrix $A \in \mathbb{C}^{n\times n}$ gibt es eine unitäre Matrix $Q \in \mathbb{C}^{n\times n}$ (d.h. $Q^{-1} = \bar{Q}^T =: Q^*$), sodaß

$$Q^*AQ = \begin{pmatrix} \lambda_1 & & & \\ & \ddots & & * \\ & & \ddots & \\ & 0 & & \ddots \\ & & & & \lambda_n \end{pmatrix} =: R. \tag{5.2.6}$$

Dabei sind $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von A.

Beweis: Beweise dieses Satzes findet man in [SB] p. 17f. und [GL] p. 313..

Die reelle Variante lautet

Satz 5.2.7 (Reelle SCHUR–Faktorisierung)**.** Zu jeder Matrix $A \in \mathbb{R}^{n\times n}$ gibt es eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$, sodaß

$$Q^T AQ = \begin{pmatrix} R_{11} & & & & \\ & \ddots & & * & \\ & & \ddots & & \\ & 0 & & \ddots & \\ & & & & R_{mm} \end{pmatrix} =: R \tag{5.2.8}$$

Dabei ist für jedes i entweder $R_{ii} \in \mathbb{R}$ oder $R_{ii} \in \mathbb{R}^{2\times 2}$. Im zweiten Fall hat R_{ii} ein Paar von konjugiert komplexen Eigenwerten. Die Menge aller Eigenwerte der R_{ii} mit $i = 1, \ldots, m$ bilden das Spektrum von A. Man nennt A in diesem Fall auch eine *Quasi–obere–Dreiecksmatrix*.

Beweis: Einen Beweis findet man wieder in [GL] p. 341f.. ∎

Man beachte noch, daß die Zerlegungen in (5.2.6) und (5.2.8) keineswegs eindeutig sind. Insgesamt haben wir aber für symmetrische Matrizen gezeigt, daß

Korollar 5.2.9. Jede symmetrische Matrix $A \in \mathbb{R}^{n\times n}$ läßt sich mittels einer orthogonalen Matrix Q durch eine ähnlichkeitstransformationen auf Diagonalgestalt

$$Q^{-1}AQ = D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \tag{5.2.10}$$

bringen. A hat somit nur reelle Eigenwerte und n linear unabhängige zueinander orthogonale Eigenvektoren, nämlich die Spalten von Q.

Beweis: Der Beweis folgt mit (5.2.8) und der Symmetrie von A. ∎

5.3 Kondition des Eigenwertproblems

Wir gehen in diesem Unterkapitel der Frage nach, wie stark sich die Eigenwerte und Eigenvektoren bei Störungen in A ändern. Dazu zunächst folgender

Satz 5.3.1. Sei $A \in \mathbb{R}^{n\times n}$ diagonalisierbar, d.h. es existiert eine Matrix $V \in \mathbb{R}^{n\times n}$ mit

$$V^{-1}AV = \operatorname{diag}(\lambda_1, \ldots, \lambda_n). \tag{5.3.2}$$

Sei μ ein Eigenwert der gestörten Matrix $A + \Delta A$. Dann gilt

$$\min_{1\le i\le n} |\lambda_i - \mu| \le \|V\|_p \|V^{-1}\|_p \|\Delta A\|_p \tag{5.3.3}$$

für alle $p = 1, 2, \ldots, \infty$.

Beweis: Beweise findet man wieder in [GL] und [SB]. ∎

Man vergleiche dieses Beispiel mit 5.2.3, in dem mit $A = I$ auch $V = I$ war, sich also die Kondition in hiesigem Sinne nicht verschlechterte. Hier beachte man aber, daß die absolute Kondition der Eigenwerte von $\kappa_p(V) = \|V\|_p\|V^{-1}\|_p$ abhängig ist, und *nicht* von $\kappa_p(A)$. Da die Spalten von V gerade die Eigenvektoren von A sind (vergleiche (5.3.2)), bedeutet 5.3.1 gerade, daß für eine diagonalisierbare Matrix die Kondition der Eigenvektorbasis (im Sinne von Maß $\kappa_p(V)$) eine große Rolle bei der Empfindlichkeit der Eigenwerte von A bezüglich Störungen spielt. Hierzu betrachten wir ein Beispiel.

Beispiel 5.3.4. Seien

$$A := \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

und die gestörte Matrix

$$A + \Delta A := \begin{pmatrix} 0 & 1 \\ \delta & 0 \end{pmatrix}$$

mit den Eigenwerten $\lambda_1 = \lambda_2 = 0$ von A und $\tilde{\lambda}_{1,2} = \pm\sqrt{\delta}$ von $A+\Delta A$ gegeben. Für die Kondition des Eigenwertproblems ergibt sich somit

$$\kappa_{\text{abs}} \geq \frac{|\tilde{\lambda}_1 - \lambda_1|}{\|\tilde{A} - A\|_2} = \frac{\sqrt{\delta}}{\delta} = \frac{1}{\sqrt{\delta}} \to \infty$$

für $\delta \to 0$. ∎

Offenbar kann das Eigenwertproblem für beliebige Matrizen also *sehr schlecht* konditioniert sein. Der folgende Satz zeigt aber, daß dieses Problem für *symmetrische Matrizen* stets gut konditioniert ist.

Satz 5.3.5. Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und μ ein Eigenwert der gestörten Matrix $A+\Delta A$. Dann gilt

$$\min_{1\leq i\leq n} |\lambda_i - \mu| \leq \|\Delta A\|_2. \tag{5.3.6}$$

Beweis: Aus Korollar 5.2.9 folgt, daß A sich mittels einer orthogonalen Matrix Q diagonalisieren läßt. Da für Q aufgrund der Orthogonalität $\kappa_2(Q) = 1$ gilt, folgt unsere Behauptung direkt aus Gleichung (5.3.3) für $p = 2$. ∎

5.4 Eigenwertabschätzungen

Wir rufen uns zunächst ein paar Eigenschaften von Eigenwerten ins Gedächtnis zurück.

Eigenschaften 5.4.1. Seien $A, B \in \mathbb{R}^{n\times n}$. Dann gilt

(i) Falls $\det(A) \neq 0$ und λ ein Eigenwert von A ist, so ist λ^{-1} Eigenwert von A^{-1}.

(ii) Ist $\lambda \in \sigma(A)$, dann ist $\alpha\lambda \in \sigma(\alpha A)$ für $\alpha \in \mathbb{C}$ beliebig.

(iii) Ist $\lambda \in \sigma(A)$, so ist $\lambda-\alpha \in \sigma(A-\alpha I)$. Hierbei nennt man α *Shift* oder *Spektralverschiebung*.

(iv) Ist $\lambda \in \sigma(A)$, so ist ebenfalls $\bar{\lambda} \in \sigma(A)$.

(v) Es gilt stets $\sigma(A) = \sigma(A^T)$.

(vi) Es gilt $\sigma(AB) = \sigma(BA)$.

∎

Wir wollen nun erreichen, aus einer direkt zugänglichen Information in A (z.B. die Einträge oder die Matrixnorm) Aussagen über die Eigenwerte von A treffen zu können, ohne explizite Rechnungen anstellen zu müssen. Dazu zunächst folgender

Satz 5.4.2. Es gilt $|\lambda| \leq \|A\|$ für jedes $\lambda \in \sigma(A)$ und jede Operatornorm.

Beweis: Sei $\lambda \in \sigma(A)$ und v zugehöriger Eigenvektor mit $\|v\| = 1$. Dann gilt

$$|\lambda| = |\lambda|\|v\| = \|\lambda v\| = \|Av\| \leq \max_{\|x\|=1} \|Ax\| = \|A\|.$$

∎

Daraus folgern wir insbesondere für den Spektralradius $\rho(A) := \max\{|\lambda| : \lambda \in \sigma(A)\}$

Korollar 5.4.3. Für den Spektralradius einer Matrix A gilt $\rho(A) \leq \|A\|$. ∎

Man kann zeigen, dass umgekehrt zu jedem $\varepsilon > 0$ eine Matrixnorm $\|\cdot\|_p$ existiert, sodaß die Ungleichung $\|A\|_p \leq \rho(A)+\varepsilon$ gilt. Nun kommen wir zu der in diesem Unterkapitel entscheidenden Aussage.

Satz 5.4.4 (Satz von Gerschgorin)**.** Seien

$$K_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\} \tag{5.4.5}$$

die Gerschgorin–Kreise. Dann gilt

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} K_i, \tag{5.4.6}$$

d.h. alle Eigenwerte liegen in der Vereinigung der Gerschgorin–Kreise.

Beweis: (übung) ∎

Der Bereich für die Eigenwerte kann weiter eingeschränkt werden durch

Korollar 5.4.7. Seien K_i^T die Gerschgorin–Kreise für A^T. Dann gilt

$$\sigma(A) \subseteq \left(\bigcup_{i=1}^{n} K_i\right) \cap \left(\bigcup_{i=1}^{n} K_i^T\right). \tag{5.4.8}$$

Falls A symmetrisch ist, sind alle Eigenwerte reell. Also gilt in diesem Fall

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} (K_i \cap \mathbb{R}). \tag{5.4.9}$$

Beweis: Die Aussagen folgen aus 5.4.1(v) zusammen mit (5.4.6). ∎

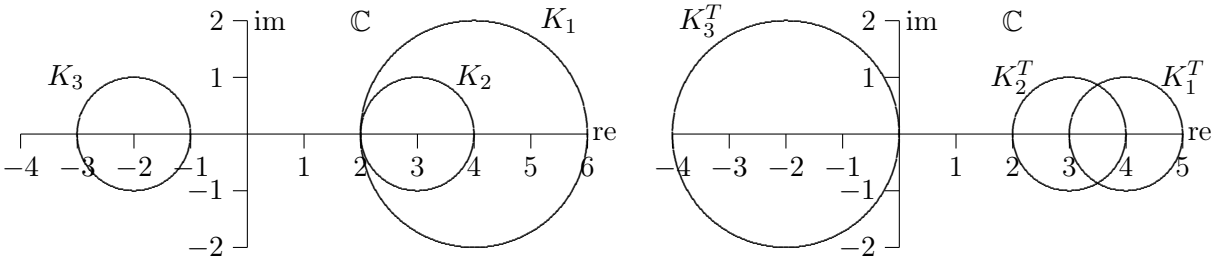
Beispiel 5.4.10. Wir betrachten die Gerschgorin–Kreise der Matrizen

$$A = \begin{pmatrix} 4 & 1 & -1 \\ 0 & 3 & -1 \\ 1 & 0 & -2 \end{pmatrix} \quad \text{und} \quad A^T = \begin{pmatrix} 4 & 0 & 1 \\ 1 & 3 & 0 \\ -1 & -1 & -2 \end{pmatrix}.$$

A hat das Spektrum $\sigma(A) = \{3.43 \pm 0.14i, -1.86\}$. Die Gerschgorin–Kreise von A und A^T sind:

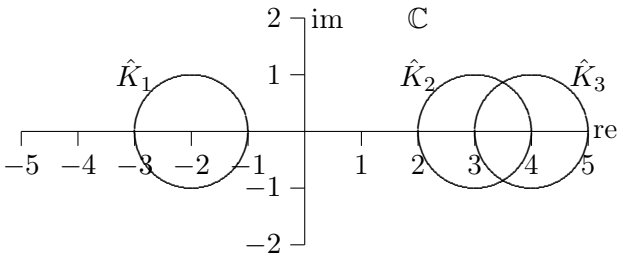
$$\begin{aligned} K_1 &= \{z : |z-4| \leq 2\} & K_1^T &= \{z : |z-4| \leq 1\} \\ K_2 &= \{z : |z-3| \leq 1\} & K_2^T &= \{z : |z-3| \leq 1\} \end{aligned}$$

$$K_3 = \{z : |z+2| \le 1\} \qquad K_3^T = \{z : |z+2| \le 2\}$$



Nimmt man nun deren Schnitt gemäß (5.4.8), so bleiben

$$\hat{K}_1 = \{z : |z+2| \le 1\}$$
$$\hat{K}_2 = \{z : |z-3| \le 1\}$$
$$\hat{K}_3 = \{z : |z-4| \le 1\}$$



übrig. Man kann sogar noch weiter spezifizieren, daß sich m Eigenwerte in nichtleerem Schnitt von m GERSCHGORIN–Kreisen befinden. ∎

In den nächsten Abschnitten werden wir verschiedene Verfahren zur numerischen Berechnung von Eigenwerten und Eigenvektoren vorstellen.

5.5 Vektoriteration

Die auf VON MISES zurückgehende (direkte) *Vektoriteration* (auch *Potenzmethode* oder *power iteration*), ist ein Verfahren zur Bestimmung des betragsgrößten Eigenwertes und des zugehörigen Eigenvektors einer Matrix A. Es liefert das Grundkonzept für die Entwicklung weiterer diesbezüglicher Verfahren. Der Einfachheit halber machen wir für die Konvergenzanalyse einige Annahmen. Wir gehen davon aus, daß A diagonalisierbar ist, also eine Basis von Eigenvektoren $v^1, \ldots, v^n$ des $\mathbb{C}^n$ mit $\|v^i\|_2 = 1$ existiert. Weiter gelte für die Eigenwerte $|\lambda_1| > |\lambda_2| \ge \ldots \ge |\lambda_n|$. Der dominante Eigenwert soll also einfach sein. Damit gilt $\lambda_1 \in \mathbb{R}$ und $v^1 \in \mathbb{R}$. Für die Vektoriteration brauchen wir einen beliebigen Startvektor $x^{(0)} \in \mathbb{R}^n$, der sich somit (theoretisch) darstellen läßt als

$$x^{(0)} = \sum_{j=1}^{n} c_j v^j.$$

Wir nehmen weiter an, daß $x^{(0)}$ so gewählt ist, daß $c_1 \neq 0$. Wenden wir nun die k-te Potenz von A auf $x^{(0)}$ an, so erhalten wir

$$A^k x^{(0)} = \sum_{j=1}^{n} c_j A^k v^j = \sum_{j=1}^{n} c_j \lambda_j^k v^j.$$

Also gilt:

$$x^{(k)} := A^k x^{(0)} = \lambda_1^k \left(c_1 v^1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k v^j \right) =: \lambda_1^k \left(c_1 v^1 + r^{(k)} \right), \tag{5.5.1}$$

wobei wegen $\left| \frac{\lambda_j}{\lambda_1} \right| < 1$ für $j = 2, \ldots, n$ folgt

$$r^{(k)} = \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k v^j \to 0 \tag{5.5.2}$$

für $k \to \infty$, was bedeutet, daß für wachsendes k in (5.5.1) der erste Eigenvektor dominiert. Um dies etwas präziser zu fassen, betrachten wir den Abstand zwischen den Unterräumen $T := \{\alpha v^1 : \alpha \in \mathbb{R}\}$ und $S^k := \{\alpha x^k : \alpha \in \mathbb{R}\}$:

$$d(S^k, T) := \min_{x \in S^k} \|x - v^1\|_2 = \min_{\alpha \in \mathbb{R}} \|\alpha x^{(k)} - v^1\|_2. \tag{5.5.3}$$

Forme (5.5.1) äquivalent um zu

$$(\lambda_1^k c_1)^{-1} x^{(k)} = v^1 + c_1^{-1} r^{(k)}$$

und setze mit $\alpha_k := (\lambda_1^k c_1)^{-1}$ in (5.5.3) ein, womit

$$d(S^k, T) \leq \|\alpha_k x^k - v^1\|_2 = |c_1^{-1}| \|r^{(k)}\|_2 = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \tag{5.5.4}$$

folgt. λ_1 kann also folgendermaßen approximiert werden:

$$\lambda^{(k)} = \frac{(x^{(k)})^T A x^{(k)}}{\|x^{(k)}\|_2^2} = \frac{(x^{(k)})^T x^{(k+1)}}{\|x^{(k)}\|_2^2}. \tag{5.5.5}$$

Da $\alpha_{k+1} = (\lambda_1^{k+1} c_1)^{-1} = (\lambda_1^k c_1)^{-1} (\lambda_1)^{-1} = \frac{\alpha_k}{\lambda_1}$, läßt sich (5.5.5) schreiben als

$$\begin{aligned}
\lambda^{(k)} &= \lambda_1 \frac{(\alpha_k x^{(k)})^T (\alpha_{k+1} x^{(k+1)})}{\|\alpha_k x^{(k)}\|_2^2} \\
&= \lambda_1 \frac{(v^1 + \mathcal{O}(|\frac{\lambda_2}{\lambda_1}|)^k)^T (v^1 + \mathcal{O}(|\frac{\lambda_2}{\lambda_1}|)^{k+1})}{\|v^1 + \mathcal{O}(|\frac{\lambda_2}{\lambda_1}|)^k\|_2^2} \\
&= \lambda_1 \frac{1 + \mathcal{O}(|\frac{\lambda_2}{\lambda_1}|^k)}{1 + \mathcal{O}(|\frac{\lambda_2}{\lambda_1}|^k)} \\
&= \lambda_1 \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right),
\end{aligned}$$

womit wir

$$\left| \lambda^{(k)} - \lambda_1 \right| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \tag{5.5.6}$$

erreicht haben.

Bemerkung 5.5.7. Falls A symmetrisch ist, gilt für die obige Konvergenz sogar

$$\left| \lambda^{(k)} - \lambda_1 \right| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^{2k} \right).$$

■

Bevor wir nun einen Algorithmus zur Vektoriteration angeben, wollen wir uns noch ein paar Gedanken über die *Skalierung der Iterierten* machen. Da nämlich $\|x^{(k)}\|_2 \to \infty$, falls $|\lambda_1| > 1$ und $\|x^{(k)}\|_2 \to 0$, falls $|\lambda_1| < 1$, ist es zweckmäßig, die Iterierten $x^{(k)}$ zu skalieren. Damit werden dann starke änderungen der Größenordnungen vermieden. Da der Unterraum S^k und die Iterierten $x^{(k)}$ in (5.5.5) nicht von der Skalierung von $x^{(k)}$ abhängen, bleiben die Größenordnungen der Konvergenzen in (5.5.4) und (5.5.6) erhalten. Nun haben wir genügend Mittel zusammen, um einen Algorithmus erstellen zu können.

Algorithmus 5.5.8 (Vektoriteration, Potenzmethode)**.** Wähle einen Startvektor $y^{(0)} \neq 0$ mit $\|y^{(0)}\|_2 = 1$. Für $k = 0, 1, \ldots$ berechne

$$\begin{aligned} \tilde{y}^{(k+1)} &= Ay^{(k)} \\ \lambda^{(k)} &= (\tilde{y}^{(k+1)})^T y^{(k)} \\ y^{(k+1)} &= \frac{\tilde{y}^{(k+1)}}{\|\tilde{y}^{(k+1)}\|_2} \end{aligned}$$

■

Hierzu sei noch bemerkt, daß unter der Annahme $y^{(0)} = x^{(0)}$ mit vollständiger Induktion nach k folgt, daß

$$y^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|_2} = \frac{A^k x^{(0)}}{\|A^k x^{(0)}\|_2}$$

gilt. Also liefert Algorithmus 5.5.8 bis auf einen Skalierungsfaktor in $x^{(k)}$ die oben analysierten Folgen $x^{(k)}$ und $\lambda^{(k)}$. Weiter ist zu beachten, daß die Konvergenzgeschwindigkeit in Algorithmus 5.5.8 wesentlich vom Verhältnis zwischen λ_2 und λ_1 abhängt. Hierzu vergleiche die Gleichungen (5.5.4), (5.5.6) und (5.5.7). Wir schließen diesen Abschnitt mit einem Beispiel ab.

Beispiel 5.5.9. Betrachte das Eigenwertproblem 5.1.2 (Sturm–Liouville–Problem). Durch Diskretisierung hatten wir dies in die Form

$$Au = \lambda Ru$$

mit $A \in \mathbb{R}^{(n-1)\times(n-1)}$ überführt. Sei $r(x) = 1$, also $R = I$, womit die Eigenwerte von A explizit bekannt sind:

$$\lambda_{n-j} = \frac{4}{h^2} \sin^2\left(\frac{1}{2} j\pi h\right)$$

mit $j = 1, \ldots, n-1$ und $h = \frac{1}{n}$. O.B.d.A. sei die Numerierung hier so gewählt, daß $\lambda_1 > \lambda_2 > \ldots > \lambda_{n-1}$ gilt. Betrachte nun

$$\begin{aligned} \left|\frac{\lambda_2}{\lambda_1}\right| &= \frac{\lambda_2}{\lambda_1} = \frac{\sin^2\left(\frac{1}{2}(n-2)\pi h\right)}{\sin^2\left(\frac{1}{2}(n-1)\pi h\right)} = \frac{\sin^2\left(\frac{1}{2}\pi - \pi h\right)}{\sin^2\left(\frac{1}{2}\pi - \frac{1}{2}\pi h\right)} \\ &= \frac{\cos^2(\pi h)}{\cos^2\left(\frac{1}{2}\pi h\right)} \doteq \frac{(1 - \frac{1}{2}(\pi h)^2)^2}{(1 - \frac{1}{2}(\frac{1}{2}\pi h)^2)^2} \\ &\doteq \frac{1 - \pi^2 h^2 + \frac{1}{4}\pi^4 h^4}{1 - \frac{1}{4}\pi^2 h^2 + \frac{1}{64}\pi^2 h^2} \approx 1 - \pi^2 h^2 \end{aligned}$$

Hieran ist ersichtlich, daß man für $h \ll 1$ mit einer sehr langsamen Konvergenz $\lambda^{(k)} \to \lambda_1$ mit einem Faktor $\approx \left|\frac{\lambda_2}{\lambda_1}\right|^2 \doteq 1 - \pi^2 h^2$ pro Iteration rechnen muß. Die Resultate für $h = \frac{1}{30}$ mit dem Startvektor $x^{(0)} = (1, 2, \ldots, 29)^T$ und $y^{(0)} = \frac{x^{(0)}}{\|x^{(0)}\|_2}$ ergeben:

k	$\lvert\lambda^{(k)} - \lambda_1\rvert$	$\frac{\lvert\lambda^{(k)}-\lambda_1\rvert}{\lvert\lambda^{(k-1)}-\lambda_1\rvert}$
1	1.8e+3	0.51
5	4.8e+2	0.82
15	1.6e+2	0.93
50	4.4e+1	0.98
100	1.7e+1	0.98
150	8.2	0.99

Als Fazit können wir also festhalten, daß Algorithmus 5.5.8 konvergiert, falls A diagonalisierbar und λ_1 ein einfacher Eigenwert ist. Aber die Konvergenz kann trotzdem sehr langsam sein, was wir auch in mehreren Fällen gesehen haben. Im folgenden Abschnitt betrachten wir jetzt eine Variante der Vektoriteration, die inverse Vektoriteration.

■

5.6 Inverse Vektoriteration

Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär und diagonalisierbar. Die Eigenwertgleichung $Av^i = \lambda_i v^i$ für $i = 1, \ldots, n$ ist äquivalent zu

$$\frac{1}{\lambda_i} v^i = A^{-1} v^i.$$

Also würde die Vektoriteration angewandt auf A^{-1} unter der Annahme

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_{n-1}| > |\lambda_n|$$

den betragsmäßig kleinsten Eigenwert λ_n von A ermitteln. Erinnern wir uns noch an die Eigenschaft 5.4.1(iii), so ist λ_i Eigenwert von A genau dann, wenn $\lambda_i - \mu$ Eigenwert von $A - \mu I$ ist. Dies kann man nutzen: Angenommen, man hätte eine Schätzung $\mu \approx \lambda_i$ eines beliebigen Eigenwertes λ_i von A, sodaß

$$|\mu - \lambda_i| < |\mu - \lambda_j| \tag{5.6.1}$$

für alle $i \neq j$ ist (implizit nehmen wir natürlich ebenfalls an, daß λ_i einfacher reeller Eigenwert ist). Dann ist $|\mu - \lambda_i|^{-1}$ der betragsmäßig *größte* Eigenwert von $(A - \mu I)^{-1}$. Zur Berechnung von $(\mu - \lambda_i)^{-1}$ und damit von λ_i wende man die Vektoriteration 5.5.8 auf $(A - \mu I)^{-1}$ wie folgt an:

Algorithmus 5.6.2 (Inverse Vektoriteration mit Spektralverschiebung)**.** Wähle zuerst einen Startvektor $y^{(0)}$ mit $\|y^{(0)}\|_2 = 1$. Sei μ so, daß Gleichung (5.6.1) gilt. Für $k = 0, 1, 2, \ldots$ berechne

$$\begin{aligned} (A - \mu I)\tilde{y}^{(k+1)} &= y^{(k)} && (5.6.3)\\ \lambda^{(k)} &= \frac{1}{(\tilde{y}^{(k+1)})^T y^{(k)}} + \mu && (5.6.4)\\ y^{(k+1)} &= \frac{\tilde{y}^{(k+1)}}{\|\tilde{y}^{(k+1)}\|_2}. \end{aligned}$$

■

Hierbei ist noch zu beachten, daß in (5.6.3) $\tilde{y}^{(k+1)} = (A - \mu I)^{-1} y^k$ als Lösung des Systems $(A - \mu I)\tilde{y}^{(k+1)} = y^{(k)}$ bestimmt wird. Um dies zu erreichen, berechnet man einmal eine QR– oder LR–Zerlegung von $A - \mu I$. Für $\mu = 0$ wird natürlich der kleinste Eigenwert bestimmt. Bei der Vektoriteration aus 5.5.8 angewandt auf $(A - \mu I)^{-1}$ strebt $(\tilde{y}^{(k+1)})^T y^{(k)}$ gegen den betragsmäßig größten Eigenwert von $(A - \mu I)^{-1}$, also gegen $\frac{1}{|\lambda_i - \mu|}$ (vergleiche (5.6.1)), d.h.

$$\lambda^{(k)} := \frac{1}{(\tilde{y}^{(k+1)})^T y^{(k)}} + \mu \to \lambda_i \tag{5.6.5}$$

für $k \to \infty$. In Algorithmus 5.5.8 hängt die Konvergenzgeschwindigkeit von $\frac{\lambda_2}{\lambda_1}$ ab. In hiesigem Algorithmus hängt die Konvergenzgeschwindigkeit von

$$\frac{\max\limits_{j\neq i} \frac{1}{|\lambda_i - \mu|}}{\frac{1}{|\lambda_i - \mu|}} = \frac{\frac{1}{\min\limits_{j\neq i} |\lambda_j - \mu|}}{\frac{1}{|\lambda_i - \mu|}} = \frac{|\lambda_i - \mu|}{\min\limits_{j\neq i} |\lambda_j - \mu|} \tag{5.6.6}$$

ab. Ist μ also eine gute Schätzung von λ_i, so gilt

$$\frac{|\lambda_i - \mu|}{\min\limits_{j \neq i} |\lambda_j - \mu|} \ll 1,$$

was bedeutet, daß das Verfahren sehr schnell konvergiert. Als Fazit läßt sich also festhalten, daß man durch eine geeignete Wahl der Spektralverschiebung μ mit der inversen Vektoriteration aus Algorithmus 5.6.2 *einzelne* Eigenwerte und Eigenvektoren von A berechnen kann. Praktisch ist es allerdings in vielen Fällen nicht ohne weiteres klar, wie man μ zu wählen hat.

Eine weitere wichtige Bemerkung ist, daß die Konvergenzgeschwindigkeit noch erheblich verbessert werden kann, wenn man μ nach jedem Schritt auf die aktuellste Approximation von $\lambda^{(k)}$ setzt. Dann muß allerdings die QR– oder LR–Zerlegung von $A - \lambda^{(k)} I$ in jedem Schritt neu berechnet werden, was den Rechenaufwand natürlich stark ansteigen läßt. Eine weitere Frage, auf die wir bisher weder in 5.5 noch hier eingegangen sind, ist, nach wie vielen Iterationen man die (inverse) Vektoriteration abbrechen sollte. Hierfür läßt sich allerdings kein allgemeingültiges Kriterium angeben, sondern über diesen *Abbruchparameter* muß bei jedem Problem separat entschieden werden. Weitere interessante Details zur Vektoriteration findet man in [H] und [GL]. Vor allem in letztgenanntem Werk befinden sich viele gerechnete Beispiele. Wir werden im nächsten Unterkapitel den wichtigsten — auf der QR–Zerlegung basierenden — Algorithmus zur Bestimmung von Eigenwerten von Eigenvektoren kennenlernen.

5.7 QR–Verfahren zur Berechnung von Eigenwerten und Eigenvektoren

In Satz 5.3.5 haben wir gesehen, daß das Eigenwertproblem für symmetrische Matrizen immer gut konditioniert ist. Deshalb wollen wir jetzt einen effektiven Algorithmus zur *gleichzeitigen* Berechnung aller Eigenwerte einer symmetrischen Matrix $A \in \mathbb{R}^{n \times n}$ herleiten. Aus den früheren Kapiteln wissen wir schon, daß es im symmetrischen Fall nur reelle Eigenwerte gibt, und eine Orthonormalbasis aus Eigenvektoren $v^1, \ldots, v^n$ des $\mathbb{R}^n$ existiert, womit wir A in eine Diagonalmatrix transformieren können:

$$Q^T A Q = \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \tag{5.7.1}$$

wobei $Q = [v^1, \ldots, v^n]$ ist. Um nun praktisch diese Diagonalgestalt zu erreichen, gibt es mehrere potentielle Ansätze. Zum einen könnte man versuchen, Q direkt in endlich vielen Schritten zu bestimmen, da die Eigenwerte Nullstellen des charakteristischen Polynoms $p(\lambda) = \det(A - \lambda I)$ sind. Damit wäre auch ein endliches Verfahren zur Bestimmung der Nullstellen eines Polynoms n-ten Grades von Nöten. Ein solches basierend auf den elementaren Rechenoperationen (mit $\sqrt{\cdot}$) kann es aber nach dem Satz von Abel nicht geben. Also ist diese erste Möglichkeit ungeeignet. Eine weitere Möglichkeit ist, A mittels ähnlichkeitstransformationen, z.B. Orthogonaltransformationen auf Diagonalgestalt zu bringen. Z.B. könnten hier die Householder–Transformationen aus (3.7.5) angewandt werden, da die Eigenwerte unter ähnlichkeitstransformationen invariant sind. Dies könnte folgendermaßen aussehen:

$$A = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} \xrightarrow{Q_1 \cdot} \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix} \xrightarrow{\cdot Q_2} \begin{pmatrix} * & 0 & 0 & 0 & 0 \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$$

Wenn wir also A zunächst mit einer orthogonalen Matrix Q_1 von links multiplizieren, um die erste Householder–Transformation auszuführen, wird die erste Spalte unter dem ersten Eintrag wie

gewünscht zu Null. Wenden wir das Verfahren nun auf die erste Zeile an, indem wir von rechts mit einer HOUSEHOLDER–Matrix Q_2 multiplizieren, um auch dort wieder Nullen ab dem zweiten Eintrag zu erzeugen, läuft die erste Spalte aber wieder voll, und die Matrix Q_1AQ_2 ist nicht mehr symmetrisch. Also ist dieses Verfahren so ungeeignet. Ganz anders ist der Fall, wenn wir A auf *Tridiagonalgestalt* transformieren wollten. Dann könnten wir das Verfahren auf kleiner werdende Matrizen Q_i anwenden, also

$$A = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} \xrightarrow{Q_1 \cdot} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix} \xrightarrow{\cdot Q_1^T} \begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix} \xrightarrow{\cdots}$$

bis schließlich eine Matrix der Form

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

entsteht. Dieses erste Ergebnis halten wir fest.

Lemma 5.7.2. Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch. Dann existiert eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$, die das Produkt von $n-2$ HOUSEHOLDER–Reflexionen ist, sodaß QAQ^T eine Tridiagonalmatrix ist. Insbesondere ist QAQ^T symmetrisch.

Beweis: Eine Iteration des obigen Prozesses und die Eigenschaften der HOUSEHOLDER–Matrizen $Q_1, \ldots, Q_{n-2}$ liefern

$$Q_{n-2} \cdots Q_1 A Q_1^T \cdots Q_{n-2}^T = \begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}.$$

Setze nun $Q := Q_{n-2} \cdots Q_1$. ∎

Hiermit haben wir unser Problem bereits auf die Berechnung der Eigenwerte einer *symmetrischen Tridiagonalmatrix* reduziert.

Berechnung der Eigenwerte einer symmetrischen Tridiagonalmatrix

Die nun folgende Idee stammt von RUTISHAUSER aus dem Jahre 1958. Er berechnete die LR–Zerlegung der Matrix

$$B = \begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

und erhielt entsprechend die zwei *Bidiagonalmatrizen*

$$B = LR = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ * & 1 & 0 & 0 & 0 \\ 0 & * & 1 & 0 & 0 \\ 0 & 0 & * & 1 & 0 \\ 0 & 0 & 0 & * & 1 \end{pmatrix} \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix}.$$

Nun vertauschte er L und R und bekam

$$\tilde{B} := RL = \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ * & 1 & 0 & 0 & 0 \\ 0 & * & 1 & 0 & 0 \\ 0 & 0 & * & 1 & 0 \\ 0 & 0 & 0 & * & 1 \end{pmatrix}.$$

Nun wiederholte er dieses Verfahren mit $\tilde{B}$. In den Jahren 1959 und 1961 wurde dieses Verfahren basierend auf QR–Zerlegung von Francis und Kublanovskaja modifiziert, da diese numerisch stabiler ist. Nun können wir hiermit eine Folge von Matrizen $\{B_k\}_{k\in\mathbb{N}}$ definieren und erhalten einen ersten Algorithmus.

Algorithmus 5.7.3. Setze $B_1 := B$. Für $k = 1, 2, \ldots$

i) bilde QR–Zerlegung von $B_k = Q_k R_k$ mit orthogonalem Q_k und oberer Dreiecksmatrix R_k.

ii) setze $B_{k+1} := R_k Q_k$.

■

Um ein paar zusätzliche Informationen über die nun definierten Matrizen zu erhalten, geben wir folgendes

Lemma 5.7.4. Die durch Algorithmus 5.7.3 definierten Matrizen B_k haben folgende Eigenschaften:

(a) B_k ist ähnlich zu B.

(b) Falls B symmetrisch ist, ist dies auch B_k.

(c) Falls B symmetrisch und tridiagonal ist, so hat auch B_k diese Eigenschaften.

Beweis: Zu (a) zeigen wir, daß B_k ähnlich zu B_{k+1} ist. Dies folgt aber direkt durch folgende Gleichung:

$$Q_k B_{k+1} Q_k^T = Q_k R_k Q_k Q_k^T = Q_k R_k = B_k.$$

Behauptung (b) zeigen wir durch vollständige Induktion nach k. Der Induktionsanfang ist trivial. Zum Induktionsschritt $k \to k+1$ betrachte

$$\begin{aligned} B_{k+1}^T &= B_{k+1}^T Q_k^T Q_k = (Q_k B_{k+1})^T Q_k = (Q_k R_k Q_k)^T Q_k \\ &= (B_k Q_k)^T Q_k = Q_k^T B_k^T Q_k = Q_k^T B_k Q_k \end{aligned}$$

wobei im letzten Schritt die Induktionsvoraussetzung angewandt wird. Hierbei sieht man, daß die Symmetrie durch ähnlichkeitstransformationen erhalten wird. Den letzten Teil (c) zeigen wir

ebenfalls über Induktion nach k mit GIVENS–Rotationen. Auch hier ist der Induktionsanfang klar. Sei nun B_k tridiagonal und symmetrisch. Konstruiere Q_k so, daß $R_k = Q_k^T B_k$ ist. Also entsteht aus der Matrix B_k mit der von oben bekannten Struktur eine Matrix $R_k = Q_k^T B_k := G_{n-1,n} \cdots G_{1,2} B_k$ mit der Struktur

$$\begin{pmatrix} * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix}.$$

Diese Struktur kommt dadurch zustande, daß man durch die verwendeten GIVENS–Rotationen $G_{1,2}, G_{2,3}, \ldots, G_{n-1,n}$, um die untere Nebendiagonale $*$ zu eliminieren, aufgrund des Nullenmusters ein *fill–in* in der 2. Nebendiagonale erzeugt. Insgesamt haben wir aber $B_{k+1} = R_k Q_k = Q_k^T B_k Q_k$. Da nach (b) aber B_{k+1} symmetrisch ist, muß die durch das fill–in entstandene Nebendiagonale verschwinden und B_{k+1} hat Tridiagonalgestalt. ∎

Nun kommt der alles entscheidende Satz dieses Kapitels, aus dem auch nachher der für uns wichtigste Algorithmus zur Berechnung der Eigenwerte folgen wird.

Satz 5.7.5. Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch mit den Eigenwerten $\lambda_1, \ldots, \lambda_n$, die die Eigenschaft

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0 \tag{5.7.6}$$

haben mögen. Weiter seien die Matrizenfolgen $\{A_k\}_{k\in\mathbb{N}}$, $\{R_k\}_{k\in\mathbb{N}}$ und $\{Q_k\}_{k\in\mathbb{N}}$ wie in Algorithmus 5.7.3 definiert: $A_1 := A$, $A_k = Q_k R_k$, $A_{k+1} := R_k Q_k$. Dann gibt es Matrizen $S_k = \operatorname{diag}(\sigma_1^{(k)}, \ldots, \sigma_n^{(k)})$ mit $|\sigma_i^{(k)}| = 1$, sodaß

$$\lim_{k\to\infty} S_{k-1} Q_k S_k = I \tag{5.7.7}$$

und

$$\lim_{k\to\infty} S_k R_k S_{k-1} = \lim_{k\to\infty} S_{k-1} A_k S_k = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) =: D. \tag{5.7.8}$$

Insbesondere gilt

$$\lim_{k\to\infty} a_{jj}^{(k)} = \lambda_j \tag{5.7.9}$$

für $j = 1, \ldots, n$ wobei a_{jj} das j-te Diagonalelement der Matrix$A_k = (a_{ij}^{(k)})_{ij}$ ist. ∎

Bevor wir dies beweisen, noch ein paar Bemerkungen zu diesem Verfahren:

(i) Mit der Aussage von Satz 5.7.5 bietet Algorithmus 5.7.3 die Konstruktion einer SCHUR–Zerlegung gemäß Korollar 5.2.9 von A.

(ii) Algorithmus 5.7.3 läßt sich als Verallgemeinerung der Vektoriteration betrachten, denn die Iteration entspricht der Projektion auf die Unterräume, die von den Spalten von A_k aufgespannt werden. Näheres hierzu findet man in [SB], p. 58ff..

(iii) Man kann Satz 5.7.5 auch allgemeiner für nicht symmetrische Matrizen A formulieren. Dann muß man als zusätzliche Voraussetzung fordern, daß $A = Y^{-1} D Y$ mit $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$

die JORDAN–Normalform von A sei, und Y in diesem Fall eine LR–Zerlegung besitzt. Unter diesen Voraussetzungen wandelt sich die Aussage (5.7.8) allerdings in

$$\lim_{k\to\infty} S_k R_k S_{k-1} = \lim_{k\to\infty} S_{k-1} A_k S_k = \begin{pmatrix} \lambda_1 & & & \\ & \ddots & * & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \tag{5.7.8a}$$

und es entsteht eine komplexe SCHUR'sche Normalform.

(iv) Falls $A \in \mathbb{C}^{n\times n}$ müssen (5.7.7) und (5.7.8) ersetzt werden durch

$$\lim_{k\to\infty} S_{k-1}^* Q_k S_k = I \tag{5.7.7a}$$

und

$$\lim_{k\to\infty} S_k^* R_k S_{k-1} = \lim_{k\to\infty} S_{k-1}^* A_k S_k = \begin{pmatrix} \lambda_1 & & & \\ & \ddots & * & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \tag{5.7.8b}$$

Beweis von Satz 5.7.5 (nach [W]) Zunächst konstruieren wir eine geschickte QR–Zerlegung von A^k. Da A symmetrisch ist, existiert eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ sodaß $Q^{-1}AQ = Q^TAQ = D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Hiermit folgt

$$A^k = Q^T D^k Q \tag{5.7.10}$$

wobei $D^k = \operatorname{diag}(\lambda_1^k, \ldots, \lambda_n^k)$ ist. Sei nun $Q = LR$ die LR–Zerlegung (ohne Pivotisierung)[1] von Q. Damit erhält (5.7.10) die Form

$$A^k = Q^T D^k LR = Q^T (D^k L D^{-k})(D^k R). \tag{5.7.11}$$

Für den Term $D^k L D^{-k}$ gilt in Komponenten:

$$(D^k L D^{-k})_{ij} = l_{ij} \left(\frac{\lambda_i}{\lambda_j}\right)^k.$$

Damit folgt aufgrund der Struktur von L als unterer Dreiecksmatrix mit Einsen auf der Diagonale:

$$D^k L D^{-k} = I + E_k \tag{5.7.12}$$

mit $\lim_{k\to\infty} E_k = 0$. Dies in (5.7.11) eingesetzt, ergibt

$$A^k = Q^T (I + E_k)(D^k R). \tag{5.7.11a}$$

Nun betrachten wir den Term

$$(I + E_k)^T (I + E_k) =: I + F_k \tag{5.7.13}$$

[1] Diese existiert, wenn alle Hauptminoren von $Q \neq 0$ sind. Ist dies nicht der Fall, so ist die LR-Zerlegung, da Q insbesondere als invertierbar angenommen ist, mit einer Pivotisierung in der Form

$$\tilde{P}Q = LR =: \tilde{Q}$$

stets erreichbar. Der Beweis läuft mit dieser Modifikation dann analog, mit dem Unterschied, daß sich die Größenanordnung der Eigenwerte ändert.

mit $F_k = E_k^T + E_k + E_k^T E_k$. Da nun (5.7.13) symmetrisch positiv definit ist, existiert eine CHOLESKY–Zerlegung gemäß (3.4.4). Sei diese durch

$$I + F_k = \tilde{R}_k^T \tilde{R}_k \tag{5.7.14}$$

mit $(\tilde{R}_k)_{ii} > 0$ und $\tilde{R}_k$ als oberer Dreiecksmatrix bezeichnet. Man kann zeigen, daß die Matrix $\tilde{R}_k$ stetig von $I + F_k$ abhängt. Also folgt aus $\lim\limits_{k\to\infty} F_k = 0$, daß $\lim\limits_{k\to\infty} \tilde{R}_k = I$ sein muß. Weiterhin ist $\tilde{Q}_k := (I + E_k)\tilde{R}_k^{-1}$ orthogonal, denn

$$\begin{aligned} \tilde{Q}_k^T \tilde{Q}_k &= \tilde{R}_k^{-T}(I + E_k)^T(I + E_k)\tilde{R}_k^{-1} = \tilde{R}_k^{-T}(I + F_k)\tilde{R}_k^{-1} \\ &= \tilde{R}_k^{-T}\tilde{R}_k^T\tilde{R}_k\tilde{R}_k^{-1} = I. \end{aligned}$$

Damit besitzt $I + E_k$ die QR–Zerlegung $I + E_k = \tilde{Q}_k\tilde{R}_k$, und für die Grenzwerte gilt

$$\lim_{k\to\infty} \tilde{Q}_k = \lim_{k\to\infty} (I + E_k)\tilde{R}_k^{-1} = I \qquad \text{und} \qquad \lim_{k\to\infty} \tilde{R}_k = I. \tag{5.7.15}$$

Also folgt für A^k in der Darstellung (5.7.11a)

$$A^k = Q^T(\tilde{Q}_k\tilde{R}_k)D^k R = (Q^T\tilde{Q}_k)(\tilde{R}_k D^k R), \tag{5.7.16}$$

womit wir den ersten Teil des Beweises abschließen. Der zweite Teil beginnt nun damit, daß wir per vollständiger Induktion nach k zeigen, daß A^k die Darstellung

$$A^k = Q_1 \cdots Q_k R_k \cdots R_1 \tag{5.7.17}$$

besitzt. Hierzu benutzen wir das Verfahren aus Algorithmus 5.7.3, das wir bisher noch nicht verwendet haben. Setze vorher noch $P_k := Q_1 \cdots Q_k$ und $U_k := R_k \cdots R_1$. Der Induktionsanfang ist klar, denn nach Definition gilt $A = A_1 = Q_1 R_1$. Nun zu $k \to k + 1$. Es gilt nach besagtem Algorithmus

$$A_{k+1} = R_k Q_k = Q_k^T A_k Q_k = \ldots = Q_k^T \cdots Q_1^T A Q_1 \cdots Q_k = P_k^{-1} A P_k.$$

Dies ist äquivalent zu

$$P_k A_{k+1} = A P_k. \tag{5.7.18}$$

Damit gilt für A^{k+1}

$$\begin{aligned} A^{k+1} &= AA^k = (AP_k)U_k = P_k A_{k+1} U_k \\ &= P_k(Q_{k+1}R_{k+1})U_k = P_{k+1}U_{k+1} \end{aligned}$$

womit A^k also die Darstellung (5.7.17) besitzt. Da aber P_k orthogonal und U_k eine obere Dreiecksmatrix ist, läßt sich die QR–Zerlegung (5.7.17) von A^k durch die QR–Zerlegung der einzelnen Matrizen $A_1, \ldots, A_k$ ausdrücken. Das bedeutet, daß wir nun (5.7.16) und (5.7.17) zusammenfassen. Hierzu beachten wir, daß die QR–Zerlegung in (5.7.17) eindeutig bis auf Reskalierung der Spalten von Q ist. Das bedeutet, daß es eine Vorzeichenmatrix

$$S_k = \operatorname{diag}(\sigma_1^{(k)}, \ldots, \sigma_n^{(k)})$$

mit $|\sigma_i(k)| = 1$ und den Eigenschaften

$$P_k = (Q^T\tilde{Q}_k)S_k^* \qquad \text{und} \qquad U_k = S_k\tilde{R}_k D^k R \tag{5.7.19}$$

gibt. Damit folgt aber

$$\lim_{k\to\infty} P_k S_k = \lim_{k\to\infty} Q^T\tilde{Q}_k = Q^T(\lim_{k\to\infty} \tilde{Q}_k) = Q^T$$

und

$$\begin{aligned} Q_k = P_{k+1}^{-1} P_k &= S_{k-1}\tilde{Q}_{k-1}^{-1}(Q^T)^{-1}Q^T\tilde{Q}_k S_k^* \\ &= S_{k-1}\tilde{Q}_{k-1}^{-1}\tilde{Q}_k S_k^*. \end{aligned}$$

Mit diesen Resultaten können wir nun an die Limites in Satz 5.7.5 herangehen. Es gilt nun offenbar

$$\lim_{k\to\infty} S_{k-1}^* Q_k S_k = I,$$

womit wir (5.7.7) bereits gezeigt haben. Für die beiden anderen Grenzwerte müssen wir noch ein wenig rechnen. Für R_k gilt

$$\begin{aligned} R_k = U_k U_{k-1}^{-1} &= (S_k\tilde{R}_k D^k R)(R^{-1}D^{-(k-1)}\tilde{R}_{k-1}^{-1}S_{k-1}^*) \\ &= S_k\tilde{R}_k D\tilde{R}_{k-1}^{-1}S_{k-1}^*. \end{aligned}$$

Hieraus folgt

$$S_k^* R_k S_{k-1} = \tilde{R}_k D\tilde{R}_{k-1}^{-1},$$

womit nun (5.7.8)

$$\lim_{k\to\infty} S_k^* R_k S_{k-1} = D$$

bewiesen ist. Schließlich folgt mit $A_k = Q_k R_k$, daß

$$S_{k-1}^* A_k S_k = (S_{k-1}^* Q_k S_k)(S_k^* R_k S_k)$$

und damit dann

$$\lim_{k\to\infty}(S_{k-1}^* Q_k S_k)(S_k^* R_k S_k) = ID = D,$$

was die Aussage von (5.7.9) ist, und unseren Beweis beendet. ∎

Nun sind wir soweit, um die praktische Durchführung in Angriff zu nehmen.

Algorithmus 5.7.20 (QR–Verfahren zur Eigenwertberechnung)**.** Berechne für symmetrisches $A \in \mathbb{R}^{n\times n}$

I.) Reduziere A mittels HOUSEHOLDER–Reflexionen auf Tridiagonalgestalt $B = Q^T AQ$ wie in Lemma 5.7.2.

II.) Wende auf B Algorithmus 5.7.3 mit GIVENS–Rotationen an, womit

$$GBG^T \approx D$$

und G das Produkt aller GIVENS–Matrizen ist. Die Spalten von GQ^T approximieren die Eigenvektoren von A.

∎

Der Aufwand für diesen Algorithmus beträgt $\mathcal{O}(\frac{4}{3}n^3)$ für Teil I und $\mathcal{O}(n^2)$ für Teil II.

Bemerkung 5.7.21.

- Falls A nichtsymmetrisch ist, transformiere A auf obere HESSENBERG–Gestalt

$$B = Q^T AQ = \begin{pmatrix} * & & & & \\ * & \ddots & & * & \\ & \ddots & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & * & * \end{pmatrix}.$$

  Danach bringt man die Matrix in II auf SCHUR'sche Normalform. Weiteres findet sich in [SB], p. 41ff..

- Wenn man eine Singulärwertzerlegung wünscht, kann man den Algorithmus entsprechend modifizieren.
- Man kann zeigen, daß die Konvergenzgeschwindigkeit der QR–Verfahrens von den Faktoren

$$\left|\frac{\lambda_{j+1}}{\lambda_j}\right|$$

für $j = 1, \ldots, n-1$ abhängt. Das bedeutet, daß im Falle

$$\left|\frac{\lambda_{j+1}}{\lambda_j}\right| \approx 1$$

für ein oder mehrere j die Effizienz sehr schlecht wird. Hier schafft dann ein QR–Algorithmus mit Spektralverschiebung anstelle von II Abhilfe (vergleiche dazu Algorithmus 5.6.2).

■

Nachdem wir nun alle im Rahmen dieser Vorlesung zur Berechnung der Eigenwerte relevanten Verfahren besprochen haben, kommen wir im nächsten Abschnitt zu der bereits in Abschnitt 4.4 erwähnten Singulärwertzerlegung.

5.8 Singulärwertzerlegung

In Satz 4.4.1 haben wir gezeigt, daß es zu jeder Matrix $A \in \mathbb{R}^{m\times n}$ orthogonale Matrizen $U \in \mathbb{O}_m$ und $V \in \mathbb{O}_n$ mit

$$U^T AV = \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p, 0, \ldots, 0)$$

mit $p = \operatorname{rang}(A) \leq \min(m, n)$ (*Singulärwertzerlegung*), und die Zahlen $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$ waren die *Singulärwerte*. Man verwendet die Singulärwertzerlegung häufig zur „Dimensionsreduktion“ in dem Sinne, daß man im Falle $p = \operatorname{rang}(A) \ll \min(m, n)$ die Matrix A weiterverarbeitet als $U\Sigma V^T$, wobei die wesentlichen Informationen über A in Σ enthalten sind. Ein Beispiel hierfür sind die nach Hackbusch benannten H–Matrizen, die bei der Diskretisierung von Integralgleichungen eine wichtige Rolle spielen. Nun aber zur *numerischen Berechnung der Singulärwerte*: Wir haben bereits in 4.4 gesehen, daß für die Singulärwerte $\sigma_i = \sigma_i(A) = \sqrt{\lambda_i(A^T A)}$ gilt, wobei $\lambda_i(A^T A)$ die Eigenwerte von $A^T A$ waren. Daher wäre es prinzipiell möglich, $A^T A$ zu berechnen und dann Algorithmus 5.7.20 darauf anzuwenden. Ist $A^T A$ symmetrisch, so ist das Eigenwertproblem auch gut konditioniert. Aber A kann schlecht konditioniert sein, sodaß sich die Konditionszahl bei der Bildung von $A^T A$ quadriert. Daher ist diese Methode im allgemeinen nicht brauchbar. Also müssen wir nach einer Alternative suchen, die direkt über A geht. Man beachte, daß die Singulärwerte bei Multiplikation von A mit orthogonalen Matrizen invariant bleiben, d.h. die Matrizen A und PAQ mit $P \in \mathbb{O}_m$ und $Q \in \mathbb{O}_n$ besitzen dieselben Singulärwerte. Um nun die Singulärwertzerlegung zu berechnen, bringen wir A zunächst mit Householder–Transformationen auf obere Bidiagonalgestalt, sodaß $B^T B$ tridiagonal ist. Dies halten wir fest.

Lemma 5.8.1. Seien $m \geq n$ und $A \in \mathbb{R}^{m\times n}$ beliebig. Dann gibt es orthogonale Matrizen $P \in \mathbb{O}_m$ und $Q \in \mathbb{O}_n$, sodaß

$$PAQ = \begin{pmatrix} B \\ O \end{pmatrix}$$

und $B \in \mathbb{R}^{n\times n}$ eine obere Bidiagonalmatrix

$$\begin{pmatrix} * & * & 0 \\ & \ddots & * \\ 0 & & * \end{pmatrix}$$

ist.

Beweis: Wir führen den Beweis mit HOUSEHOLDER–Reflexionen:

$$A = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} \xrightarrow{P_1\cdot} \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix} \xrightarrow{\cdot Q_1} \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix}$$

$$\xrightarrow{P_2\cdot} \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{pmatrix} \xrightarrow{\cdots} \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} B \\ 0 \end{pmatrix}$$

Also ist $\begin{pmatrix} B \\ 0 \end{pmatrix} = P_{m-1} \cdots P_1 A Q_1 \cdots Q_{n-2}$. Setze noch $P := P_{m-1} \cdots P_1$ und $Q := Q_1 \cdots Q_{n-2}$. ∎

Nun müssen wir die Singulärwerte einer oberen Bidiagonalmatrix B berechnen. Dazu wenden wir GIVENS–Rotationen von links und rechts auf $B^T B$ an (vergleiche dazu den Beweis von Lemma 5.7.4(c)). Dies läßt sich wie folgt interpretieren: Wir starten mit

$$B^T B = \begin{pmatrix} * & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \end{pmatrix} \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix} = \begin{pmatrix} * & * & 0 & 0 & 0 \\ b_1 & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}.$$

Wenn wir nun Element $b_1 \neq 0$ mit GIVENS–Rotationen von links zu Null machen, erzeugt dies einen Eintrag $b_2 \neq 0$, sodaß die Matrix B die Gestalt

$$\begin{pmatrix} * & * & b_2 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix}$$

annimmt. Nun machen wir b_2 von rechts zu Null, was uns einen weiteren Eintrag $b_3 \neq 0$ liefert:

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & b_3 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix}.$$

Diesen müssen wir natürlich auch auslöschen, was uns $b_4 \neq 0$ einbringt:

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & b_4 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix}.$$

Dieses Verfahren fährt so fort. Da dieses Vorgehen einem Hinterherjagen nach den durch fill-in erzeugten Elementen gleicht, nennt man dieses auch *chasing*. Wer hierzu mehr Details lesen will, dem sei [GL], p. 452ff. als Lektüre empfohlen. Wenn man dieses Verfahren nun wiederholt, entspricht ein Durchlauf einem Iterationsschritt B_k. Man kann zeigen, daß $\lim_{k\to\infty} B_k = \Sigma$ ist, wobei die Matrix Σ dann die Singulärwerte auf der Diagonalen hat. Hieraus stellen wir den Algorithmus zusammen:

Algorithmus 5.8.2 (QR–Verfahren zur Singulärwertberechnung)**.** Sei $A \in \mathbb{R}^{m\times n}$ beliebig vorgegeben.

(I) Reduziere A mittels HOUSEHOLDER–Reflexionen durch $P \in \mathbb{O}_m$ und $Q \in \mathbb{O}_n$ auf obere Bidiagonalgestalt $PAQ = \begin{pmatrix} B \\ 0 \end{pmatrix}$.

(II) Wende auf B das „chasing“ mit GIVENS–Rotationen an.

Das Ergebnis ist dann $GBG^T \approx \Sigma$. ∎

Der Aufwand für diesen Algorithmus ist $\mathcal{O}(\frac{4}{3}n^3)$ bei (I) und $\mathcal{O}(n^2)$ bei (II).

Damit haben wir Kapitel 5 abgeschlossen. Es gibt noch weitere wichtige Verfahren zur Berechnung von Eigenwerten. Für große symmetrische dünnbesetzte Matrizen gibt es das auf Iterationen beruhende LANCZOS–Verfahren. Dieses behandeln wir später. Wer jetzt schon etwas darüber lesen will, schaue in [H], p. 259ff. nach. Wir werden uns im nächsten Kapitel der iterativen numerischen Lösung von nichtlinearen Gleichungen und Gleichungssystemen zuwenden, um dann etwas später auf iterative Verfahren zur Lösung von linearen Gleichungssystemen zu kommen.

6 Interpolation mit Polynomen

6.1 Vorbemerkungen

In diesem Kapitel wollen wir Probleme der folgenden Art angehen:

Aufgabe 6.1.1. Gegeben seien Stützstellen $x_0, \ldots, x_n \in \mathbb{R}$ und Daten $f_0, \ldots, f_n \in \mathbb{R}$. Sei G_n ein $(n+1)$–dimensionaler Raum stetiger Funktionen. Bestimme ein $g_n \in G_n$ mit der Eigenschaft

$$g_n(x_i) = f_i \tag{6.1.2}$$

für alle $i = 0, \ldots, n$. ∎

Der entscheidende Unterschied zu Kapitel 4 ist die in (6.1.2) geforderte Gleichheit, was die Interpolation von der Approximation trennt. Daher sagt man auch, daß die Funktionswerte in (6.1.2) interpoliert werden. Ein Verfahren hierzu, welches wir im folgenden diskutieren werden, ist die LAGRANGE–Interpolation. Wird diese Interpolation mittels Polynomen durchgeführt, ist also $G_n = \mathcal{P}_n$, so nennt man dies *Polynominterpolation*. Die Existenz und Eindeutigkeit der g_n ist nicht von vornherein klar. Wir werden deshalb im folgenden derartige Sätze beweisen. Interpolationen wurden früher benutzt, um etwa Tafelwerte von Logarithmustafeln zu interpolieren. Dies geschah oftmals mit einer linearen Interpolation, um Zwischenwerte zwischen den Stützstellen zu bestimmen. Heute kommen Interpolationen etwa im *Computer Aided Geometric Design (CAGD)* zur Visualisierung großer Datensätze zur Anwendung, so z.B. beim Entwurf von Karrosserieteilen. Hierbei verwendet man allerdings stückweise Polynome, sogenannte *Splines*, die wir in Kapitel 7 besprechen. Um diese aber bilden zu können, benötigt man Aussagen über Polynominterpolation. Weiter ist die Polynominterpolation oft hilfreich bei der Konstruktion numerischer Verfahren für Integration und partielle Differentialgleichungen. Die in 6.1.1 gestellte Aufgabe muß nicht nur auf Funktionswerte beschränkt sein. Es können ebenfalls Ableitungen als Stützwerte vorgeschrieben werden, was auf die sogenannte HERMITE–Interpolation. führt.

6.2 LAGRANGE– und HERMITE–Interpolationsaufgabe

Funktionswerte und Ableitungen sind Beispiele linearer Funktionale auf einem Funktionenraum $\mathcal{F}$. Es bezeichne

$$\mathcal{F}' = \{\mu \;\mid\; \mu : \mathcal{F} \to \mathbb{R} \;\text{ linear}\}$$

die Menge der *linearen Funktionale* auf $\mathcal{F}$. Mit diesen neuen Begriffen formulieren wir die eingangs gestellte Aufgabe noch einmal neu:

Allgemeines Interpolationsproblem 6.2.1. Es seien die linearen Funktionale $\mu_0, \ldots, \mu_n$ auf G_n, wobei G_n ein $(n+1)$–dimensionaler Teilraum eines Funktionenraums $\mathcal{F}$ sei, vorgegeben. Zu $f \in \mathcal{F}$ finde ein $g_n \in G_n$, sodaß

$$\mu_i(f) = \mu_i(g_n) \tag{6.2.2}$$

für alle $i = 0, \ldots, n$ sei. ∎

Ein Beispiel haben wir in (6.1.2) mit $\mu_i(f) = f(x_i)$ bereits gesehen. Die Vorgabe von Funktionswerten nennt man LAGRANGE–Interpolation. Die Bedingungen für eine HERMITE–Interpolation könnten

$$\begin{aligned} \mu_0(f) &= f(x_0); \quad \mu_1(f) = f'(x_0) \\ \mu_2(f) &= f(x_1); \quad \mu_3(f) = f'(x_1) \end{aligned}$$

lauten. Als nächstes werden wir die Existenz und Eindeutigkeit der Lösung des Problems 6.2.1 zeigen.

Satz 6.2.3. Sei $\{\varphi_0, \ldots, \varphi_n\}$ eine Basis von G_n. Aufgabe 6.2.1 besitzt genau dann eine eindeutige Lösung, wenn

$$\det(\mu_i(\varphi_j))_{i,j=0,\ldots,n} \neq 0 \tag{6.2.4}$$

ist.

Beweis: Sei $\{\varphi_j\}_{j=0,\ldots,n}$ eine Basis von G_n. Dann läßt sich jedes $g_n \in G_n$ als $g_n = \sum_{j=0}^{n} \alpha_j \varphi_j$ darstellen. Damit lautet Aufgabe 6.2.1 dann

$$\mu_i(f) = \mu_i(\sum_{j=0}^{n} \alpha_j \varphi_j) = \sum_{j=0}^{n} \alpha_j \mu_i(\varphi_j)$$

für $i = 0, \ldots, n$, was äquivalent ist zu

$$A\alpha = b,$$

wobei $A_{ij} = \mu_i(\varphi_j)$ und $b = \mu_i(f)$ sind. Dieses System besitzt genau dann eine eindeutige Lösung, wenn $\det(A) \neq 0$ ist. ∎

Man beachte, daß das Interpolationsproblem 6.2.1 natürlich nur dann eine eindeutige Lösung haben kann, wenn die Anzahl der Interpolationsbedingungen gleich der Dimension des Raumes G_n, also $n + 1$ ist. Hierzu noch ein

Beispiel 6.2.5. Es sei $G_n = \mathcal{P}_n$ der Raum der Polynome vom Grad n. Wir führen für die Stützstellen $x_0 < x_1 < \ldots < x_n$ mit $\mu_i(f) = f(x_i)$ eine LAGRANGE–Interpolation durch. Eine Basis für G_n sind die *Monome* $\{x^j \mid j = 0, \ldots, n\}$. Nun gilt mit dieser Basis

$$\mu_i(\varphi_j)_{i,j=0,\ldots,n} = V_n := \begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix}, \tag{6.2.6}$$

wobei die Matrix V_n VANDERMONDE–Matrix heißt. Da $\det(V_n) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_j - x_i) \neq 0$ ist für $x_i \neq x_j$ für alle $i, j = 0, \ldots, n$, erfüllen wir mit unseren Voraussetzungen (6.2.4). Also ist das LAGRANGE–Interpolationspolynom in diesem Fall eindeutig lösbar. ∎

Für Polynome, also $G_n = \mathcal{P}_n$, folgt aus dem Fundamentalsatz der Algebra

Satz 6.2.7 (Allgemeine HERMITE–Interpolation für Polynome)**.** Es seien die Stützstellen $x_0 \leq x_1 \leq \ldots \leq x_n$ und für $j = 0, \ldots, n$ die linearen Funktionale

$$\mu_j(f) = f^{(\ell)}(x_j) \tag{6.2.8}$$

mit $\ell = \max\{r \mid x_j = x_{j-r}\}$ und hinreichend glattem f auf $[x_0, \ldots, x_n]$ gegeben. Dann existiert ein eindeutiges Polynom

$$P_n(x) := P(f|x_0, \ldots, x_n)(x) \in \mathcal{P}_n$$

mit der Eigenschaft

$$\mu_j(P_n) = \mu_j(f) \tag{6.2.9}$$

für alle $j = 0, \ldots, n$.

Man interpretiert die Interpolation von Ableitungen formal durch zusammenfallende Stützstellen.

6.3 Darstellung des Interpolationspolynoms

In diesem Abschnitt besprechen wir die LAGRANGE–Interpolation mit Polynomen. Seien dazu $x_0 < x_1 < \ldots < x_n$ paarweise verschiedene Stützstellen, und für $i = 0, \ldots, n$ gelte

$$\mu_i(f) = f(x_i). \tag{6.3.1}$$

Wir beginnen mit der LAGRANGE–Darstellung:

Lemma 6.3.2. Das (eindeutige) Interpolationspolynom $P_n(x) := P(f|x_0, \ldots, x_n) \in \mathcal{P}_n$, das

$$\mu_j(P_n) = P_n(x_j) = f(x_j) \tag{6.3.3}$$

für alle $j = 0, \ldots, n$ erfüllt, läßt sich explizit als

$$P_n(x) = \sum_{j=0}^{n} f(x_j)\ell_{jn}(x) \tag{6.3.4}$$

schreiben, wobei

$$\ell_{jn}(x) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{x - x_k}{x_j - x_k} \tag{6.3.5}$$

für $j = 0, \ldots, n$ die LAGRANGE–Fundamentalpolynome sind.

Beweis: Aufgrund der gewählten Darstellung sind die $\ell_{jn} \in \mathcal{P}_n$ für alle $j = 0, \ldots, n$. Weiter gilt $\ell_{jn}(x_i) = \delta_{ij}$. Daraus folgt

$$P_n(x_i) = \sum_{j=0}^{n} f(x_j)\ell_{jn}(x_i) = f(x_i).$$

Also löst P_n die Interpolationsaufgabe. Zur Eindeutigkeit: Sei $\tilde{P}_n$ eine weitere Lösung. Dann ist $Q := P_n - \tilde{P}_n \in \mathcal{P}_n$ und hat $n + 1$ Nullstellen. Ergo ist $Q \equiv 0$. ∎

Die Darstellung (6.3.4) des Interpolationspolynoms nennt man LAGRANGE–Darstellung. Sie ist nützlich für viele theoretische Fragen, aber numerisch instabil für Stützstellen $x_i \approx x_j$. Außerdem ist sie wegen der Produktbildung $\prod_{k=0}^{n}$ zu rechenaufwendig. Also ist es ratsam, sich nach weiteren Darstellungen für Interpolationspolynome umzusehen. Eine sehr natürlich wirkende Darstellung von P_n ist die bezüglich der monomialen Basis

$$P(f|x_0, \ldots, x_n)(x) = \sum_{j=0}^{n} a_j x^j \tag{6.3.6}$$

mit $a_j \in \mathbb{R}$. Dies ist die *Potenzform*–Darstellung. Zur Berechnung der Koeffizienten a_j muß man das lineare Gleichungssystem

$$V_n \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}$$

lösen, wobei

$$V_n = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \quad \text{und} \quad f(x_i) = \sum_{j=0}^{n} a_j x_i^j$$

sind. Da $\kappa_2(V_n) = \|V_n\|_2\|V_n^{-1}\|_2$ für großes sehr groß werden kann, ist dies für numerische Berechnungen ungeeignet. Auch die Lösung des linearen Gleichungssystems kann problematisch werden, was wir bereits in Kapitel 3 gesehen haben. Ein weiterer Nachteil sowohl der LAGRANGE– wie auch der Potenzform–Darstellung ist, daß man bei Hinzunahme einer neuen Stützstelle stets die gesamte Darstellung neu berechnen muß. Demgegenüber hat die NEWTON–Darstellung der P_n eine Art Aufdatierungscharakter. Dazu zunächst folgendes

Lemma 6.3.7. Für die LAGRANGE–Interpolationspolynome $P_{n-1}(x) = P(f|x_0,\ldots,x_{n-1}) \in \mathcal{P}_{n-1}$ und $P_n = P(f|x_0,\ldots,x_n) \in \mathcal{P}_n$ gilt

$$P_n(x) = P_{n-1}(x) + \delta_n\omega_n(x) \tag{6.3.8}$$

mit

$$\omega_n(x) := \prod_{i=0}^{n-1}(x - x_i) \in \mathcal{P}_n \tag{6.3.9}$$

und

$$\delta_n := \frac{f(x_n) - P_{n-1}(x_n)}{\omega_n(x_n)}. \tag{6.3.10}$$

■

Hierbei nennt man $\{\omega_i, i = 0,\ldots,n\}$ die NEWTON–Basis von $\mathcal{P}_n$. Salopp gesagt, nimmt man immer einen Faktor $(x-x_{n-1})$ hinzu, denn es gilt: $\omega_0 = 1$, $\omega_1 = x-x_0$, $\omega_2 = (x-x_0)(x-x_1),\ldots$.

Beweis: Per definitionem sind $P_{n-1} \in \mathcal{P}_{n-1}$ und $\omega_n = \prod_{i=0}^{n-1}(x - x_i) \in \mathcal{P}_n$. Damit ist

$$Q_n(x) := P_{n-1}(x) + \delta_n\omega_n(x) \in \mathcal{P}_n.$$

Wir müssen zeigen, daß Q_n an den Stellen $x_0,\ldots,x_n$ interpoliert, also $Q_n = P_n$ ist. Dazu betrachten wir für $i = 0,\ldots,n-1$ den Term

$$Q_n(x_i) = P_{n-1}(x_i) + \delta_n\omega_n(x_i).$$

Da $\omega_n(x_i)$ nach Definition hier verschwindet, ist dies äquivalent zu

$$Q_n(x_i) = P_{n-1}(x_i) = f(x_i).$$

Untersuchen wir nun den Fall $i = n$, so gilt

$$Q_n(x_n) = P_{n-1}(x_n) + \delta_n\omega_n(x_n) = f(x_n).$$

Da $\delta_n = \frac{f(x_n)-P_{n-1}(x_n)}{\omega_n(x_n)} \neq 0$, folgt mit Satz 6.2.7, daß Q_n das eindeutige LAGRANGE–Interpolationspolynom von f an den Stellen $x_0,\ldots,x_n$ ist, also $Q_n = P(f|x_0,\ldots,x_n) = P_n$. ■

Man beachte, daß der führende Koeffizient δ_n in (6.3.8) von f und den x_i abhängt. Daher hat sich auch die Schreibweise

$$\delta = [x_0,\ldots,x_n]f \quad \text{oder auch} \quad \delta_n = f[x_0,\ldots,x_n] \tag{6.3.11}$$

eingebürgert. Eine wiederholte Anwendung der Argumentation aus Lemma 6.3.7 liefert uns die NEWTON'sche Interpolationsformel

$$P(f|x_0,\ldots,x_n)(x) = \sum_{i=0}^{n}([x_0,\ldots,x_i]f)\omega_i(x) \tag{6.3.12}$$

oder ausgeschrieben

$$P(f|x_0,\ldots,x_n)(x) = [x_0]f + ([x_0,x_1]f)\omega_1(x) + \ldots + ([x_0,\ldots,x_n]f)\omega_n(x).$$

Dies bedingt die Frage nach einer effizienten Berechnung der $[x_0,\ldots,x_n]f$. Dazu brauchen wir das folgende Lemma, welches auch bei der numerischen Integration eine große Rolle spielt:

Lemma 6.3.13 (Lemma von AITKEN)**.** Es gilt

$$P(f|x_0,\ldots,x_n) = \frac{x-x_0}{x_n-x_0}P(f|x_1,\ldots,x_n)(x) + \frac{x_n-x}{x_n-x_0}P(f|x_0,\ldots,x_{n-1})(x).$$

Die Interpolierende an $x_0,\ldots,x_n$ ist also eine lineare Interpolation zwischen zwei Interpolationspolynomen auf $x_1,\ldots,x_n$ und $x_0,\ldots,x_{n-1}$.

Beweis: Wir setzen x_i in die rechte Seite ein, und schauen uns die Terme für verschiedene Werte von i getrennt an. Für $0 < i < n$ gilt:

$$\begin{aligned}\frac{x_i-x_0}{x_n-x_0}P(f|x_1,\ldots,x_n)(x_i) &+ \frac{x_n-x_i}{x_n-x_0}P(f|x_0,\ldots,x_{n-1})(x_i)\\ &= \frac{x_i-x_0}{x_n-x_0}f(x_i) + \frac{x_n-x_i}{x_n-x_0}f(x_i)\\ &= f(x_i) = P(f|x_0,\ldots,x_n)(x_i),\end{aligned}$$

womit wir die Behauptung für $0 < i < n$ bereits gezeigt haben. Sei $i = 0$. Dann gilt

$$\frac{x_0-x_0}{x_n-x_0}P(f|x_1,\ldots,x_n)(x_0) + \frac{x_n-x_0}{x_n-x_0}P(f|x_0,\ldots,x_{n-1})(x_0) = P(f|x_0,\ldots,x_{n-1})(x_0) = f(x_0),$$

und für $i = n$ gilt

$$\frac{x_n-x_0}{x_n-x_0}P(f|x_1,\ldots,x_n)(x_n) + \frac{x_n-x_n}{x_n-x_0}P(f|x_0,\ldots,x_{n-1})(x_n) = P(f|x_1,\ldots,x_n)(x_n) = f(x_n).$$

Also beide Seiten stimmen für alle $x_0,\ldots,x_n$ überein und sind eindeutige Interpolationspolynome vom Grad $\leq n$, woraus sich die Behauptung ergibt. ∎

Zusätzlich gilt

Bemerkung 6.3.14. Für paarweise verschiedene x_i gilt

$$[x_0,\ldots,x_n]f = \frac{[x_1,\ldots,x_n]f - [x_0,\ldots,x_{n-1}]f}{x_n-x_0}.$$

Hierbei nennt man die linke Seite *dividierte Differenzen der Ordnung n von f.*

Beweis: Setze die NEWTON–Darstellung (6.3.12) in (6.3.13) ein und führe einen Koeffizientenvergleich durch. ∎

Aus Gleichung (6.3.11) wissen wir

$$[x_i]f = f(x_i). \tag{6.3.15}$$

Damit erhalten wir das folgende Schema zur Berechnung der dividierten Differenzen:

	0	1	2	3	⋯
x_0	$[x_0]f$				
		$[x_0,x_1]f$			
x_1	$[x_1]f$		$[x_0,x_1,x_2]f$		
		$[x_1,x_2]f$		$[x_0,x_1,x_2,x_3]f$	
x_2	$[x_2]f$		$[x_1,x_2,x_3]f$		
		$[x_2,x_3]f$			
x_3	$[x_3]f$				
$\vdots$					

(6.3.16)

Für den *Rechenaufwand* dieses Verfahrens gilt: Die Anzahl der Divisionen beträgt: $n+(n-1)+\ldots+1=\frac{1}{2}n(n+1)\doteq\frac{n^2}{2}$. Weitere nützliche Eigenschaften der dividierten Differenzen sind:

Eigenschaften 6.3.17. Für die dividierten Differenzen gilt:

(a) $[x_0,\ldots,x_n]f$ ist unabhängig von der Reihenfolge der x_i.

(b) Ist $p\in\mathcal{P}_n$, so gilt
$$[x_0,\ldots,x_n]p=0.$$

(c) Für ein $x_i<x_{i+1}$ aus $x_0\le\ldots\le x_n$ existiert ein $\xi\in(x_0,x_n)$ mit der Eigenschaft
$$[x_0,\ldots,x_n]f=\frac{f^{(n)}(\xi)}{n!},$$
falls $f\in\mathcal{C}^n([a,b])$.

(d) Gilt speziell für zusammenfallende Knoten $x_0=\ldots=x_n$, so ist
$$[x_0,\ldots,x_0]f=\frac{f^{(n)}(x_0)}{n!},$$
falls $f\in\mathcal{C}^n([a,b])$.

(e) Für $g,h\in\mathcal{C}^n$ und eine beliebige Knotenfolge $x_0\le\ldots\le x_n$ gilt die Leibniz–Regel
$$[x_0,\ldots,x_n](gh)=\sum_{i=0}^{n}([x_0,\ldots,x_n]g)([x_i,\ldots,x_n]h).$$

∎

Damit können wir durch häufige Auswertung von $P_n(x)$ folgendes erreichen: Stelle $P_n(x)$ explizit in Newton–Darstellung auf. Berechne dann die dividierten Differenzen mit dem Rekursionsschema (6.3.16). Zu dessen Auswertung verwende das Horner–Schema. Damit haben wir folgenden

Algorithmus 6.3.18. Gegeben seien x_i und $[x_i]f$ für $i=0,\ldots,n$.

1.) Berechne $[x_0, \ldots, x_i]f$ für $i = 0, \ldots, n$ mit Schema (6.3.16).

2.) Setze $p = \delta_n$ für $k = n-1, \ldots, 0$ und berechne

$$p = \delta_k + (x - x_k)p.$$

Dann ist $P_n(x) = p$.

■

Bei der Auswertung von P_n an nur wenigen einzelnen Stellen stellt man P_n nicht explizit auf sondern benutzt das folgende Schema unter Verwendung von Lemma 6.3.13: Setze $P_{i,k} = P(f|x_{i-k}, \ldots, x_i)$ für $0 \leq k \leq i \leq n$. Speziell gilt dann $P_{n,n}(x) = P(f|x_0, \ldots, x_n)(x)$ und $P_{i,0} = P(f|x_i)(x) = f(x_i)$. Mit 6.3.13 gilt dann

$$P_{i,k} = \frac{x - x_{i-k}}{x_i - x_{i-k}} P_{i,k-1}(x) + \frac{x_i - x}{x_i - x_{i-k}} P_{i-1,k-1}(x),$$

was die Neville'sche Interpolationsformel ist. Damit haben wir das Schema von Neville und Aitken:

	$P_{i,0}$	$P_{i,1}$	$P_{i,2}$
x_0	$f(x_0)$		
		$P_{1,1}$	
x_1	$f(x_1)$		$P_{2,2}$
		$P_{2,1}$	
x_2	$f(x_2)$		$P_{3,2}$
		$P_{3,1}$	
x_3	$f(x_3)$		
$\vdots$			

(6.3.19)

Dies entspricht einem Aufwand von etwa n^2 Mutiplikationen und Divisionen. Seien für ein konkretes Beispiel $n = 2, x_0 = 0, x_1 = 1$ und $x_2 = 2$. Werten wir dies an $x = 0.5$ aus, so ergibt sich $f(0) = 1, f(1) = 4$ und $f(2) = 2$.

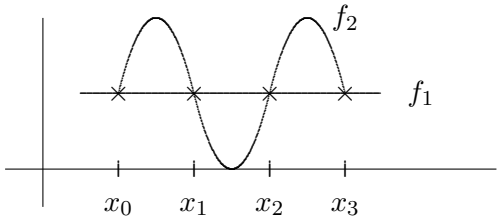
	$P_{i,0}$	$P_{i,1}$	$P_{i,2}$
0	1		
		2.5	
1	4		$\frac{25}{8}$
		5	
2	2		

Für die lineare Interpolation, also $p \in \mathcal{P}_1$ benötigt man mit $n = 1$ 2 Stützstellen und für die kubische mit $p \in \mathcal{P}_3$ und $n = 3$ 4 Stützstellen.

6.4 Verfahrensfehler der LAGRANGE–Interpolation

Wir gehen in diesem Abschnitt der Frage nach, wie stark das Interpolationspolynom von der zu interpolierenden Funktion abweicht.

In nebenstehendem Bild sind durch die Stützstellen $x_0 < \ldots < x_n$ zwei Funktionen f_1 und f_2 gelegt. Das Interpolationspolynom erkennt aber nicht, von welcher Funktion die Daten kommen.



Zur Analyse dieses Problems greifen wir auf die NEWTON–Darstellung zurück. Für festes $x \in [x_0, x_n]$ gilt immer

$$f(x) = P(f|x_0, \ldots, x_n)(x) \in \mathcal{P}_{n+1},$$

was

$$\begin{aligned} f(x) - P(f|x_0, \ldots, x_n)(x) &= P(f|x_0, \ldots, x_n, x)(x) - P(f|x_0, \ldots, x_n)(x) \\ &= ([x_0, \ldots, x_n, x]f)\omega_{n+1}(x) \end{aligned}$$

impliziert. Nun gilt mit 6.3.17(c), daß

$$[x_0, \ldots, x_n]f = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Eine zentrale Abschätzung liefert der folgende

Satz 6.4.1. Sei $f \in \mathcal{C}^{n+1}([x_0, x_n])$. Dann existiert ein $\xi \in [x_0, x_n]$ mit

$$f(x) - P(f|x_0, \ldots, x_n)(x) = \omega_{n+1}(x)\frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{6.4.2}$$

Insbesondere gilt

$$\|f - P(f|x_0, \ldots, x_n)\|_\infty \leq \|\omega_{n+1}\|_\infty \frac{1}{(n+1)!}\|f^{(n+1)}\|_\infty, \tag{6.4.3}$$

wobei $\|f\|_\infty = \max\limits_{x \in [x_0, \ldots, x_n]} |f(x)|$ ist. ∎

Schauen wir uns nochmal gesondert die Abschätzung für das Restglied (6.4.3) an. Wir bemerken, daß $\omega_{n+1} = \prod\limits_{j=0}^{n}(x - x_j)$ von den Stützstellen $x_0, \ldots, x_n$ abhängt, nicht jedoch von f. Umgekehrt hängt $\|f^{(n+1)}\|_\infty$ natürlich von f, aber nicht von den x_i ab. Daher spielt ω_{n+1} offensichtlich eine wichtige Rolle in (6.4.3). Deshalb machen wir zu ω_{n+1} einige Bemerkungen. Seien dazu die x_i äquidistant, also $x_i = x_0 + ih$ mit $h = \frac{1}{n}$ und n ungerade. Man kann zeigen, daß für ein $x = \overline{x} = x_0 + \frac{1}{2}h$ oder $x = \overline{x} = x_n - \frac{1}{2}h$ in der Nähe des Randes

$$|\omega_{n+1}(\overline{x})| = \frac{1}{4n}e^{-n+1}$$

gilt. Ist nun $\tilde{x} = \frac{1}{2}(x_0 + x_n)$, also $\tilde{x}$ in der Mitte des Intervalls, so gilt

$$|\omega_{n+1}(\tilde{x})| \approx \left(\frac{1}{2}\right)^{n+1} ne^{-n+1}.$$

Für großes n ist

$$|\omega_{n+1}(\overline{x})| \gg |\omega_{n+1}(\tilde{x})|.$$

Das bedeutet, die lokalen Maxima von ω_{n+1} liegen nahe den Endpunkten. Allgemeiner gilt

Bemerkung 6.4.4. Für andere Wahlen von x_i kann das Verhältnis verschiedener Funktionswerte von ω_{n+1} wesentlich besser sein. Beispielsweise, wenn die x_i Nullstellen der TSCHEBYSCHEFF–Polynome sind.

Die TSCHEBYSCHEFF–Polynome sind eine spezielle Folge von Polynomen T_k exakt vom Grad k, die bezüglich einem speziellen gewichteten Skalarprodukt auf $[-1,1]$ orthogonal sind, d.h.

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x) T_m(x)\,\mathrm{d}x = \begin{cases} 0 & \text{falls } n \neq m \\ \pi & \text{falls } n = m = 0 \\ \frac{\pi}{2} & \text{falls } n = m \neq 0 \end{cases}$$

Explizit lautet das k-te Polynom

$$T_k(x) = \cos(k \arccos x)$$

mit $x \in [-1,1]$. Die T_k genügen weiter der *Drei–Term–Rekursionsformel*

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

für $k \geq 2$ und mit $T_0(x) = 1$ sowie $T_1(x) = x$. Wählt man als Stützstellen für die Interpolationsaufgabe auf $[-1,1]$ die Nullstellen von ${T_n}^2$, d.h.

$$x_j = \cos\left(\frac{2j+1}{2n+2}\pi\right)$$

für $j = 0, \ldots, n-1$, so wird $\|\omega\|_{\infty,[-1,1]}$ unter allen (normierten) Polynomen mit reellen Nullstellen x_j minimal. Einen Beweis dieser Minimax–Eigenschaft für die TSCHEBYSCHEFF–Polynome findet man in [DH], p. 215ff.. Weiter stellen wir fest

Bemerkung 6.4.5. Obige Herleitung gilt auch für $x \notin [x_0, x_n]$. Ersetze in einem solchen Fall dann $[x_0, x_n]$ durch die *konvexe Hülle*

$$\mathrm{conv}(x_0, \ldots, x_n, x),$$

also das kleinste Intervall, das alle x_i und x enthält. ∎

Prinzipiell könnte man P auch zur Approximation von f an einer Stelle x^* außerhalb von $[x_0, x_n]$ verwenden. Dieses Verfahren nennt man *Extrapolation*. Aber außerhalb von $[x_0, x_n]$ wachsen die Werte von $\omega_{n+1}(x)$ sehr schnell an. Zum Schluß dieses Abschnitts noch

Bemerkung 6.4.6. Obiger Satz läßt sich auch für die HERMITE–Interpolation angeben, wenn

$$\omega_{n+1}(x) = \prod_{i=0}^{n} (x - x_i)$$

und die Ableitungen durch die entsprechenden Vielfachheiten $x_i = x_{i+k}$ charakterisiert sind.

∎

Die bisher zur Interpolation vorgestellten Ergebnisse entsprechen dem *klassischen Teil der Numerik*. Maßgeblich beteiligt an der damaligen Entwicklung waren LAGRANGE und NEWTON. Hauptsächliche Forschungsbereiche waren damals die Interpolation von Funktionen sowie Orthogonalpolynome. Dieses ganze Gebiet gehört heute der numerischen Analysis unter dem Oberbegriff *Approximationstheorie* an.

[2] Man kann zeigen, daß T_n in besagtem Intervall genau n einfache Nullstellen hat.

6.5 Grenzen der Polynominterpolation

Auf den ersten Blick suggeriert Satz 6.4.1, daß man mit einer Erhöhung der Anzahl der Stützstellen und damit von n, beliebig gute Approximationen an f bezüglich der Norm $\|\cdot\|_\infty$ erreichen könnte. Dem ist aber nicht so, wie das folgende Beispiel von RUNGE zeigt. Betrachte die Funktion $f(x) = \frac{1}{1+x^2} \in \mathcal{C}^\infty(\mathbb{R})$. Man kann zeigen, daß für äquidistante Stützstellen

$$x_j = -5 + \frac{10i}{n}$$

mit $i = 0, \ldots, n$ die Folge von Interpolationspolynomen

$$P_n(x) = P(f|x_0, \ldots, x_n)(x)$$

auf $[-5, 5]$ divergiert. Insbesondere an den Intervallgrenzen treten immer stärkere Oszillationen auf.

Eine geeignete Alternative zur besseren Interpolation und Approximation sind Splines, die wir im folgenden Kapitel diskutieren wollen. Diese Theorie beruht sehr stark auf dem *Approximationssatz* von WEIERSTRASS. Dieser besagt, daß sich jede Funktion $f \in \mathcal{C}^0([0, 1])$ beliebig genau durch Polynome genügend hohen Grades approximieren läßt bezüglich der Norm $\|\cdot\|_\infty$.

7 Splinefunktionen

7.1 Historische Vorbemerkungen

Der Begriff *Spline* heißt wörtlich übersetzt dünne Holzlatte. Die für die Numerik wichtige Idee wurde bereits im 18. Jahrhundert im Schiffsbau verwendet. Man zwang Splines durch bestimmte Knotenpunkte. Dadurch stellte sich für Rumpflinien von Schiffen eine günstige Kurve ein, d.h. die Holzlatte nimmt eine Lage ein, bei der die mittlere quadratische Krümmung

$$\left(\int_a^b \frac{S''(x)^2}{\left(1 + S'(x)^2\right)^3} \, \mathrm{d}x \right)^{\frac{1}{2}}$$

minimal wird. Unter der Annahme, daß $S'(x) \ll 1$, können wir den Nenner vernachlässigen. Wir müssen nun eine interpolierende Funktion finden, für die

$$S(x_i) = f_i \tag{7.1.1}$$

für $i = 1, \ldots, n$ und $x_i \in [a, b]$ gilt und die zusätzlich das Glattheitsmaß

$$\left(\int_a^b S''(x)^2 \, \mathrm{d}x \right)^{\frac{1}{2}} \tag{7.1.2}$$

minimiert. Dadurch wird ein „Überschießen“, d.h. Erzeugen von Oszillationen wie bei der Polynominterpolation, vermieden. Bereits im 18. Jahrhundert erkannten Euler und die Gebrüder Bernoulli, daß die Aufgabe (7.1.1) unter der Bedingung (7.1.2) von einer Funktion S gelöst wird, die die folgenden Eigenschaften hat:

- $S\big|_{[x_i, x_{i+1})} \in \mathcal{P}_3$;
- $S \in \mathcal{C}^2([a, b])$.

Mathematisch versteht man unter dem Begriff *Spline* ein stückweises Polynom, das an Stützstellen x_i zusätzlich Glattheitseigenschaften hat.

7.2 Dimensionsbetrachtungen

Bisher bezeichnete $\mathcal{P}_{k-1}$ die Polynome vom *Grad* höchstens $k-1$. Wir verwenden nun dafür auch Π_k, die Polynome der *Ordnung* höchstens k. Nun definieren wir für ein Gitter $\Delta = \{\tau_i\}_{i=0}^{\ell+1}$ mit $\ell + 2$ paarweise verschiedenen Knoten

$$a = \tau_0 < \tau_1 < \ldots < \tau_{\ell+1} = b \tag{7.2.1}$$

den *Splineraum*

$$\mathcal{S}_{k,\Delta} = \left\{ S \in \mathcal{C}^{k-2}([a, b]) \;\middle|\; S\big|_{[\tau_i, \tau_{i+1})} \in \Pi_k \quad \text{für alle } i = 0, \ldots, \ell \right\}. \tag{7.2.2}$$

Im Falle $k = 4$ spricht man von *kubischen Splines*, welche den Polynomen vom Grad drei und $S \in \mathcal{C}^2([a, b])$ entsprechen. Ist $k = 2$, so nennt man die zugehörigen Splines *linear*. Dies sind offenbar die Polynome vom Grad 1 und $S \in \mathcal{C}^0([a, b])$. Sie entsprechen den Polygonzügen.

Wir gehen nun der Frage der Dimension von $\mathcal{S}_{k,\Delta}$ nach. Wir geben uns auf $[a, \tau_i)$ ein beliebiges Polynom $P \in \Pi_k$ vor. Dieses hat offensichtlich k Freiheitsgrade. Das Polynomstück Q auf dem nächsten Intervall $[\tau_1, \tau_2)$ muß sich in $\mathcal{C}^{k-2}$ anschließen, d.h. es muß

$$P^{(j)}(\tau_1) = Q^{(j)}(\tau_1)$$

für $j = 0, \ldots, k-2$ gelten. Also bleiben noch $k-1$ Bedingungen, was heißt, daß für Q selbst ein Freiheitsgrad übrig bleibt. Analog bleibt auch für jedes weitere Teilintervall genau ein Freiheitsgrad übrig. Da man ℓ Teilintervalle $[\tau_i, \tau_{i+1})$ für $i = 1, \ldots, \ell$ gewählt hat, gilt

Satz 7.2.3. $\mathcal{S}_{k,\Delta}$ ist ein $\mathbb{R}$-Vektorraum und es gilt $\dim(\mathcal{S}_{k,\Delta}) = k + \ell$. ∎

Wir machen weiter mit der Frage nach einer Basis für $\mathcal{S}_k, \Delta$. Dazu stellen wir zunächst fest, daß $\mathcal{P}_{k-1} \subset \mathcal{S}_{k,\Delta}$, also $\Pi_k \subset \mathcal{S}_{k,\Delta}$. Für *abgebrochene Potenzen* gilt

$$(x - \tau_i)_+^{k-1} := \begin{cases} (x - \tau_i)^{k-1} & \text{falls } x > \tau_i \\ 0 & \text{falls } x \leq \tau_i \end{cases} \tag{7.2.3a}$$

Damit haben wir die Frage beantwortet.

Satz 7.2.4. Die Menge

$$\{x^i \mid i = 0, \ldots, k-1\} \cup \{(x - \tau_j)_+^{k-1} \mid j = 1, \ldots, \ell\} \tag{7.2.5}$$

bildet eine Basis für $\mathcal{S}_{k,\Delta}$.

Beweis: Wegen 7.2.3 werden $k + \ell$ Basisfunktionen benötigt. In (7.2.5) sind $k + \ell$ Funktionen. Wie im Beweis von (7.2.3) folgert man nun, daß diese den Raum $\mathcal{S}_{k,\Delta}$ erzeugen. Es ist noch zu zeigen, daß die Funktionen in (7.2.5) linear unabhängig sind. Dazu sei $S \in \mathcal{S}_{k,\Delta}$ mit

$$S(x) = \sum_{i=0}^{k-1} a_i x^i + \sum_{j=1}^{\ell} b_j (x - \tau_j)_+^{k-1} = 0 \tag{7.2.6}$$

für alle $x \in [a, b]$. Es ist zu zeigen, daß alle a_i und b_j verschwinden. Dazu wenden wir die linearen Funktionale

$$\mu_r(f) := \frac{1}{(k-1)!} \left(f^{(k-1)}(\tau_r^+) - f^{(k-1)}(\tau_r^-) \right)$$

mit $r = 1, \ldots, \ell$ und τ_r^+, τ_r^- als rechts– bzw. linksseitige Grenzwerte auf S an. Dann gilt

$$0 = \mu_r(S) = \mu_r \left(\sum_{i=0}^{k-1} a_i x^i \right) + \sum_{j=1}^{\ell} b_j \mu_r \left((x - \tau_j)_+^{k-1} \right) = b_r.$$

Da die $(k-1)$–te Ableitung auf $\mathcal{P}_{k-1}$ stetig ist, verschwindet die Differenz in μ_r und damit der erste μ_r–Term in obiger Gleichung.
Im zweiten Term ist $\mu_r \left((x - \tau_j)_+^{k-1} \right) = \delta_{jr}$, da die Differenz in μ_r für $j \neq r$ verschwindet. Also folgt $b_r = 0$ für alle $r = 1, \ldots, \ell$ und damit $S(x) = 0$ für alle $x \in [a, b]$. Da Monome linear unabhängig sind, folgt $a_i = 0$ für $i = 0, \ldots, k-1$. ∎

Allerdings ist die in (7.2.5) angegebene Basis für praktische Zwecke ungeeignet, da die Basisfunktionen nicht lokal sind. Weiterhin sind für Werte $\tau_i \approx \tau_{i+1}$ abgebrochene Potenzen

$$(x - \tau_i)_+^{k-1} \approx (x - \tau_{i+1})_+^{k-1}$$

und diese damit fast linear abhängig, d.h. die Auswertung eines Splines S in der Darstellung bezüglich (7.2.5) und (7.2.6) ist schlecht konditioniert bezüglich Störungen in den Koeffizienten b_i. Desweiteren haben die Koeffizienten keine geometrische Bedeutung. Daher wird es im folgenden unsere Aufgabe sein, eine geeignete Basis für $\mathcal{S}_{k,\Delta}$ zu konstruieren.

7.3 B–Splines

Zur Motivation der nun betrachteten Ausführungen beginnen wir mit zwei Beispielen.

Beispiel 7.3.1. Seien $k = 1$ und

$$\mathcal{S}_{k,\Delta} = \mathcal{S}_{1,\Delta} = \left\{ S \;\middle|\; S\big|_{[\tau_i,\tau_{i+1})} \in \Pi_1,\ i = 0,\dots,\ell \right\}$$

stückweise konstante Funktionen. Eine Basisfunktion für diesen Raum könnte die charakteristische Funktion

$$\chi_{[\tau_i,\tau_{i+1})}(x) = \begin{cases} 1 & \text{falls } x \in [\tau_i, \tau_{i+1}) \\ 0 & \text{falls } x \notin [\tau_i, \tau_{i+1}) \end{cases}$$

sein. Falls $i = \ell$, wähle $\chi_{[\tau_i,\tau_{i+1}]}$, um $\tau_{\ell+1} = b$ nicht auszuschließen. Für den Fall $k = 2$ und

$$\mathcal{S}_{k,\Delta} = \mathcal{S}_{2,\Delta} = \left\{ S \in \mathcal{C}^0([a,b]) \;\middle|\; S\big|_{[\tau_i,\tau_{i+1})} \in \Pi_2,\ i = 0,\dots,\ell \right\}$$

könnte eine Basis aus Hutfunktionen bestehen. ∎

Verallgemeinerungen dieser Basen sind die folgenden, siehe [Bo] für eine umfassende Darstellung.

Definition 7.3.2. Für eine Stützstellenfolge $\Delta = \{\tau_i\}_{i=0}^{\ell+1}$ mit der Eigenschaft (7.2.1) definieren wir die *B–Splines* $N_{i,k}(x)$ der Ordnung k bezüglich $\tau_i, \dots, \tau_{i+k}$ für $k = 1, \dots, \ell$ und $i = 0, \dots, \ell - k + 1$ *rekursiv* durch

$$\begin{aligned} N_{i,1}(x) &= \begin{cases} \chi_{[\tau_i,\tau_{i+1})}(x) & \text{falls } i < \ell \\ \chi_{[\tau_i,\tau_{i+1}]}(x) & \text{falls } i = \ell \end{cases} \\ N_{i,k}(x) &= \underbrace{\frac{x - \tau_i}{\tau_{i+k-1} - \tau_i}}_{\in[0,1] \text{ für } x \in [\tau_i,\tau_{i+k}]} N_{i,k-1}(x) + \underbrace{\frac{\tau_{i+k} - x}{\tau_{i+k} - \tau_{i+1}}}_{\in[0,1] \text{ für } x \in [\tau_i,\tau_{i+k}]} N_{i+1,k-1}(x) \end{aligned} \tag{7.3.3}$$

Da die Gewichte in (7.3.3) nach Konstruktion $\in [0,1]$ für $x \in [\tau_i, \tau_{i+k}]$, handelt es sich um eine Konvexkombination, die numerisch sehr stabil ist.

∎

B–Splines haben folgende nützliche

Eigenschaften 7.3.4. Für die in (7.3.3) definierten B–Splines gilt

(a) $\operatorname{supp}(N_{i,k}) \subseteq [\tau_i, \tau_{i+k}]$ (lokaler Träger), $\quad \operatorname{supp}(f) := \overline{\{x \mid f(x) \neq 0\}}$

(b) $N_{i,k}(x) \geq 0$ für alle $x \in [a,b]$ (Nichtnegativität)

(c) $N_{i,k}(x)|_{[\tau_j,\tau_{j+1}]} \in \Pi_k$, d.h. $N_{i,k}$ ist ein stückweises Polynom der Ordnung k.

(d) $N_{i,k} \in \mathcal{C}^{k-2}([a,b])$ für jedes $i = 0, \dots, \ell - k + 1$ für τ_i paarweise verschieden wie in (7.2.1).

∎

Die rekursive Darstellung in (7.3.3) ist günstig für die praktische Auswertung, allerdings ist eine geschlossene Darstellung für theoretische Zwecke oft geeigneter. Diese wollen wir nun bestimmen.

Lemma 7.3.5. Die in (7.3.3) definierten B-Splines lassen sich auch als

$$N_{i,k}(x) = (\tau_{i+k} - \tau_i)\left([\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-1}\right) \tag{7.3.6}$$

geschlossen darstellen, wobei $[\tau_i, \ldots, \tau_{i+k}]f$ die durch (6.3.14) definierten dividierten Differenzen sind und die abgebrochenen Potenzen $(\cdot - x)_+^{k+1}$ aus (7.2.3a) dem $f(\cdot)$ in (6.3.14) mit Platzhalterargument „$\cdot$“ entsprechen.

Beweis: Wir führen den Beweis über Induktion nach k. Für den Induktionsanfang sei $k = 1$. Es gilt unter Verwendung von (7.3.6)

$$\begin{aligned}(\tau_{i+1} - \tau_i)\left([\tau_i, \tau_{i+1}](\cdot - x)_+^0\right) &= (\tau_{i+1} - \tau_i)\frac{[\tau_{i+1}](\cdot - x)_+^0 - [\tau_i](\cdot - x)_+^0}{\tau_{i+1} - \tau_i} \\ &= (\tau_{i+1} - x)_+^0 - (\tau_i - x)_+^0 \\ &= \chi_{(-\infty,\tau_{i+1})} - \chi_{(-\infty,\tau_i)} = \chi_{[\tau_i,\tau_{i+1})} = N_{i,1}.\end{aligned}$$

Es gelte bereits die Darstellung (7.3.6) für $k-1$ fest aber beliebig. Wir führen den Induktionsschritt von $k-1 \to k$ durch. Dazu formen wir $N_{i,k}$ in die Darstellung (7.3.6) um. Es ist

$$\begin{aligned}N_{i,k}(x) &= (\tau_{i+k} - \tau_i)[\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-1} \\ &= (\tau_{i+k} - \tau_i)[\tau_i, \ldots, \tau_{i+k}]\left(\chi_{(-\infty,\cdot)}(x)(\cdot - x)^{k-1}\right) \\ &= (\tau_{i+k} - \tau_i)[\tau_i, \ldots, \tau_{i+k}]\left((\cdot - x)\chi_{(-\infty,\cdot)}(x)(\cdot - x)^{k-2}\right) \\ &= (\tau_{i+k} - \tau_i)[\tau_i, \ldots, \tau_{i+k}]\left((\cdot - x)(\cdot - x)_+^{k-2}\right).\end{aligned}$$

Mit der Leibnizregel (6.3.17) folgt

$$\begin{aligned}N_{i,k}(x) &= (\tau_{i+k} - \tau_i)\left[\sum_{j=i}^{i+k}[\tau_i, \ldots, \tau_j](\cdot - x)[\tau_j, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2}\right] \\ &= (\tau_{i+k} - \tau_i)\Big[[\tau_i](\cdot - x)[\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} + \\ &\qquad [\tau_i, \tau_{i+1}](\cdot - x)[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} + \\ &\qquad [\tau_i, \tau_{i+1}, \tau_{i+2}](\cdot - x)[\tau_{i+2}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} + \ldots\Big].\end{aligned}$$

Da $(\cdot - x) \in \Pi_2$, gilt nach (6.3.17)(b), daß $[\tau_1, \ldots, \tau_m](\cdot - x) = 0$ für alle $m > 2$. Folglich bleiben nur die ersten zwei Terme der Summe erhalten. Wende nun (6.3.14) an

$$\begin{aligned}N_{i,k}(x) = (\tau_{i+k} - \tau_i)\Big[&(\tau_i - x)[\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} \\ &+ \frac{(\tau_{i+1} - x) - (\tau_i - x)}{\tau_{i+1} - \tau_i}[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2}\Big]\end{aligned}$$

Es gilt $\frac{(\tau_{i+1}-x)-(\tau_i-x)}{\tau_{i+1}-\tau_i} = 1$, und es folgt nach erneuter Anwendung von (6.3.14)

$$
\begin{aligned}
N_{i,k}(x) &= (\tau_{i+k}-\tau_i)\left[(\tau_i - x)\left(\frac{[\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2} - [\tau_i,\ldots,\tau_{i+k-1}](\cdot - x)_+^{k-2}}{\tau_{i+k}-\tau_i}\right)\right. \\
&\qquad \left. + [\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2}\right] \\
&= (\tau_i - x)\left([\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2} - [\tau_i,\ldots,\tau_{i+k-1}](\cdot - x)_+^{k-2}\right) \\
&\qquad + (\tau_{i+k}-\tau_i)[\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2} \\
&= (\tau_i - x + \tau_{i+k} - \tau_i)[\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2} - (\tau_i - x)[\tau_i,\ldots,\tau_{i+k-1}](\cdot - x)_+^{k-2} \\
&= (\tau_{i+k} - x)[\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2} - (\tau_i - x)[\tau_i,\ldots,\tau_{i+k-1}](\cdot - x)_+^{k-2} \\
&= \frac{x-\tau_i}{\tau_{i+k-1}-\tau_i}N_{i,k-1}(x) + \frac{\tau_{i+k}-x}{\tau_{i+k}-\tau_{i+1}}N_{i+1,k-1}(x).
\end{aligned}
$$

■

Bemerkung 7.3.7. Obige Rekursionsformeln (7.3.3) lassen sich genauso angeben, wenn Stützstellen τ_i zusammenfallen. Man beachte aber, daß $N_{i,k}(x) = \chi_{[\tau_i,\tau_{i+1})}(x) \equiv 0$ falls $\tau_i = \tau_{i+1}$. Entsprechende Terme werden dann in der Rekursion zu Null gesetzt. ■

Weiter gilt noch

Bemerkung 7.3.8. Die Rekursionsformeln für die Ableitungen von B–Splines erhält man direkt aus der expliziten Darstellung (7.3.6):

$$
\begin{aligned}
N'_{i,k}(x) &= (k-1)(\tau_{i+k}-\tau_i)\left([\tau_i,\ldots,\tau_{i+k}](\cdot - x)_+^{k-2}\right)(-1) \\
&= -(k-1)(\tau_{i+k}-\tau_i)\left(\frac{[\tau_{i+1},\ldots,\tau_{i+k}](\cdot - x)_+^{k-2} - [\tau_i,\ldots,\tau_{i+k-1}](\cdot - x)_+^{k-2}}{\tau_{i+k}-\tau_i}\right) \\
&= -(k-1)\left[\frac{1}{\tau_{i+k}-\tau_{i+1}}N_{i+1,k-1}(x) - \frac{1}{\tau_{i+k-1}-\tau_i}N_{i,k-1}(x)\right] \\
&= (k-1)\left[\frac{N_{i,k-1}(x)}{\tau_{i+k-1}-\tau_i} - \frac{N_{i+1,k-1}(x)}{\tau_{i+k}-\tau_{i+1}}\right],
\end{aligned}
$$

folglich reduziert sich die Berechnung der Ableitungen auf die Auswertung einer Linearkombination von B–Splines niedrigerer Ordnung. ■

Wir kommen nun zur *Auswertung von Splinefunktionen.* Auf der Basis von B–Splines lassen sich nun Splinefunktionen angeben. Um die Intervallgrenzen mit berücksichtigen zu können, betrachte die erweiterte Knotenfolge

$$T := \{\theta_i\}_{i=1}^{n+k} \tag{7.3.9}$$

mit $\theta_i < \theta_{i+k}$ für $i = 1,\ldots,n$, also

$$\theta_1 = \ldots = \theta_k = a < \theta_{k+1} \le \ldots \le \theta_n < b = \theta_{n+1} = \ldots = \theta_{n+k}.$$

Man sieht, daß a und b k-fach besetzt sind.
Definiere nun das lineare Erzeugnis der B–Splines auf T als

$$\mathcal{N}_k(T) = \mathcal{N}_{k,T} := \operatorname{span}\left(\{N_{i,k} : i = 1,\ldots,n\}\right), \tag{7.3.10}$$

d.h. jedes Element $S \in \mathcal{N}_k(T)$ besitzt eine Darstellung

$$S(x) = \sum_{i=1}^{n} c_i N_{i,k}(x) \tag{7.3.11}$$

für $x \in [a,b]$. Durch Einsetzen der Rekursionsformel (7.3.3) bekommt man daraus

$$S(x) = \sum_{i=r+1}^{n} c_i^{[r]}(x) N_{i,k-r}(x), \tag{7.3.12}$$

wobei

$$c_i^{[r]} := \begin{cases} c_i & \text{falls } r = 0 \\ \dfrac{x-\theta_i}{\theta_{i+k-r}-\theta_i} c_i^{[r-1]}(x) + \dfrac{\theta_{i+k-r}-x}{\theta_{i+k-r}-\theta_i} c_{i-1}^{[r-1]}(x) & \text{falls } r > 0 \\ 0 & \text{falls } \theta_{i+k-r} = \theta_i \end{cases} \tag{7.3.13}$$

Speziell für den Fall $r = k-1$ folgt für $\theta \in [\theta_i, \theta_{i+1})$ aus (7.3.12)

$$N_{i,k-r}(x) = N_{i,1}(x) = \chi_{[\theta_i,\theta_{i+1})}$$

und damit

$$S(x) = c_i^{[k-1]}(x) \tag{7.3.14}$$

für $x \in [\theta_i, \theta_{i+1})$. Zur rekursiven Berechnung der $c_i^{[r]}(x)$ bietet sich ein NEVILLE–artiges Schema an:

$$\begin{array}{ccccccccc} c_{j-k+1} & & & & & & & & \\ & \searrow & & & & & & & \\ c_{j-k+2} & \to & c_{j-k+2}^{[1]}(x) & & & & & & \\ & \searrow & & \searrow & & & & & \\ c_{j-k+3} & \to & c_{j-k+3}^{[1]}(x) & \to & c_{j-k+3}^{[2]} & & & & \\ \vdots & & & & & & & & \\ & \searrow & & \searrow & & \cdots & \searrow & & \\ c_j & \to & c_j^{[1]}(x) & \to & c_j^{[2]}(x) & \cdots & \to & c_j^{[k-1]} \end{array} \tag{7.3.15}$$

Der Aufwand zur Auswertung von S entspricht dem zur Auswertung *eines* B–Splines nach (7.3.3).

Wir müssen noch zeigen, daß die $N_{i,k}$ tatsächlich eine Basis für den Splineraum $\mathcal{S}_{k,\Delta}$ bilden. Vorher bringen wir aber noch einige theoretische Ergebnisse. Wir beginnen mit der *Reproduktion von Polynomen* und dem folgenden

Satz (MARSDEN–Identität): Für alle $x \in [a,b]$ und $\sigma \in \mathbb{R}$ gilt

$$\begin{aligned} (x-\sigma)^{k-1} &= \sum_{i=1}^{n} \prod_{j=1}^{k-1} (\theta_{i+j} - \sigma) N_{i,k}(x) \\ &= \sum_{i=1}^{n} \varphi_{i,k}(\sigma) N_{i,k}(x) \end{aligned} \tag{7.3.16}$$

mit $\varphi_{i,k}(\sigma) := \prod_{j=1}^{k-1} (\theta_{i+j} - \sigma)$.

Beweis: Der Beweis läuft über vollständige Induktion nach k. Der Induktionsanfang für $k = 1$ ist einfach, da

$$(x-\sigma)^0 = 1 = \sum_{i=1}^{n} 1 \cdot N_{i,1}(x) = \sum_{i=1}^{n} \chi_{(\theta_i,\theta_{i+1})}(x) = \chi_{(\theta_1,\theta_{n+1})}(x) = \chi_{(a,b)}(x) = 1$$

gilt. Wir führen den Induktionsschritt von $k-1 \to k$ aus. Dazu nehmen wir an, daß die Behauptung für $r \leq k-1$ gilt. Betrachte die rechte Seite in (7.3.16), die die Darstellung (7.3.11) mit $c_i := \varphi_{i,k}(\sigma)$ hat:

$$S(x) = \sum_{i=1}^{n} c_i N_{i,k}(x) = \sum_{i=2}^{n} c_i^{[1]}(x) N_{i,k-1}(x)$$

Wende nun Definition (7.3.13) von $c_i^{[r]}$ für $r = 1$ an:

$$S(x) = \sum_{i=2}^{n} \left(\frac{x-\theta_i}{\theta_{i+k-1}-\theta_i} c_i + \frac{\theta_{i+k-1}-x}{\theta_{i+k-1}-\theta_i} c_{i-1} \right) N_{i,k-1}(x)$$

Nun sind genau nach Definition $c_i = \varphi_{i,k}(\sigma)$ und $c_{i-1} = \varphi_{i-1,k}(\sigma)$. Folglich gilt:

$$S(x) = \sum_{i=2}^{n} \left(\frac{x-\theta_i}{\theta_{i+k-1}-\theta_i} \prod_{j=1}^{k-1} (\theta_{i+j}-\sigma) + \frac{\theta_{i+k-1}-x}{\theta_{i+k-1}-\theta_i} \prod_{j=1}^{k-1} (\theta_{i-1+j}-\sigma) \right) N_{i,k-1}(x)$$

Wende eine Indexverschiebung im zweiten Produkt an, sodaß $\prod_{j=1}^{k-1}(\theta_{i-1+j}-\sigma) = \prod_{j=0}^{k-2}(\theta_{i+j}-\sigma)$. Gleiche nun beide Produkte an durch herausziehen des letzten Faktors des 1. Produkts und des ersten Faktors des 2. Produkts. Ziehe dann die gemeinsamen Faktoren $\prod_{j=1}^{k-2}(\theta_{i+j}-\sigma)$ heraus:

$$S(x) = \sum_{i=2}^{n} \prod_{j=1}^{k-2} (\theta_{i+j}-\sigma) \left(\frac{x-\theta_i}{\theta_{i+k-1}-\theta_i} (\theta_{i+k-1}-\sigma) + \frac{\theta_{i+k-1}-x}{\theta_{i+k-1}-\theta_i} (\theta_i-\sigma) \right) N_{i,k-1}(x)$$

Der Ausdruck in der großen Klammer ist linear von x abhängig und genau gleich $x-\sigma$, da es die lineare Interpolation von $x-\sigma$ an θ_i und θ_{i+k-1} ist.

Betrachte nun

$$\operatorname{supp}(N_{1,k-1}) \subseteq [\theta_1, \theta_{1+k-1}] = [\theta_1, \theta_k] = 0$$

Daraus folgt, daß $N_{1,k-1}(x) = 0$ für $x \in [a,b]$, und es nach der Definition von $\varphi_{i,k-1}(\sigma)$ unter Anwendung der Induktionsvoraussetzung gilt:

$$\begin{aligned} S(x) &= (x-\sigma) \sum_{i=2}^{n} \varphi_{i,k-1}(\sigma) N_{i,k-1}(x) \\ &= (x-\sigma) \sum_{i=1}^{n} \varphi_{i,k-1}(\sigma) N_{i,k-1}(x) \\ &= (x-\sigma)^{k-1} \end{aligned}$$

■

Einige Konsequenzen hieraus sind

Korollar 7.3.17. Die Menge der Polynome vom Grad höchstens $k-1$ auf $[a,b]$ ist eine Untermenge von span $(\{N_{i,k} \mid i = 1,\dots,n\})$ auf der Knotenfolge T, also

$$\mathcal{P}_{k-1} \subseteq \mathcal{N}_k(T).$$

Beweis: Betrachte die ℓ-te Ableitung der MARSDEN–Identität nach σ, ausgewertet an $\sigma = 0$. Es gilt:

$$\begin{aligned}\left(\frac{d}{d\sigma}\right)^{\ell}(x-\sigma)^{k-1}\Big|_{\sigma=0} &= \Big((k-1)\cdots(k-\ell)(x-\sigma)^{k-1-\ell}(-1)^{\ell}\Big)\Big|_{\sigma=0}\\ &= (k-1)\cdots(k-\ell)x^{k-1-\ell}(-1)^{\ell}\\ &= \sum_{i=1}^{n}\varphi_{i,k}^{(\ell)}(0)N_{i,k}(x)\end{aligned}$$

Aus der letzten Identität folgt mit $m = k-1-\ell$ folgende Darstellung

$$x^m = \frac{(-1)^{k-1-m}}{(k-1)\cdots(m+1)}\sum_{i=1}^{n}\varphi_{i,k}^{(k-1-m)}(0)N_{i,k}(x) \tag{7.3.18}$$

für Monome mit $m = 0,\dots,k-1$.

Folglich lassen sich die Monome bis Grad $k-1$ als Linearkombination von B-Splines $N_{i,k}$ der Ordnung k darstellen. Da $\mathcal{P}_{k-1}$ die Monome $x^0,\dots,x^{k-1}$ als Basis besitzt, folgt die Behauptung. ∎

Speziell für $m = 0$ ist

$$\Big(\varphi_{i,k}^{(k-1)}(\sigma)\Big)\Big|_{\sigma=0} = (k-1)(k-2)\cdots 2\cdot 1\cdot(-1)^{k-1} = (k-1)!(-1)^{k-1}$$

und damit unabhängig von σ. Denn nach Definition gilt

$$\varphi_{i,k} = \prod_{j=1}^{k-1}(\theta_{i+j}-\sigma) = (-\sigma)^{k-1} + \mathcal{O}(\sigma^{k-2})$$

Damit folgt direkt aus (7.3.18)

$$x^0 = 1 = \sum_{i=1}^{n} N_{i,k}(x) \tag{7.3.19}$$

für alle $x \in [a,b]$, was bedeutet, daß die B–Splines eine *Zerlegung der Eins* bilden. Aus (7.2.4) wissen wir, daß die Menge

$$\{x^i, i = 0,\dots,k-1, \quad \text{und} \quad (x-\tau_j)_+^{k-1}, j = 0,\dots,l\}$$

eine Basis für $S_{k,\Delta}$ ist. Diese Basis ist *global* linear unabhängig, also auf ganz $[a,b]$. Im Unterschied dazu gilt wegen der Lokalität der B–Splines

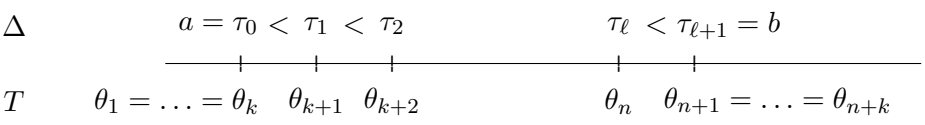
Satz 7.3.20. Die $N_{i,k}$ sind lokal linear unabhängig, d.h. gilt

$$\sum_{i=1}^{n} c_i N_{i,k}(x) = 0$$

für $x \in (c,d) \subseteq [a,b]$, so ist $c_i = 0$, falls $(c,d) \cap (\theta_i, \theta_{i+k}) \neq \emptyset$.

Beweis: O.B.d.A. enthalte das Intervall (c,d) keine Knoten θ_i. Ansonsten zerlege man (c,d) in Teilintervalle. Wegen der Darstellung der Potenzen in (7.3.18) und Korollar 7.3.17 lassen sich alle Polynome mit Grad $\leq k-1$ auf (c,d) durch B–Splines darstellen und, da $\dim \Pi_{k-1} = k$ auf (c,d), gibt es nur k B–Splines, die nicht verschwinden. Diese müssen demnach linear unabhängig sein. ∎

Jetzt sind wir in der Lage zu zeigen, daß die B–Splines eine Basis für $\mathcal{S}_{k,\Delta}$ bilden. Dazu müssen wir die Stützstellen und Knoten geeignet wählen. In Erinnerung an (7.2.1) ergibt sich das folgende Schema:



Damit haben wir

Satz 7.3.21. Mit Δ, T und $n = k + \ell$ wie oben gilt

$$\mathcal{S}_{k,\Delta} = \mathcal{N}_{k,T},$$

d.h. die B–Splines der Ordnung k bilden eine Basis für den Raum der Splinefunktionen $\mathcal{S}_{k,\Delta}$.

Beweis: Wegen 7.3.4 (c): $N_{i,k}\big|_{[\tau_j,\tau_{j+1})} \in \mathcal{P}_{k-1}$ und $N_{i,k} \in \mathcal{C}^{k-2}([a,b])$ folgt

$$\mathcal{N}_{k,T} \subseteq \mathcal{S}_{k,\Delta}.$$

Aufgrund der lokalen linearen Unabhängigkeit der $N_{i,k}$ gilt

$$\dim(\mathcal{N}_{k,T}) = n = k + \ell = \dim(\mathcal{S}_{k,\Delta}),$$

woraus unsere Behauptung folgt. ∎

Allgemeiner kann man sogar zeigen, daß man für k–fache Randknoten in T jeweils μ_i–fache innere Knoten θ_i wählen kann, also $\theta_1 = \ldots = \theta_k$ und $\theta_{k+1} = \ldots = \theta_{k+\mu_1}$ usw.. Dies drücken wir aus in folgendem

Satz 7.3.22. Es ist

$$S_{k,\mu,\Delta} = \mathcal{N}_{k,T}.$$

Ist $\mu_j = 1$ für alle $j = 1, \ldots, \ell$, so gilt

$$S_{k,\mu,\Delta} = \mathcal{S}_{k,\Delta}.$$

In diesem Fall hat man also eine höchstmögliche Glattheit an den Stützstellen erreicht. Ist $\mu_j > 1$, so fordert man weniger Glattheit an den einzelnen Stützstellen. Zum Schluß des Abschnitts geben wir noch

Satz 7.3.23 (Stabilität der B–Spline Basis)**.** Man kann zeigen, daß mit $\mathbf{c} := (c_i)_{i=1,\ldots,n}$

$$C_k \|\mathbf{c}\|_\infty \leq \left\| \sum_{i=1}^{n} c_i N_{i,k} \right\|_{\infty,[a,b]} \leq \|\mathbf{c}\|_\infty,$$

d.h. die B–Splines bilden eine *unkonditionell stabile Basis* für $\mathcal{S}_{k,\Delta}$, denn unabhängig von der Knotenfolge T läßt sich $S = \sum_{i=1}^{n} c_i N_{i,k}$ von oben und unten durch die Entwicklungskoeffizienten c_i abschätzen. Daher sagt man auch, daß die B–Splines eine *gut konditionierte* Basis bilden.

Beweis: Die obere Abschätzung läuft über eine Zerlegung der Eins, die untere mit Konstruktion einer dualen Basis. ∎

7.4 Splineinterpolation mit B–Splines

Die ursprüngliche Motivation für die Konstruktion der B–Splines waren Interpolationsprobleme. Sei $T = \{\theta_i\}_{i=1,\ldots,n+k}$ eine erweiterte Knotenfolge wie oben, also

$$\begin{array}{lcccccccc} \Delta & & a=\tau_0 & < & \tau_1 & < & \tau_2 & \quad\ldots\quad & \tau_\ell & < & \tau_{\ell+1}=b \\ & & | & & | & & | & & | & & | \\ T & \theta_1=\ldots=\theta_k & & \theta_{k+1} & & \theta_{k+2} & & & \theta_n & \theta_{n+1}=\ldots=\theta_{n+k} & \end{array}$$

Mit $\mathcal{N}_{k,T} = \operatorname{span}(\{N_{i,k} \mid i = 1,\ldots,n\})$ und $\dim(\mathcal{N}_{k,T}) = n$ folgt, daß man n Bedingungen stellen muß, um eine wohldefinierte Interpolationsaufgabe zu stellen. Folgender Satz besagt, wann eine Interpolationsaufgabe für alle Daten eindeutig lösbar ist.

Satz 7.4.1. Sei $T = \{\theta_i\}_{i=1,\ldots,n+k}$ wie oben und seien $x_1 < \ldots < x_n \in [a,b]$ Stützstellen. Das Problem

$$\text{Finde zu Daten } f_1,\ldots,f_n \text{ ein } S \in \mathcal{N}_{k,T} \text{ mit } S(x_i) = f_i \text{ für } i = 1,\ldots,n \tag{7.4.2}$$

besitzt genau dann eine eindeutige Lösung, wenn

$$x_i \in \begin{cases} [\theta_i, \theta_{i+k}) & \text{falls } i = 1 \\ (\theta_i, \theta_{i+k}) & \text{falls } i = 2,\ldots,n-1 \\ (\theta_i, \theta_{i+k}] & \text{falls } i = n \end{cases} \tag{7.4.3}$$

für alle $i = 1,\ldots,n$, d.h. wenn in den Träger jedes B–Splines genau eine Stützstelle fällt.

Beweis: Nach Satz 6.2.3 ist die Interpolationsbedingung genau dann eindeutig lösbar, wenn $\det((N_{i,k}(x_j))_{i,j=1,\ldots,n}) \neq 0$ ist. Dies ist nach dem Satz von Schoenberg–Whitney genau dann der Fall, wenn die $N_{i,k}(x_i) \neq 0$ für alle $i = 1,\ldots,n$ sind. Dies aber ist äquivalent dazu, daß die x_i im Innern der Träger liegen. ∎

Bemerkung 7.4.4. Man kann sogar zeigen, daß der Fall, daß $A = (N_{i,k}(x_j))_{i,j=1,\ldots,n}$ *total positiv* ist, d.h. alle $r \times r$ Unterdeterminanten nichtnegativ sind, äquivalent dazu ist, daß $A\mathbf{c} = \mathbf{f}$ stabil mit Gauss–Elimination ohne Pivotisierung gelöst werden kann. ∎

Man beachte, daß wegen $\operatorname{supp}(N_{i,k}) = [\theta_i, \theta_{i+k}]$ die Matrix A die Bandbreite $k-1$ besitzt, also dünn besetzt ist. Für den Fall $k = 4$ leiten wir $A\mathbf{c} = \mathbf{f}$ her:

Kubische Spline–Interpolation mit B–Splines

Für $k = 4$ wandelt sich unser Schema zu:

$$\begin{array}{lcccccccc} \Delta & & a=x_1 & < & x_2 & < & x_3 & \quad\ldots\quad & x_{n-3} & < & x_{n-2}=b \\ & & | & & | & & | & & | & & | \\ T & \theta_1=\ldots=\theta_4 & & \theta_5 & & \theta_6 & & & \theta_n & \theta_{n+1}=\ldots=\theta_{n+5} & \end{array}$$

Wähle nun Stützstellen $x_1 = \theta_4,\ x_2 = \theta_5, \ldots, x_{n-2} = \theta_{n+1}$. Dies stellt $n-2$ Bedingungen an x_i. Da aber $\dim(\mathcal{N}_{k,T}) = n$ fehlen uns also noch zwei Bedingungen. Typische *Wahlmöglichkeiten* sind

(I) *Vollständige kubische Spline–Interpolation:* Zusätzlich zu den Funktionswerten gibt man noch die ersten Ableitungen am Rand an. Somit erhält man einen vollständigen Satz von Interpolationsbedingungen:

$$S(\theta_i) = f(\theta_i) \tag{7.4.2a}$$

für $i = 4, \ldots, n+1$ zusammen mit

$$S'(a) = f'(a)$$
$$S'(b) = f'(b).$$

(II) *Natürliche Spline–Interpolation:* Anschaulich entspricht diese Methode dem Auslaufen der dünnen Holzlatten am Rande, denn man fordert

$$S''(a) = S''(b) = 0.$$

(III) *Keine–Knoten–Bedingung:* Falls am Rand keine Knoten zur Verfügung stehen, verschweiße kubische Polynomstücke auf $[\theta_4, \theta_5], [\theta_5, \theta_6]$ zu einem. Am rechten Rand verfahre entsprechend.

Für die vollständige kubische Spline–Interpolation gilt

Satz 7.4.6. Zu jedem $f \in \mathcal{C}^2([a,b])$ existiert ein eindeutiger Spline $S = I_4 f \in \mathcal{N}_{4,T}$, sodaß für $i = 4, \ldots, n+1$

$$\begin{aligned} (I_4 f)(\theta_i) &= f(\theta_i), \\ (I_4 f)'(a) &= f'(a) \\ (I_4 f)'(b) &= f'(b), \end{aligned} \tag{7.4.7}$$

gilt. Desweiteren erfüllt $I_4 f$ die *Extremaleigenschaft*

$$\|(I_4 f)''\|_2 \leq \|g''\|_2 \tag{7.4.7a}$$

für jede Interpolierende $g \in \mathcal{C}^2([a,b]) \cap L_2([a,b])$, die auch (7.4.7) erfüllt, und es gilt die Fehlerabschätzung

$$\|f - I_4 f\|_{\infty,[a,b]} \leq \frac{5}{384} h^4 \|f^{(4)}\|_{\infty,[a,b]} \tag{7.4.8}$$

für jedes $f \in \mathcal{C}^4([a,b])$, wobei $h = \max_i |\theta_{i+1} - \theta_i|$ und $\|f\|_{\infty,[a,b]} := \max_{x \in [a,b]} |f(x)|$ ist.

Beweis: Die Existenz und Eindeutigkeit zeigt man mit Satz 6.2.3. Die Extremaleigenschaft findet man etwa in [DH], Kapitel 7.4. Ein Beweis der Fehlerabschätzungen steht in [SB2], §2.4. ∎

Approximationsgüte von Splines

Bemerkung. *Zur Lösung der Aufgabe 7.4.7 mit 7.4.7a vergleiche mit der zu Anfang gestellten Interpolationsaufgabe 7.1.1 unter Einbeziehung von 7.1.2.*

Beachte.

- *In 7.4.8 kann das Einfügen von Stützstellen bewirken, daß h kleiner und somit die Abschätzung genauer wird.*
- *7.4.8 ist* unabhängig *von der* Lage der Stützstellen*, im Unterschied zur Polynominterpolation, bei der die rechte Seite von (6.4.3) $\|\omega_n\|_\infty \frac{1}{n} \|f^{(n)}\|_\infty$ lautete).*

Berechnung der vollständigen kubischen Spline-Interpolation mit B-Splines

Wir wollen $S = I_4 f$ berechnen aus den Interpolationsbedinungen 7.4.2a

$$\begin{aligned} S(\theta_i) &= f(\theta_i), \quad i = 4, \dots, n+1 \\ S'(a) &= f'(a), \quad S'(b) = f'(b) \end{aligned}$$

Folglich ist ein $S \in \mathcal{N}_{4,T}$ gesucht und damit ein lineares Gleichungssystem zu lösen (vgl. Beweis von Satz 7.4.1).

$$S(\theta_i) \left(= \sum_{j=1}^{n} c_j N_{j,k}(\theta_i) \right) = f(\theta_i), \quad i = 4, \dots, n+1 \tag{7.4.9}$$

Es gilt (z.B. mit Rekursionsformeln 7.3.3)

$$\begin{aligned} N_{j,4}(\theta_4) &= 0, \quad j = 2, 3, \dots \\ N_{j,4}(\theta_{n+1}) &= 0, \quad j = n-1, n-2, \dots \end{aligned}$$

Das heißt, die B-Splines $N_{j,4}$ verschwinden an a, b für $j = 2, \dots, n-1$.

Desweiteren

$$\begin{aligned} N_{1,4}(\theta_4) = 1 \quad \text{wegen} \quad \sum_{i=1}^{n} N_{i,k}(x) = 1 \forall x \in [a,b] \\ N_{n,4}(\theta_{n+1}) = 1 \quad \text{(Zerlegung der Eins)} \end{aligned}$$

Eingesetzt in 7.4.9 folgt damit

$$\begin{aligned} S(\theta_4) &= c_1 = f(\theta_4) \\ S(\theta_{n+1}) &= c_n = f(\theta_{n+1}). \end{aligned}$$

Folglich müssen in 7.4.9 nur noch die $n-2$ Koeffizienten $c_2, \dots, c_{n-1}$ berechnet werden.

Mit gleicher Argumentation schließt man aus Bedingungen an Ableitungen & Rekursionsformeln für Ableitungen von B-Splines 7.3.8

$$\begin{aligned} c_2 &= \frac{f'(a) - f(a) N'_{1,4}(a)}{N'_{2,4}(a)} \\ c_{n-1} &= \frac{f'(b) - f(b) N'_{n,4}(b)}{N'_{n-1,4}(b)} \end{aligned}$$

Damit sind nur noch die $n-4$ Koeffizienten $\underline{\tilde{c}} := (c_3, \dots, c_{n-2})^T$ zu bestimmen.

Löse folglich $\tilde{A}\underline{\tilde{c}} = \underline{\tilde{f}}$, wobei

$$\tilde{A} := \begin{pmatrix} N_{3,4}(\theta_5) & N_{4,4}(\theta_5) & & & \\ N_{3,4}(\theta_6) & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & N_{n-2,4}(\theta_{n-1}) \\ & & & N_{n-3,4}(\theta_n) & N_{n-2,4}(\theta_n) \end{pmatrix}$$

eine Tridiagonalmatrix der Bandbreite $k-1=3$ und $\underline{\tilde{f}} := (f_3, \ldots, f_{n-2})^T$ mit

$$\begin{aligned} f_3 &:= f(\theta_5) - c_2 N_{2,4}(\theta_5) \\ f_j &:= f(\theta_{j+2}) \qquad\qquad j = 4, \ldots, n-3 \\ f_{n-2} &:= f(\theta_n) - c_{n-1} N_{n-1,4}(\theta_n) \end{aligned}$$

ist.

7.5 Multivariate Splines

Die Theorie zu mehrdimensionalen Splines lässt sich nicht annähernd so einheitlich entwickeln und darstellen wie die univariate Splinetheorie [Bo].

Beispiel 7.5.1 (Einfachster Fall)**.** Tensorproduktansätze *auf* Rechteckgittern *für zwei Variablen* x, y.

Gegeben zwei Knotenfolgen $T = \{\theta_i\}_{i=1,\ldots,n+k}$ *für* x *und* $U = \{u_j\}_{j=1,\ldots,m+k}$ *für* y *auf Intervallen* $[a,b]$ *und* $[c,d]$.
Daraus ergibt sich ein Knotengitter mit den Knoten (θ_i, u_j), $i = 1, \ldots, n+k; j = 1, \ldots, m+k$.

Setze $N_{i,j}(x,y) := N_{i,k,T}(x) N_{j,k,U}(y)$, *welches dieselbe Ordnung* k *in* x- *und* y-*Richtung hat. Der Splineraum* $\mathcal{N}_k(T,U) := \operatorname{span}\{N_{i,j} \mid i = 1, \ldots, n, j = 1, \ldots, m\}$ *hat somit Dimension* mn.

Allgemeines Interpolationsproblem 7.5.2. *Seien Stützstellen* (x_r, y_s), $r = 1, \ldots, n$, $s = 1, \ldots, m$ *und Daten* $f_{r,s}$ *gegeben. Finde*

$$S(x,y) := \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} N_{i,j}(x,y) \in \mathcal{N}_k(T,U),$$

sodaß für alle r, s *die Voraussetzung* $S(x_r, y_s) = f_{r,s}$ *erfüllt ist.*

Zur Lösung von 7.5.2 gilt es ein lineares Gleichungssystem $A\underline{\mathbf{c}} = \underline{\mathbf{f}}$ zu lösen: Betrachte somit

$$S(x_r, y_s) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} N_{i,j}(x_r, y_s) = f_{r,s}$$

Laut Definition von $N_{i,j}(x,y)$ als Tensorprodukt und nach dem Vertauschen der Summen ergibt sich

$$\sum_{j=1}^{m} \underbrace{\left(\sum_{i=1}^{n} c_{i,j} N_{i,k,T}(x_r) \right)}_{=:a_j^r} N_{j,k,U}(y_s) = f_{r,s}$$

Die Lösung lässt sich wie folgt auf eindimensionale Probleme reduzieren:

- Bestimme zunächst für jedes $r = 1, \ldots, n$ die Koeffizienten a_j^r als Lösung von

$$\sum_{j=1}^{m} a_j^r N_{j,k,U}(y_s) = f_{r,s},$$

  welches ein lineares Gleichungssystem für $s = 1, \ldots, m$ ist.
  Folglich sind n eindimensionale Systeme vom Rang m zu lösen.

- Löse dann noch für $j = 1, \ldots, m$

$$\sum_{i=1}^{n} c_{i,j} N_{i,k,T}(x_r) = a_j^r$$

  für $r = 1, \ldots, n$.
  Folglich handelt es sich um m eindimensionale Systeme vom Rang n.

Daraus ergibt sich eine Komplexität von $m\mathcal{O}(n) + n\mathcal{O}(m)$ arithmetischen Operationen, da die einzelnen Systeme aufgrund des kompakten Trägers von B-Splines Bandmatrizen beinhalten, die jeweils mit $\mathcal{O}(n)$ bzw. $\mathcal{O}(m)$ Operationen gelöst werden können.

Zwar ist $A \in \mathbb{R}^{(nm)\times(nm)}$ auch dünn besetzt, aber direkte Löser würden Fill-Ins erzeugen und die dünne Besetzung aufheben.

Beispiel 7.5.3 (Anderer Standardfall)**.** *Splines auf Dreiecksgittern ($\hat{=}$ Finite Elemente)*

Bei Courant-Finite-Elementen ist $N_{i,j}(x, y) \neq N_i(x)N_j(y)$ wie bei Tensorprodukt-Hutfunktionen. Die Lösung des zugehörigen linearen Gleichungssystems erfolgt daher in der Regel mit iterativen Lösern.

8 Numerische Quadratur

Gegeben ist eine (stückweise) stetige Funktion f auf einem Intervall $[a,b] \subset \mathbb{R}$. Gesucht ist das (Riemann-)Integral (siehe Abb. 1)

$$I(f) := \int_a^b f(x)\,\mathrm{d}x \tag{8.0.1}$$

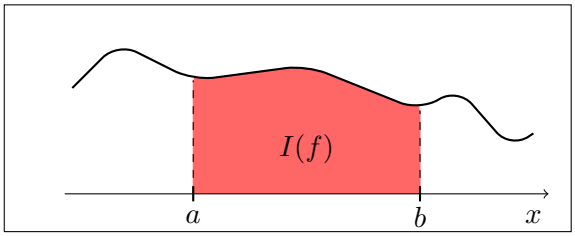


Abbildung 1: Integral $I(f)$ auf $[a,b]$

Häufig ist $I(f)$ nicht geschlossen (*analytisch*) theoretisch berechenbar. Dies führt auf das Problem, eine approximative Berechnung des Flächeninhalts durchzuführen. Dies nennt man Numerische Integration, Numerische Quadratur, Numerische Kubatur falls $I(f) = \int_Q f(x)\,\mathrm{d}x$ für $Q \subset \mathbb{R}^n,\ n > 1$.

8.1 Einleitung, klassische Methoden

Aufgabe: Berechne $I(f) := \int_a^b f(x)\,\mathrm{d}x$ mit $f \in \mathcal{C}([a,b])$.

Idee: Ersetze f durch eine möglichst ähnliche Funktion $\tilde{f}$, sodaß $I(\tilde{f})$ berechenbar ist.

Kondition der Quadratur

Betrachte den Ausgabefehler der Integration einer Funktion f mit ihrer Störung $\tilde{f}$:

$$|I(f) - I(\tilde{f})| = \left| \int_a^b (f(x) - \tilde{f}(x))\,\mathrm{d}x \right| \le \int_a^b |f(x) - \tilde{f}(x)|\,\mathrm{d}x \le (b-a)\|f - \tilde{f}\|_\infty$$

Folglich ist die absolute Kondition des Integrationsproblems gut, wenn $\tilde{f}$ die Funktion f möglichst genau approximiert.

Für den relativen Fehler gilt:

$$\frac{|I(f) - I(\tilde{f})|}{|I(f)|} \le \frac{(b-a)\|f - \tilde{f}\|_\infty}{\left| \int_a^b f(x)\,\mathrm{d}x \right|} \frac{\|f\|_\infty}{\|f\|_\infty} = \frac{\left| \int_a^b \|f\|_\infty\,\mathrm{d}x \right|}{\left| \int_a^b f(x)\,\mathrm{d}x \right|} \frac{\|f - \tilde{f}\|_\infty}{\|f\|_\infty} = \kappa_{\mathrm{rel}} \frac{\|f - \tilde{f}\|_\infty}{\|f\|_\infty}$$

Falls f *stark oszilliert*, kann

$$\left| \int_a^b f(x)\,\mathrm{d}x \right| \ll (b-a)\|f\|_\infty$$

gelten und folglich $\kappa_{\text{rel}} \gg 1$. Daraus ergibt sich eine *schlechte* relative Kondition.

Klassische Integrationsverfahren.

1. Unterteile $[a, b]$ in n Teilintervalle $[t_k, t_{k+1}]$ für $k = 0, \ldots, n-1$. Die t_k nennt man **Stützstellen**.

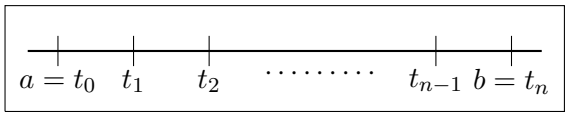


Abbildung 2: Intervallaufteilung von $[a, b]$

2. Approximiere f auf jedem Teilintervall $[t_{k-1}, t_k]$ durch eine *einfach zu integrierende* Funktion g_k. Dies führt auf

$$I(f) = \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x \approx \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} g_k(x)\,\mathrm{d}x.$$

Speziell: Wähle t_k äquidistant, das heißt $t_k = a + hk$, wobei $h := \frac{b-a}{n}$ (oder $h := t_k - t_{k-1}$) als Schrittweite, Diskretisierungsparameter oder Gitterweite bezeichnet wird.

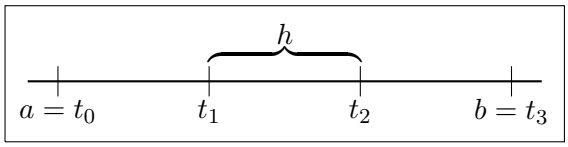


Abbildung 3: Schrittweitendarstellung mit $n = 3$

Wir betrachten im Folgenden äquidistante Stützstellen.

Beispiel 8.1.1. Rechteckregel
Wähle

$$g_k(x) := \begin{cases} f(t_{k-1}) & \text{falls } x \in [t_{k-1}, t_k] \\ 0 & \text{sonst} \end{cases}$$

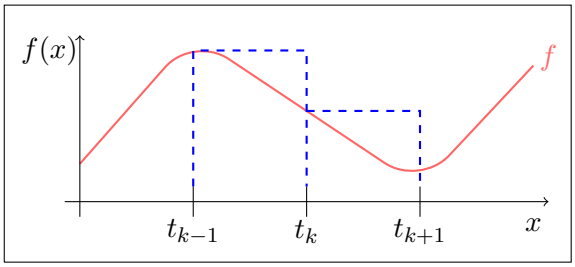


Abbildung 4: $g_k(x)$ auf einem Teilintervall

Man erhält somit

$$I(f) \approx R(h) := h \sum_{k=1}^{n} f(t_{k-1})$$

Berechnung der Güte der Approximation in Abhängigkeit von der Gitterweite h unter der Annahme, dass der Integrand f „genügend glatt“ auf $[a,b]$ ist, mittels Taylorentwicklung:
Taylorentwicklung von f um t_{k-1}: $f(x) = f(t_{k-1}) + (x - t_{k-1})f'(\xi)$, $x \in [t_{k-1}, t_k], \xi \in [t_{k-1}, t_k]$

$$\Rightarrow hf(t_{k-1}) \overset{h=t_k-t_{k-1}}{=} \int_{t_{k-1}}^{t_k} f(t_{k-1})\,\mathrm{d}x \overset{\text{Taylor}}{=} \int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x - f'(\xi) \underbrace{\int_{t_{k-1}}^{t_k} (x - t_{k-1})\,\mathrm{d}x}_{=\frac{(x-t_{k-1})^2}{2}|_{t_{k-1}}^{t_k} = \frac{h^2}{2}}$$

$$= \int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x + \mathcal{O}(h^2)$$

$$\Rightarrow R(h) = h\sum_{k=1}^{n} f(t_{k-1}) = \sum_{k=1}^{n}\Big(\int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x\Big) + n\mathcal{O}(h^2) \overset{h=\frac{b-a}{n}}{=} I(f) + \mathcal{O}(h)$$

das heißt, der Fehler ist von der Ordnung $\mathcal{O}(h)$, also ein Verfahren 1. Ordnung. Dies bedeutet, dass der Fehler

$$|E_h(f)| := |I(f) - R(h)| = \mathcal{O}(h)$$

für $h \to 0$ ($h < 1$) von der Ordnung $\mathcal{O}(h)$ gegen 0 geht (Fehlerabschätzung). Allgemein gilt $|E_h(f)| = \mathcal{O}(h^r)$ für ein Verfahren $r-$ter Ordnung ($r > 0$) und $h = 2^{-j}, j = 0, 1, 2, \ldots$.

Beachte, dass folgende wesentliche Annahmen erfüllt sein müssen:

- Der Integrand muss „genügend glatt“ sein (wegen der Taylorentwicklung).
- Die Schrittweite h muss immer gleich sein, das heißt die Stützstellen müssen äquidistant sein.

Ist beides nicht erfüllt, kann man die obige Aussage nicht treffen.

Beispiel 8.1.2. Mittelpunktsregel
Wähle

$$g_k(x) := \begin{cases} f(\frac{t_{k-1}+t_k}{2}) & \text{falls } x \in [t_{k-1}, t_k] \\ 0 & \text{sonst} \end{cases}$$

Dann erhält man

$$R(h) = h\sum_{k=1}^{n} f\left(\frac{t_{k-1} + t_k}{2}\right) \approx I(f)$$

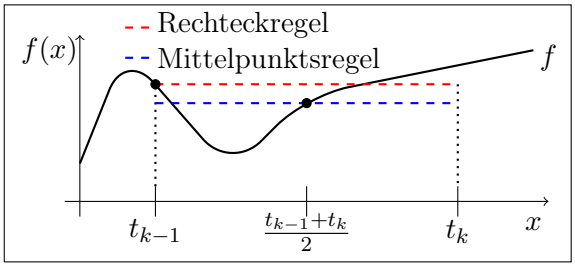


Abbildung 5: Anschaulicher Unterschied der Regeln

Man kann mit der Taylorentwicklung zeigen (mit $\int_{t_{k-1}}^{t_k} (x - \frac{1}{2}(t_{k-1} - t_k))\ dx = 0$), dass für $f \in C^2([t_{k-1}, t_k])$ gilt:

$$\int_{t_{k-1}}^{t_k} g_k(x)\,\mathrm{d}x = \int_{t_{k-1}}^{t_k} f\left(\frac{t_{k-1}+t_k}{2}\right)\,\mathrm{d}x = \int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x - \frac{f''(\xi_k)}{24}h^3$$

für ein $\xi_k \in [t_{k-1}, t_k]$.
Damit folgt:

$$\begin{aligned} R(h) &= h\sum_{k=1}^{n} f\left(\tfrac{t_{k-1}+t_k}{2}\right) = \sum_{k=1}^{n}\int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x - \sum_{k=1}^{n}\tfrac{f''(\xi_k)}{24}h^3 \\ &= \int_a^b f(x)\,\mathrm{d}x - \sum_{k=1}^{n}\tfrac{f''(\xi_k)}{24}h^3 =: I(f) - E_h(f) \end{aligned}$$

Für den *Diskretisierungsfehler* (Verfahrensfehler) erhält man somit:

$$|E_h(f)| \le \frac{h^3}{24}\sum_{k=1}^{n}|f''(\xi_k)| \le \frac{h^3}{24}\cdot n \cdot \underbrace{\|f''\|_\infty}_{:=\max_{x\in[a,b]}|f''(x)|} \overset{h=\frac{b-a}{n}}{=} \frac{h^2}{24}(b-a)\|f''\|_\infty$$

Das heißt, die Mittelpunktsregel ist ein Verfahren 2. Ordnung, da $|E_h(f)| = \mathcal{O}(h^2)$ gilt (der Fehler geht mit $\mathcal{O}(h^2)$ gegen 0 für $h \to 0$).

Am bekanntesten ist das folgende

Beispiel 8.1.3. Trapezregel
Ersetze f auf $[t_{k-1}, t_k]$ durch eine lineare Interpolation von $f(t_{k-1})$ und $f(t_k)$, das heißt ersetze f auf $[a, b]$ durch einen Polygonzug (ein linearer Spline, d.h. eine stetige, stückweise lineare Funktion) durch $f(t_k), k = 0, \ldots, n$.

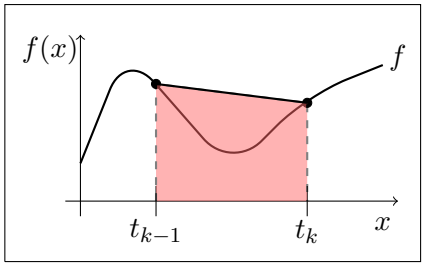


Abbildung 6: Ersetzung von f durch Polygonzug

Wähle also

$$g_k(x) := \begin{cases} \frac{x-t_{k-1}}{h}f(t_{k-1}) + \frac{t_k-x}{h}f(t_k) & \text{falls } x \in [t_{k-1}, t_k] \\ 0 & \text{sonst} \end{cases}$$

(Dabei ist $g_k(t_{k-1}) = f(t_k)$ und $g_k(t_k) = f(t_{k-1})$.)
Wir erhalten:

$$\int_{t_{k-1}}^{t_k} g_k(x)\,\mathrm{d}x = h \cdot \frac{1}{2}\left(f(t_{k-1}) + f(t_k)\right)$$

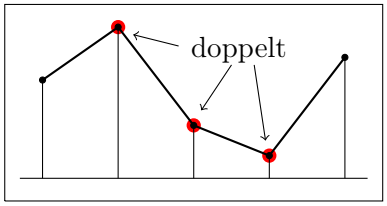


Abbildung 7: doppelte t_k's

(Die Fläche ist ein Trapez.)
Somit ist die Approximation von $I(f)$ die Trapezsumme

$$R(h) = h\left(\frac{1}{2}(f(a)+f(b)) + \sum_{k=1}^{n-1} f(t_k)\right)$$

Wie bei der Mittelpunktsregel kann man für $f \in C^2([t_{k-1}, t_k])$ zeigen:

$$h\int_{t_{k-1}}^{t_k} g_k(x)\,\mathrm{d}x = h\frac{1}{2}(f(t_{k-1})+f(t_k)) = \int_{t_{k-1}}^{t_k} f(x)\,\mathrm{d}x + \frac{f''(\xi_k)}{12}h^3$$

für ein $\xi_k \in [t_{k-1}, t_k]$ und es ergibt sich

$$|E_h(f)| = |I(f) - R(h)| \leq \ldots \leq h^2\frac{(b-a)}{12}\|f''\|_\infty,$$

also ebenso ein Verfahren 2. Ordnung.

8.2 Die Newton-Cotes-Formeln

Allgemein. *Schreibe nun $[c,d]$ anstelle von $[t_{k-1}, t_k]$ für ein typisches Teilintervall. Obige Beispiele sind von folgendem Typ:*
Seien $x_0, \ldots, x_m \in [c,d]$ paarweise verschieden.

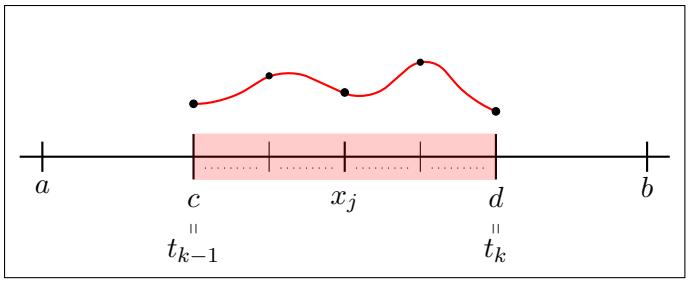


Abbildung 8: typisches Teilintervall $[c,d] \subset [a,b]$

Approximiere f durch das Interpolationspolynom $P(f|x_0, \ldots, x_m)(x) \in \mathbb{P}_m$ (Menge der Polynome vom Grad m), das heißt

$$\tilde{I}(f) := \int_c^d f(x)\,\mathrm{d}x \approx \int_c^d P(f|x_0, \ldots, x_m)(x)\,\mathrm{d}x =: I_m(f). \tag{8.2.1}$$

Beachte: die rechte Seite ist einfach zu berechnen, da $P(f|x_0,\dots,x_m)(x)$ ein Polynom ist.
Beispiele:

Rechteckregel:	$m=0, x_0=c$
Mittelpunktsregel:	$m=0, x_0=\frac{c+d}{2}$
Trapezregel:	$m=1, x_0=c, x_1=d$

Bemerkung 8.2.2.
Sei $I_m(f)$ definiert wie in (8.2.1). Dann gilt für jedes Polynom $Q \in \mathbb{P}_m$:

$$I_m(Q) = I(Q) = \int_c^d Q(x)\,\mathrm{d}x,$$

das heißt die Quadraturformel ist exakt vom Grad m.

Beweis: Die Interpolation von $m+1$ Punkten liefert ein eindeutiges Interpolationspolynom $P(Q|x_0,\dots,x_m)(x) \in \mathbb{P}_m$ mit $P(Q|x_0,\dots,x_m) = Q$ für jedes $Q \in \mathbb{P}_m$. Die Integration gleicher Integranden liefert das gleiche Integral. ■

Die praktische Auswertung von $I_m(f)$ benötigt die Darstellung des Interpolationspolynoms. Wähle eine Darstellung über Lagrange-Fundamentalpolynome

$$P(f|x_0,\dots,x_m)(x) = \sum_{j=0}^{m} f(x_j) \prod_{\substack{k=0\\k\neq j}}^{m} \frac{x-x_k}{x_j-x_k}$$

(Daraus folgt:

$$P(f|x_0,\dots,x_m)(x_i) = \sum_{j=0}^{m} f(x_j) \underbrace{\prod_{\substack{k=0\\k\neq j}}^{m} \frac{x_i-x_k}{x_j-x_k}}_{=\delta_{ij}} = f(x_i)$$

für $i=0,\dots,m$).

Lemma 8.2.3.
Es gibt Gewichte $c_0,\dots,c_m$, sodaß $I_m(f)$ definiert in (8.2.1) die Darstellung

$$I_m(f) = \tilde{h} \cdot \sum_{j=0}^{m} c_j f(x_j)$$

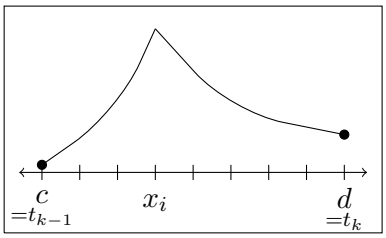


Abbildung 9: $I(f)$ auf Intervall $[c,d]$

mit $\tilde{h} := d-c$ erhält. Die Gewichte sind gegeben durch

$$c_j = \frac{1}{\tilde{h}} \int_c^d \prod_{\substack{k=0\\k\neq j}}^{m} \frac{x-x_k}{x_j-x_k}\,\mathrm{d}x.$$

Beweis: Übung.

Zur Wahl der Stützstellen x_j: Wählt man auf $[c,d]$ äquidistante Stützstellen $x_j := c + \frac{j}{m}\tilde{h}$ mit $\tilde{h} = d-c, j = 0,\ldots,m$, führt dies auf die Newton-Cotes-Formeln (*Interpolatorische Quadraturregeln*)

$$I_m(f) := \tilde{h}\sum_{j=0}^{m} c_j f(c+\xi_j\tilde{h}) \tag{8.2.4}$$

wobei $\xi_j := \frac{j}{m}$ (*normierte Stützstellen*) und die Gewichte c_j aus Lemma 8.2.3 sind.

Spezialfälle.
Hierbei sei:

m: der Grad des Interpolationspolynoms

D: der Diskretisierungsfehler $|E_h(f)| := |I_m(f) - \int_c^d f(x)dx|$

$\mathcal{O}$: die Ordnung auf dem Intervall $[a,b]$

m	Name	Stützstellen ξ_j	Gewichte c_j	$D \leq$	$\mathcal{O}$
0	Rechteckregel	0	1	$\mathcal{O}(h^2)$	1
0	Mittelpunktsregel	$\frac{1}{2}$	1	$h^3\frac{1}{24}\|f''\|_{\infty,[c,d]}$	2
1	Trapezregel	$0,1$	$\frac{1}{2},\frac{1}{2}$	$h^3\frac{1}{12}\|f''\|_{\infty,[c,d]}$	2
2	Simpsonregel	$0,\frac{1}{2},1$	$\frac{1}{6},\frac{2}{3},\frac{1}{6}$	$(\frac{h}{2})^5\frac{1}{90}\|f^{(4)}\|_{\infty,[c,d]}$	4
3	$\frac{3}{8}$-Regel	$0,\frac{1}{3},\frac{2}{3},1$	$\frac{1}{8},\frac{1}{3},\frac{2}{3},\frac{1}{8}$	$(\frac{h}{3})^5\frac{3}{80}\|f^{(4)}\|_{\infty,[c,d]}$	4
4	Milne-Regel	$0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1$	$\frac{7}{90},\frac{32}{90},\frac{12}{90},\frac{32}{90},\frac{7}{90}$	$(\frac{h}{4})^7\frac{8}{345}\|f^{(6)}\|_{\infty,[c,d]}$	6

Für $m > 6$ werden die Formeln numerisch unbrauchbar, wegen negativer Gewichte (c_j zum Teil < 0) besteht die Gefahr der Auslöschung.
Beachte, dass die $h-$te Potenz im Restglied für m und $m+1$ gleich ist, falls m gerade ist.

Nun zur wiederholten Anwendung der Newton-Cotes-Formeln:
Unterteile dazu $[a,b]$ in Teilintervalle $[t_{k-1},t_k]$ mit $t_k = a+kh$ und $h = \frac{b-a}{n}$.

Beispiel 8.2.5.
Die Anwendung der Simpsonregel (d.h. quadratisches Interpolationspolynom) auf jedem Teilintervall $[c,d] = [t_{k-1},t_k], k = 1,\ldots,n$ liefert die summierte Simpsonregel

$$R(h) = \int_a^b f(x)\,\mathrm{d}x + E_h(f)$$

mit

$$R(h) = h\left(\frac{1}{6}f(t_0) + \frac{2}{3}f(\frac{t_0+t_1}{2}) + \frac{1}{3}f(t_1) + \frac{2}{3}f(\frac{t_1+t_2}{2}) + \frac{1}{3}f(t_2) + \ldots + \frac{1}{6}f(t_n)\right)$$

und

$$|E_h(f)| = |R(h) - I(f)| \leq \frac{h^4}{2880}(b-a)\|f^{(4)}\|_{\infty,[a,b]}.$$

8.3 Die Euler-MacLaurinsche Summenformel

Zur Abschätzung des Diskretisierungsfehlers bei den Newton-Cotes-Formeln wurden die Identitäten

$$hf\left(\frac{c+d}{2}\right) = \int_c^d f(x)\,\mathrm{d}x - \frac{f''(\xi)}{24}h^3, \qquad \xi \in [c,d] \quad \text{(Mittelpunktsregel)}$$

$$h\frac{1}{2}\left(f(c)+f(d)\right) = \int_c^d f(x)\,\mathrm{d}x + \frac{f''(\xi)}{12}h^3, \qquad \xi \in [c,d] \quad \text{(Trapezregel)}$$

verwendet, die über die Taylorentwicklung hergeleitet wurden.

Diese sind Spezialfälle der Euler-MacLaurinschen Summenformel. Dazu werden Bernoullizahlen benötigt, die wiederum über Bernoullipolynome definiert werden.
Zunächst setzen wir $[c,d] = [0,1]$.

Bemerkung 8.3.1. *Die Folge von Polynomen $\{B_k(x)\}_{k\geq 0}$ definiert durch*

$$\begin{aligned} B_0(x) &:= 1 \\ B_1(x) &:= x - \frac{1}{2} \\ B_2(x) &:= x^2 - x + \frac{1}{6} \\ B_3(x) &:= 1^3 - \frac{3}{2}x^2 + \frac{1}{2}x \\ B_4(x) &:= x^4 - 2x^3 + x^2 - \frac{1}{30} \end{aligned}$$

heißen Bernoulli-Polynome. *Sie sind allgemein durch die Rekursion*

$$B'_{k+1}(x) = (k+1)B_k(x), \quad k = 1,2,\ldots$$

definiert, die B_{i+1} allerdings nur bis auf eine Konstante festlegt. Diese ist so gewählt, daß

$$B_{2k+1}(0) = B_{2k+1}(1) = 0, \quad k > 0$$

Weiterführendes zu Bernoullipolynomen findet man in [S].

Man nennt $B_k := B_k(0)$ Bernoullizahlen, die ungleich Null für gerades $k > 1$ sind.

$$B_0 = 1, \quad B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_6 = \frac{1}{42}, \quad B_8 = -\frac{1}{30}$$

Lemma 8.3.2 (Euler-MacLaurinsche Summenformel)**.** *Es gilt für $g \in \mathcal{C}^{2s+2}([0,1])$*

$$\int_0^1 g(x)\,\mathrm{d}x = \frac{g(0)}{2} + \frac{g(1)}{2} + \sum_{k=1}^{s} \frac{B_{2k}}{(2k)!}\left(g^{(2k-1)}(0) - g^{(2k-1)}(1)\right) - \frac{B_{2s+2}}{(2s+2)!}g^{(2s+2)}(\xi), \quad \xi \in (0,1).$$

Beweis: (vollständiger Beweis zu finden z.B. in [S])
Forme $\int_0^1 g(x)\,\mathrm{d}x$ sukzessive durch partielle Integration um und verwende die Rekursionsformel für Bernoullipolynome folgendermaßen

$$\begin{aligned} \int_0^1 g(x)\,\mathrm{d}x &= \int_0^1 B_0(x)g(x)\,\mathrm{d}x = \Big[B_1(x)g(x)\Big]_0^1 - \int_0^1 B_1(x)g'(x)\,\mathrm{d}x \\ &= \Big[B_1(x)g(x)\Big]_0^1 - \Big[\frac{1}{2}B_2(x)g(x)\Big]_0^1 - \frac{1}{2}\int_0^1 B_2(x)g''(x)\,\mathrm{d}x \\ &= \ldots \end{aligned}$$

∎

Nun leiten wir die Darstellungen für $\int_0^n g(x)\,\mathrm{d}x$ und daraufhin für $\int_a^b f(x)\,\mathrm{d}x$ her, welche uns Zugang zu Extrapolationsverfahren verschaffen.
Die Wiederholte Anwendung von 8.3.2 auf Integrale $\int_{i-1}^{i}$ und Summation über $i = 1, \ldots, n$ liefert nun

$$\begin{aligned}\int_0^n g(x)\,\mathrm{d}x = \sum_{i=1}^{n}\int_{i-1}^{i} g(x)\,\mathrm{d}x = \frac{g(0)}{2} + g(1) + \ldots + g(n-1) + \frac{g(n)}{2} \\ + \sum_{k=1}^{s}\frac{B_{2k}}{(2k)!}\left(g)^{2k-1)}(0) - g^{(2k-1)}(n)\right) \\ - \frac{B_{2s+2}}{(2s+2)!}\sum_{i=1}^{n} g^{(2s+2)}(\xi_i), \quad \xi_i \in (i-1, i) \\ = \ldots - \frac{B_{2s+2}}{(2s+2)!} n g^{(2s+2)}(\xi), \quad \xi \in (0, n),\end{aligned} \tag{8.3.3}$$

denn

$$\min_i\left(g^{(2s+2)}(\xi_i)\right) \le \frac{1}{n}\sum_{i=1}^{n} g^{(2s+2)}(\xi_i) \le \max_i\left(g^{(2s+2)}(\xi_i)\right)$$

und damit gibt es für ein stetiges $g^{(2s+2)}$ ein $\xi \in (\min_i(\xi_i), \max_i(\xi_i))$ mit
$g^{(2s+2)}(\xi) = \frac{1}{n}\sum_{i=1}^{n} g^{(2s+2)}(\xi_i)$.

Für ein beliebiges Intervall $[a, b]$ mit äquidistanten Bruchstellen $t_k = a + kh$, $k = \frac{b-a}{n}$, $k = 0, \ldots, n$ leite die Summenformel 8.3.3 mittels Variablensubstitution wie folgt her. Setze $g(x) = f(a + xh)$ und folgere

- $\int_0^n g(x)\,\mathrm{d}x = \int_0^n f(a + xh)\,\mathrm{d}x \underset{t=a+xh}{=} \int_a^{a+nh=b} f(t)\,\frac{\mathrm{d}t}{h} = \frac{1}{h}\int_a^b f(t)\,\mathrm{d}t$
- $g^{(k)}(x) = f^{(k)}(a + xh)h^k, \quad k \ge 0$
- $\frac{g(0)}{2} + g(1) + \ldots + g(n-1) + \frac{g(n)}{2} = \frac{f(a)}{2} + f(a+h) + f(a+2h) + \ldots + f(b-h) + \frac{f(b)}{2}$ $= \frac{1}{k}T(k)$, wobei $T(h) = R(h)$ die *Trapezsumme* aus 8.1.3 ist.

Damit ergibt sich aus 8.3.3 insgesamt

$$\begin{aligned}\frac{1}{h}\int_a^b f(t)\,\mathrm{d}t = \frac{1}{h}T(h) + \sum_{k=1}^{s}\frac{B_{2k}}{(2k)!}h^{2k-1}\left(f^{(2k-1)}(a) - f^{(2k-1)}(b)\right) \\ - \frac{B_{2s+2}}{(2s+2)!} n h^{2s+2} f^{(2s+2)}(\xi), \quad \xi \in (a, b)\end{aligned}$$

oder noch etwas umgeschrieben mit h und $T(h)$ auf der linken Seite und Vertauschen von $f(a)$ und $f(b)$

$$\begin{aligned}T(h) = \int_a^b f(t)\,\mathrm{d}t + \sum_{k=1}^{s}\frac{B_{2k}}{(2k)!}h^{2k}\left(f^{(2k-1)}(b) - f^{(2k-1)}(a)\right) \\ + \frac{B_{2s+2}}{(2s+2)!}(b-a)h^{2s+2} f^{(2s+2)}(\xi), \quad \xi \in (a, b)\end{aligned} \tag{8.3.4}$$

Beachte: Dies ist eine Entwicklung der Trapezsumme nach Potenzen von h. 8.3.4 lässt sich auch schreiben als

$$T(h) = \tau_0 + \tau_1 h^2 + \tau_2 h^4 + \ldots + t_5 h^{25} + \alpha_{s+1}(h)h^{2s+2} \tag{8.3.5}$$

wobei

$$\tau_0 := \int_a^b f(t)\,\mathrm{d}t = I(f)$$
$$\tau_i := \frac{B_{2i}}{(2i)!}\left(f^{(2i-1)}(b) - f^{(2i-1)}(a)\right)$$
$$\alpha_{s+1}(h) := \frac{B_{2s+2}}{(2s+2)!}(b-a)f^{(2s+2)}(\xi(h)), \quad \xi = \xi(h) \in (a,b)$$

mit

$$|\alpha_{s+1}(h)| \leq \left|\frac{B_{2s+2}}{(2s+2)!}(b-a)\right| \|f^{(2s+2)}\|_\infty \tag{8.3.6}$$

für alle $h = \frac{b-a}{n}$.

Entwicklungen der Form 8.3.5, bei denen die Koeffizienten τ_i unabhängig von h sind und das Restglied unabhängig von h beschränkt ist, heißen *asymptotische Entwicklungen.*

Für $h = 0$ liefert $T(h)$ das gewünschte Integral

$$T(0) = \tau_0 = \int_a^b f(t)\,\mathrm{d}t$$

Als nächstes wird anhand der asymptotischen Entwicklung für die Trapezsumme 8.3.5 gezeigt, wie man die Genauigkeit verbessern kann.

8.4 Extrapolation

Die Extrapolation ist ein wichtiges Prinzip der Numerik und liefert eine allgemeine Methode zur Verbesserung der Genauigkeit.
Im Fall der numerischen Integration lässt sich dies wie folgt für die Trapezregel anwenden.
Erinnere: Zur Berechnung von $\int_a^b f(x)\,\mathrm{d}x$ liefert die Trapezsumme
$T(h) = h\left(\frac{f(a)}{2} + f(a+h) + \ldots + f(b-h) + \frac{f(b)}{2}\right)$ 8.1.3 eine Approximation der Ordnung $\mathcal{O}(h^2)$.

Andererseits besitzt die Trapezsumme $T(h)$ eine asymptotische Entwicklung 8.3.5, falls $f \in \mathcal{C}^{2s+2}([a,b])$:

$$T(h) - \tau_0 = \tau_1 h^2 + \tau_2 h^4 + \tau_3 h^6 + \ldots + \tau_5 h^{25} + \mathcal{O}(h^{2s+2}) \tag{8.4.1}$$

Idee: Zur Verbesserung der Genauigkeit:

- Betrachte $T(h)$ für Schrittweiten $h_0 = h, h_1 = \frac{h}{2}, h_2 = \frac{h}{4}, \ldots, h_m = \frac{h}{2^m}$.
- Bilde aus $T(h_0), T(h_1), T(h_2), \ldots, T(h_m)$ ein Interpolationspolynom in h^2, welches dann in $h = 0$ ausgewertet wird. Das Verfahren heißt deshalb „Extrapolation“, da $h \notin \{h_0, \ldots, h_m\}$.

Betrachte 8.4.1 und wende das obige Verfahren durch Bildung von $T\left(\frac{h}{2}\right)$ an.

$$T\left(\frac{h}{2}\right) - \tau_0 = \frac{\tau_1}{4}h^2 + \underbrace{\frac{\tau_2}{2^4}}_{=:\hat{\tau}_2} h^4 + \underbrace{\frac{\tau_3}{2^6}}_{=:\hat{\tau}_3} h^6 + \ldots + \underbrace{\frac{\tau_s}{2^{2s}}}_{=:\hat{\tau}_s} h^{2s} + \mathcal{O}(h^{2s+2})$$

Eliminiere den h^2-Term durch Multiplikation mit $\frac{4}{3}$ und Subtraktion des $\frac{1}{3}$-fachen von 8.4.1.

$$\left(\frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h)\right) - I(f) = \hat{\tau}_2 h^4 + \hat{\tau}_3 h^6 + \ldots \tag{8.4.2}$$

Man sieht, daß durch einfaches Kombinieren von Trapezsumme bezüglich h und $\frac{h}{2}$ die Genaugkeit auf h^4 steigern lässt.

Die Annäherungsformel

$$T_1(h) := \frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h) \tag{8.4.3}$$

lässt sich auch wie folgt erklären: Man bestimmt das lineare Interpolationspolynom der Funktion

$$T(\sqrt{x}) = \tau_0 + \tau_1 x + \ldots + \tau_s x^s + \mathcal{O}(x^{s+1}), \quad x > 0$$

zu den Punkten $\left(h^2, T(h)\right)$ und $\left(\left(\frac{h}{2}\right)^2, T\left(\frac{h}{2}\right)\right)$.

$$P\left(T\left(\sqrt{\cdot}\right)\middle| h^2, \left(\frac{h}{2}\right)^2\right)(x) = T(h) + \frac{T\left(\frac{h}{2}\right) - T(h)}{\left(\frac{h}{2}\right)^2 - h^2}(x - h^2)$$

und es gilt $P(h^2) = T(h)$ und $P\left(\left(\frac{h}{2}\right)^2\right) = T\left(\frac{h}{2}\right)$.

Werte P an der Stelle $x = 0$ aus, da wir $T(0)$ extrapolieren wollen:

$$\begin{aligned} P\left(T\left(\sqrt{\cdot}\right)\middle| h^2, \left(\frac{h}{2}\right)^2\right)(0) &= T(h) + \frac{T\left(\frac{h}{2}\right) - T(h)}{\left(\frac{h}{2}\right)^2 - h^2}(0 - h^2) \\ &= T(h) + \left(T\left(\frac{h}{2}\right) - T(h)\right)\frac{4}{3} \\ &= \frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h) = T_1(h) \end{aligned}$$

Führe nun die obige Idee weiter mit der Gleichung 8.4.3. Es gilt nach 8.4.2

$$\begin{aligned} T_1(h) - I(f) &= \hat{\tau}_2 h^4 + \ldots + \hat{\tau}_s h^{2s} + \mathcal{O}(h^{2s+2}) \\ T_1\left(\frac{h}{2}\right) - I(f) &= \hat{\tau}_2 \frac{h^4}{2^4} + \ldots + \hat{\tau}_s \frac{h^{2s}}{2^{2s}} + \mathcal{O}(h^{2s+2}) \end{aligned}$$

Multipliziere die erste Gleichung mit $\frac{1}{15}$ und ziehe sie vom $\frac{16}{15}$-fachen der zweiten Gleichung ab:

$$\begin{aligned} & \frac{16}{15}\left(T_1\left(\frac{h}{2}\right) - I(f)\right) - \frac{1}{15}\left(T_1(h) - I(f)\right) && = \hat{\tau}_3 h^6 + \ldots + \mathcal{O}(h^{2s+2}) \\ \Leftrightarrow \quad & \frac{16}{15}T_1\left(\frac{h}{2}\right) - \frac{1}{15}T_1(h) - I(f) = \hat{\tau}_3 h^6 + \ldots + +\mathcal{O}(h^{2s+2}) \end{aligned}$$

Folglich hat

$$T_2(h) := \frac{16T_1\left(\frac{h}{2}\right) - T_1(h)}{15}$$

einen Fehler der Ordnung $\mathcal{O}(h^6)$.

$T_2(h)$ lässt sich wie $T_1(h)$ über *Extrapolation* erklären, d.h. es gilt

$$P\left(T\left(\sqrt{\cdot}\right)\middle| h^2, \left(\frac{h}{2}\right)^2, \left(\frac{h}{4}\right)^2\right)(0) = T_2(h).$$

$T_2(h)$ ist die Auswertung an $x = 0$ des *quadratischen* Interpolationspolynoms von $T(\sqrt{\cdot})$ an den Stützstellen $h^2, \left(\frac{h}{2}\right)^2, \left(\frac{h}{4}\right)^2$.

Allgemeines Schema: Sei h eine feste Anfangsschrittweite (z.B. $h = b - a$). Definiere

$$\begin{aligned} T_{i,0} &:= T\left(\frac{h}{2^i}\right), \quad i = 0, 1, 2, \dots \\ T_{i,j} &:= \frac{4^j T_{i,j-1} - T_{i-1,j-1}}{4^j - 1}, \quad j = 1, 2, \dots \quad i \geq j \end{aligned} \tag{8.4.4}$$

Diese Rekursion lässt sich in Form des *Romberg-Schemas* (50er Jahre) bildlich darstellen.

$$\begin{array}{lccccccccc} T(h) = T_{0,0} & & & & & & & & & \\ & \searrow & & & & & & & & \\ T\left(\frac{h}{2}\right) = T_{1,0} & \to & T_{1,1} & & & & & & & \\ & \searrow & & \searrow & & & & & & \\ T\left(\frac{h}{4}\right) = T_{2,0} & \to & T_{2,1} & \to & T_{2,2} & & & & & \\ & \searrow & & \searrow & & \searrow & & & & \\ T\left(\frac{h}{8}\right) = T_{3,0} & \to & T_{3,1} & \to & T_{3,2} & \to & T_{3,3} & & & \\ \vdots & & \vdots & & \vdots & & & \ddots & & \\ & \searrow & & \searrow & & \searrow & & & \searrow & \\ T\left(\frac{h}{2^s}\right) = T_{s,0} & \to & T_{s,1} & \to & T_{s,2} & \to & T_{s,3} & \dots & \to & T_{s,s} \end{array}$$

Dieses Schema ist so angelegt, daß der Fehler in der j-ten Spalte von der Ordnung $\mathcal{O}(h^{2j})$ ist. Folglich liefert $T_{s,s}$ eine Approximation für $I(f)$ von der Ordnung $\mathcal{O}(h^{2s})$.

Zur praktischen Durchführung des Extrapolationstableaus 8.4.4

Man muss $T(h), T\left(\frac{h}{2}\right), T\left(\frac{h}{4}\right), \dots, T\left(\frac{h}{2^s}\right)$ berechnen. Greife dabei auf schon berechnete Werte zurück.

Beispielsweise für $h = b - a$ gilt $T(h) = h\left(\frac{f(a)}{2} + \frac{f(b)}{2}\right)$ und es folgt
$T\left(\frac{h}{2}\right) = \frac{h}{2}\left(\frac{f(a)}{2} + f\left(a + \frac{h}{2}\right) + \frac{f(b)}{2}\right)$

Praktisches Problem: Oft weiß man nicht im Vorhinein, wie s zu wählen ist, damit das Integral mit gewünschter Genauigkeit approximiert wird. Anderenfalls könnte man zuerst alle Funktionswerte $f\left(a + \frac{h}{2^s}\right)$ berechnen, die in $T\left(\frac{h}{2^s}\right)$ auftreten, und dann „von fein auf grob arbeiten“. Stattdessen berechnet man einige Spalten des Tableaus und prüft etwa $|T_{i-1,j} - T_{i,j}| \leq \varepsilon c$, wobei c eine grobe Näherung für $\int_a^b |f(x)|\,\mathrm{d}x$ und $\varepsilon > 0$ beliebig klein ist.

Weiterführende Informationen findet man beispielsweise in [DH].

Bemerkung 8.4.5. *Das Rombergschema beruht auf der jeweiligen Halbierung der Schrittweite, also $h_0 = b - a, h_1 = \frac{h_0}{2}, h_2 = \frac{h_0}{4}, \dots$. Diese nennt man die* Rombergfolge.
Es gibt auch andere Folgen, wie beispielsweise $h_0 = b - a, h_1 = \frac{h_0}{2}, h_2 = \frac{h_0}{3}, \dots$, die man Bulirschfolge *nennt.*

Der Vorteil der Bulirschfolge gegenüber der Rombergfolge ist die Tatsache, daß die Rechenarbeit zur Auswertung von f während der Berechnung nicht so schnell ansteigt, man aber den Fehler auch nicht zu stark erhöht.

Bemerkung 8.4.6. *Der Ausgangspunkt für das Extrapolationsscheme war die Existenz einer asymptotischen Entwicklung 8.4.1. Eine Verallgemeinerung auf andere Situationen ist möglich, z.B. mit numerischer Differentiation oder gewissen Diskretisierungsschemata bei gewöhnlichen Differentialgleichungen. Näheres hierzu und eine Fehlerabschätzung findet man in [S] und [DH].*

8.5 Gauß-Quadratur

Bei Newton-Cotes-Formeln (8.2.4) entsprechen sich im Wesentlichen der Grad der Exaktheit (d.h. bis zu diesem Grad werden Polynome exakt integriert) und die Anzahl der Stützstellen ξ_j. Diese wurden willkürlich äquidistant gewählt.

Kann man durch geschickte Wahl der Stützstellen ξ_j und der Gewichte c_j den Exaktheitsgrad erhöhen?
Zunächst hat man bei Newton-Cotes-Formeln $2m+2$ Freiheitsgrade, nämlich $x_0, \ldots, x_m$ Stützstellen und $c_0, \ldots, c_m$ Gewichte für ein Interpolationspolynom vom Grad m und den Exaktheitsgrad m oder $m+1$ auf $[c,d]$.

Sei nun

- $\omega = \omega(x) > 0$ eine feste Gewichtsfunktion auf $[c,d] \subseteq \mathbb{R}$
- $\langle f, g\rangle := \int_c^d \omega(t) f(x) g(x)\,\mathrm{d}x$ ein zugehöriges gewichtetes Skalarprodukt
- $\{P_k\}_{k\geq 0}$ eine Folge von Polynomen, wobei P_k exakt vom Grad k ist, die

$$\langle P_i, P_j\rangle := \int_c^d \omega(x) P_i(x) P_j(x)\,\mathrm{d}x = 0 \quad \text{für } i \neq j \tag{8.5.1}$$

erfüllen (Orthogonalpolynome bzgl. ω über $[c,d]$).

Satz 8.5.2.
Zu beliebigem $m \in \mathbb{N}$ existieren eindeutige, paarweise verschiedene Stützstellen $x_0, \ldots, x_m$ mit $c = x_0 < x_1 < x_2 < \cdots < x_m = d$ und eindeutige Gewichte $\omega_0, \ldots, \omega_m$, sodaß

$$\sum_{j=0}^{m} \omega_j f(x_j) = \int_c^d f(x)\,\mathrm{d}x + E(f) \tag{8.5.3}$$

mit $E(f) = 0$ für alle $f \in \mathbb{P}_{2m+1}$ gilt, das heißt, die Quadratur ist exakt vom Grad $2m+1$.

Genauer gilt: Die x_j sind die Nullstellen des $(m+1)$-ten Orthogonalpolynoms P_{m+1} bzgl. ω und

$$\omega_j := \int_c^d \omega(x) \underbrace{\prod_{\substack{k=0\\k\neq j}}^{m} \frac{x - x_k}{x_j - x_k}}_{\text{Lagrange-Fundamentalpolynome}} \mathrm{d}x > 0.$$

Beachte, dass eine geschickte Wahl der Stützstellen und Gewichte grob eine Verdopplung des Exaktheitsgrades erlaubt.

Beweis: Ein Beweis dieses Satzes findet sich in [S] (Satz 3.5.12). ∎

Erinnere nun aus Beispiel 8.1.1 und (8.2.1) an die

Eigenschaften von Orthogonalpolynomen

(i) Zu jedem Skalarprodukt mit Gewichtsfunktion ω bzgl. $[c,d]$ gibt es eindeutig bestimmte Orthogonalpolynome mit führendem Koeffizienten 1 (d.h. $P_k(t) = 1 \cdot t^k + \ldots$).

(ii) Das Orthogonalpolynom P_k (vom Grad exakt k) hat genau k einfache Nullstellen $c < x_i < d, i = 0, \ldots, k-1$. Daraus folgt $P_k(x) = 1 \cdot \prod_{i=0}^{k-1} (x - x_i)$.

Berechnung von Orthogonalpolynomen

Zur Erinnerung: $\mathbb{P}_N$, die Menge der Polynome von maximalem Grad N auf $[c,d]$, besitzt eine Basis $u_i(x) := x^i, i = 0, \ldots, N$. Setze $\|\cdot\|^2 := \langle \cdot, \cdot \rangle$. Der Algorithmus für das (Gram-)Schmidt-Orthonormalisierungsverfahren zur Berechnung von Orthogonalpolynomen ist:

$$
\begin{aligned}
v_0(x) &:= \frac{u_0(x)}{\|u_0(x)\|}, \text{ für } i = 1, \ldots, N, \\
\widetilde{v_i}(x) &:= u_i(x) - \sum_{k=0}^{i-1} \langle v_k, u_k \rangle u_k(x), \\
v_i(x) &:= \frac{\widetilde{v_i}(x)}{\|\widetilde{v_i}(x)\|}.
\end{aligned}
$$

Motivation hinter der Gauß-Quadratur

Die Gauß-Quadratur mithilfe von Orthogonalpolynomen spielt in der Stochastik eine große Rolle. Dies verdeutlichen wir anhand eines Beispiels aus der Finanzmathematik:

Beispiel (Unsichere Auszahlung)**.** *Sei eine unsichere Auszahlung X als Zufallsvariable modelliert. Häufig wird eine Normalverteilung mit Erwartungswert 0 und Varianz 1 angenommen, die wir als*

$$X \sim \mathcal{N}(0,1)$$

notieren.
Für die Approximation des Erwartungswerts

$$\mathbb{E}[X] := \int_{\mathbb{R}} t\omega(t)\,\mathrm{d}t$$

mit Dichte

$$\omega(x) := \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

eignet sich die Gauß-Quadratur. Denn als Gewichtsfunktion kann die Dichte $\omega(x)$ herangezogen werden.
Die entsprechenden Orthogonalpolynome $\{P_k\}_{k \in \mathbb{N}_0}$, welche orthogonal bezüglich des gewichteten Skalarproduktes

$$\langle f, g \rangle := \int_{\mathbb{R}} \omega(x) f(x) g(x)\,\mathrm{d}x$$

sind, sind die Hermitepolynome $H_0(x) = 1$, $H_1(x) = x$, $H_2(x) = x^2 - 1$, ... mit

$$\langle H_i(x), H_j(x) \rangle = \sqrt{2\pi} j! \delta_{ij}$$

Analog lassen sich höhere Momente

$$\mathbb{E}[X^k] := \int_{\mathbb{R}} x^k \omega(x)\,\mathrm{d}x$$

oder die Varianz

$$\sigma^2(X) := \int_{\mathbb{R}} (x - \mathbb{E}[X])^2 \omega(x)\,\mathrm{d}x$$

effizient numerisch approximieren.

Entsprechende Orthogonalpolynome, Gewichtsfunktionen und Werte ihrer Skalarprodukte sind für die gängigen Wahrscheinlichkeitsverteilungen bekannt. Sie sind in dem sogenannten Askey-Schema zusammengefasst [AW].

Die folgende Tabelle bietet einen Überblick der wichtigsten Fälle (mit $\|\cdot\| := \langle\cdot,\cdot\rangle$ aus (8.5.1))

Verteilung	Polynome	Gewicht $\omega(x)$	$\|P_j\|$	Träger $[c,d]$
Gauß–	Hermite	$\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$	$\sqrt{2\pi} j!$	$(-\infty,\infty)$
Gamma–	Laguerre	$x^\alpha e^{-x}$	$\frac{\Gamma(j+\alpha+1)}{n!}\delta_{ij}$	$[0,\infty)$
Beta–	Jacobi	$(1-x)^\alpha(1+x)^\beta$	$\dots$	$[-1,1]$
Gleich–	Legendre	1	$\frac{2}{2j+1}$	$[-1,1]$

Bemerkung.

- Durch geeignete Skalierung kann ein anderer, gewünschter Träger erzielt werden.
- Die Exponentialverteilung ist ein Spezialfall der Gammaverteilung mit $\alpha = 0$.
- Für diskrete Verteilungen, z.B. Poisson– und Binomialverteilungen, sind ebenfalls (diskrete) Orthogonalpolynome bekannt, die jedoch selten Verwendung finden.
- In der Physik werden eher Hermitepolynome verwendet, die orthogonal bezüglich e^{-x^2} sind.

Bemerkung. *Für die hochdimensionale Integration mancher Probleme aus der Finanzmathematik wie etwa in Beispiel 1.1 benötigen wir für die Berechnung von Erwartungswerten mit normalverteilten Zufallsvariablen das Gewicht $\omega(x) = \exp(-\frac{x^2}{2})$ als Dichte der Normalverteilung. Diesen Fall bezeichnet man ebenfalls in diesem Zusammenhang als Gauß-Hermite-Quadratur, die zugehörigen Quadraturformeln bezeichnet man gemäß Satz 8.5.2 dann z.B. als Gauß-Hermite-Quadraturformeln.*

Zwei Spezialfälle von Orthogonalpolynomen

Legendre-Polynom:

$$P_0(x) = 1,\ P_1(x) = x,\ P_2(x) = \frac{1}{2}(3x^2-1),\ P_3(x) = \frac{1}{2}(5x^3-3x),\ P_4(x) = \frac{1}{8}(35x^4-30x^2+3).$$

Hermite-Polynome:

$$H_0(x) = 1,\ H_1(x) = 2\alpha_1 x,\ H_2(x) = \alpha_2(4x^2-2),\ \dots$$

wobei $\alpha_j := 2^{-\frac{j}{2}}(j!)^{-\frac{1}{2}}\pi^{-\frac{1}{4}}$.

8.6 Schwierige Integranden

Bei den bisher diskutierten Methoden zur Berechnung von

$$\int_a^b f(x)\,\mathrm{d}x$$

treten Probleme auf, wenn der Integrand folgendermaßen ist:

- *f unstetig*: Wenn die Unstetigkeitsstellen bekannt sind, unterteile das Integral dort. Wenn nicht und wenn diese nicht mitbehandelt werden, kommt es zu keiner Konvergenz oder falschen Ergebnissen. Hier bieten sich *adaptive Quadraturverfahren* an, die in [DH] behandelt werden.
- *f Nadelimpuls, „cusp“*: Hierbei versagt jedes Quadraturprogramm, sofern die Spitze genügend schmal ist.
- *f schwach singulär*, d.h., $f^{(k)}$ existiert nicht für ein oder mehrere k. Dies gilt z.B. für $f(t) = t^{-\alpha} g(t)$ mit $\alpha > -1$, $\alpha \neq 0$, $g(t) \in \mathcal{C}^\infty$.

  Beispiel. *Betrachte das Integral*

  $$\int_0^\pi \sqrt{t}\cos(t)\,\mathrm{d}t.$$

  Die Ableitung $f'(t) = \frac{\cos(t)}{2\sqrt{t}} + \sqrt{t}(-\sin(t))$ hat eine Polstelle bei $t = 0$.

  Bei solchen Integranden werden adaptive Quadraturverfahren extrem langsam; nichtadaptive liefern falsche Ergebnisse. Manchmal ist es möglich, die Singularität durch Substitution zu beseitigen:

  $$s = \sqrt{t} \Rightarrow s^2 = t, \quad \frac{\mathrm{d}t}{\mathrm{d}s} = 2s, \quad \int_0^\pi \sqrt{t}\cos(t)\,\mathrm{d}t = \int_0^{\sqrt{\pi}} s\cos(s^2)(2s)\,\mathrm{d}s.$$

- Faustregel: Weiß man nicht, ob f glatt ist, muss die Quadraturmethode sorgfältig gewählt werden.

8.7 Zweidimensionale Integration

Quadraturformeln auf dem Einheitsquadrat $[0,1]^2$, bzw. dem Einheitssimplex $\Delta := \left\{x_1, x_2 \in [0,1] \,\middle|\, x_1 + x_2 \leq 1\right\} \subset [0,1]^2$, lassen sich mittels affiner Transformationen auf beliebige Rechtecke bzw. Dreiecke anwenden, da affine Transformationen den Grad der Exaktheit erhalten.

8.7.1 Integration über das Einheitsquadrat

Zur Berechnung von

$$I(f) := \int_0^1 \int_0^1 f(x,y)\,\mathrm{d}x\mathrm{d}y$$

kann man in jede Richtung die Newton-Cotes-Formeln einsetzen (Tensorproduktansatz) mit $[c,d] = [0,1]$, $h = d - c = 1$, $x_i = \frac{i}{m}$, $y_j = \frac{j}{m}$, $i = 0,\ldots,m$, $j = 0,\ldots,m$ und folglich

$$I(f) \approx h^2 \sum_{i=0}^m \sum_{j=0}^m c_i c_j f(x_i, y_j) =: I_m.$$

Man kann dann zeigen, daß I_m exakt für alle Polynome $p \in \operatorname{span}\left\{x^{k_1} y^{k_2} \middle| 0 \leq k_1, k_2 \leq m\right\}$ ist.

8.7.2 Integration über das Einheitsdreieck

Diese Quadraturformeln sind besonders bei der Lösung partieller Differentialgleichungen mit finiten Elementen (stückweisen Polynomen auf Dreiecksgittern) von Bedeutung.

Aufgabe. *Finde Quadraturformeln, sodaß auf Δ alle Monome der Form $x^{k_1}y^{k_2}$ mit $0 \leq k_1, k_2$ und $k_1 + k_2 \leq m$ exakt integriert werden.*

Beispiel. *Quadraturformeln mit Exaktheitsgrad 1:*

1. $Q(f) = \frac{1}{2} f\left(\frac{1}{3}, \frac{1}{3}\right)$
2. $Q(f) = \frac{1}{6}\left[f(0,0) + f(1,0) + f(0,1)\right]$

Quadraturformeln mit Exaktheitsgrad 2:

1. $Q(f) = \frac{1}{6}\left[f\left(\frac{1}{2}, 0\right) + f\left(0, \frac{1}{2}\right) + f\left(\frac{1}{2}, \frac{1}{2}\right)\right]$
2. $Q(f) = \frac{1}{6}\left[f\left(\frac{1}{6}, \frac{1}{6}\right) + f\left(\frac{2}{3}, \frac{1}{6}\right) + f\left(\frac{1}{6}, \frac{2}{3}\right)\right]$

8.8 Hochdimensionale Integration

Aufgabe 8.8.1 (Finanznumerik)**.** *Numerische Berechnung von*

$$\int_{[0,1]^d} f(x)\,\mathrm{d}x = \underbrace{\int_0^1 \cdots \int_0^1}_{d\text{-mal}} f(x_1, \ldots, x_d)\,\mathrm{d}x_1 \ldots \mathrm{d}x_d$$

mit $f : [0,1]^d \to \mathbb{R}$ als gegebene multivariate Funktion und Raumdimension $d \gg 1$.

Motivation*: Problemstellungen in der Finanznumerik*

- *Mortgage-Backed Securities (MBS, durch Hypotheken gesicherte Wertpapiere) typischerweise mit $d = 256$*
- *Bewertungen pfadabhängiger Optionen typischerweise mit $d = 2^{10}$, wobei d die Anzahl der Zeitschritte ist.*

Idee: Wende für $d > 1$ die eindimensionale Quadraturformel in jeder Raumdimension an. Auswertung des Integranden z.B. mit der Trapezregel an $N = 2^d$ Stellen.

Problem: Für $d \gg 1$ (z.B. $d = 256$) ist $N = 2^{256} \approx 1.2 \cdot 10^{77}$ und damit das Integral praktisch unmöglich zu berechnen.

Wichtig für die Güte der Approximation ist die Kenntnis über Quadraturfehler.

Klassische Quadraturformeln wie die Newton-Cotes-Formeln aus den Abschnitten 8.1 und 8.2 mit Polynomen vom Grad $r - 1$ in jeder Raumdimension liefern die Fehlerabschätzung

$$\left| \int_{[0,1]}^d f(x)\,\mathrm{d}x - \frac{1}{N} \sum_{i=1}^{N} a_i f(z_i) \right| \lesssim N^{-r/d} \tag{8.8.2}$$

(Trapezregel: Für $d = 1$ ist $r = 2$, also folglich der Fehler $\lesssim N^{-2}$)

Problem: Diese Fehlerabschätzung (8.8.2) ist unbrauchbar für $d \geq 1$ (außer bei der Wahl $r = d$, was allerdings für große d eine Approximation mit einem stark oszillierenden Polynom bedeuten würde).

Alternative: Monte-Carlo-Methoden (MC-Methoden)
Verwende anstelle fest gewählter Stützstellen Zufallszahlen z_i wie in (8.8.2) und approximiere

$$\int_0^1 f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} f(z_i)$$

Die Fehlerabschätzung wird unabhängig von der Raumdimension d und es gilt:

$$\left|\int_{[0,1]^d} f(x)\,\mathrm{d}x - \frac{1}{N}\sum_{i=1}^{N} f(z_i)\right| \lesssim N^{-1/2}$$

für eine wachsende Anzahl N von Stützstellen.
Die Fehlerreduktionsrate ist jetzt zwar konstant, aber mit $\frac{1}{2}$ sehr langsam.

Für weiterführende Literatur über die vor allem in den letzten Jahren entwickelten sogenannten Quasi-Monte-Carlo-Methoden und Multilevel-Monte-Carlo-Methoden seien [DKS] und [G] empfohlen.

Notationsverzeichnis

Allgemein

$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{C}$	Menge der komplexen Zahlen
$\|\cdot\|$	Beliebige Norm auf $\mathbb{R}^n$ oder Operatornorm für Matrizen, siehe (2.1.15a)
$\|\cdot\|_\infty$	Unendlichnorm, siehe (2.1.6), oder Maximumsnorm für Matrizen, siehe 2.1.15b
$\|\cdot\|_1$	siehe 2.1.15b
$\|\cdot\|_2$	siehe 2.1.15b
$\delta_{i,k}$	Kronecker-Delta
$\mathcal{C}^k(\mathbb{K})$	Menge der k-mal stetig differenzierbaren Funktionen auf $\mathbb{K}$
$\mathbb{O}_n(\mathbb{R})$	Menge der Orthogonalmatrizen, siehe (3.7.3a)
$\mathcal{O}(\cdot)$	Landau-Symbol, siehe Definition (1.4)
$o(\cdot)$	Landau-Symbol, siehe Definition (1.5)
$\doteq$	In 1. Näherung, siehe (2.1.4)
∇f	Gradient von f, siehe (2.1.9)
$Df(x)$	JACOBI–Matrix, siehe (2.1.15)

Kapitel 2

κ_{rel}	Relative Kondition, siehe (2.1.6)
κ_{abs}	Absolute Kondition, siehe (2.1.6a)
$\kappa(A)$	Kondition der Matrix A, siehe (2.1.17)
$E_\delta(x)$	Menge der Eingabedaten in einer δ–Umgebung von x, siehe (2.1.10a)
$L_{\text{rel}}(\delta)$	Lipschitz-Konstante der Kondition, siehe (2.1.12)
$\mathrm{fl}(x)$	Reduktionsabbildung, siehe (2.2.3)
$\mathbb{M}(b, m, e_{\min}, e_{\max})$	Menge der im Rechner darstellbaren Gleitkommazahlen, siehe (2.2.2a)
eps	Relative Maschinengenauigkeit, siehe (2.2.4)
ε	Absolute Maschinengenauigkeit, siehe (2.2.6)

Kapitel 3

Q_v	HOUSEHOLDER–Transformation, siehe (3.7.5)
$G_{i,k}$	GIVENS–Rotation, siehe (3.7.22)
φ_{ik}	Codierung einer GIVENS–Rotation, siehe (3.7.32)

Kapitel 4

$\perp_{\langle\cdot,\cdot\rangle}$	Orthogonal bezüglich Skalarprodukt $\langle\cdot,\cdot\rangle$, siehe (4.3.5)
A^+	Pseudoinverse von A, siehe (4.4.5)

Kapitel 6

x_i	Stützstelle, siehe (6.1.2)
$\mathcal{F}'$	Menge der linearen Funktionale auf einem Funktionenraum $\mathcal{F}$, siehe (6.2.1)
V_n	VANDERMONDE–Matrix, siehe (6.2.6)
$\mathcal{P}_n$	Raum der Polynome vom Grad n, siehe (6.2.5)
$[x_0, ..., x_n]f$	Dividierte Differenzen der Ordnung n von f, siehe (6.3.14)

Kapitel 7

Π_k	Polynome der Ordnung höchstens k, entspricht $\mathcal{P}_{k-1}$
$\Delta = \{\tau_i\}_{i=0}^{\ell+1}$	Gitter mit $\ell + 2$ paarweise verschiedenen Knoten τ_i, siehe (7.2.1)
$\mathcal{S}_{k,\Delta}$	Splineraum, siehe (7.2.2)
$(x - \tau_i)_+^{k-1}$	Abgebrochene Potenz, siehe (7.2.3a)
$\chi_{[\tau_i,\tau_{i+1})}(x)$	Charakteristische Funktion auf $[\tau_i, \tau_{i+1})$
$N_{i,k}(x)$	B-Splines der Ordnung k, siehe (7.3.3)
$T = \{\theta_i\}_{i=1}^{n+k}$	Gitter mit erweiterten Randbedingungen, siehe (7.3.9)
$\mathcal{N}_k(T) = \mathcal{N}_{k,T}$	Lineares Erzeugnis der B-Splines auf T, siehe (7.3.10)
$c_i^{[r]}$	B-Spline-Koeffizienten, siehe (7.3.13)

Kapitel 8

$I(f)$	Integral über f, siehe (8.0.1)
$P(f\|x_0, \ldots, x_m)(x)$	Interpolationspolynom vom Grad m, siehe (8.2.1)
$\mathbb{P}_m$	Polynome vom Grad m, siehe (8.2.1)
$I_m(f)$	Approximation an $I(f)$ vom Grad m, siehe (8.2.1)
$B_k(x)$	Bernoullipolynome, siehe 8.3.1
$B_k := B_k(0)$	Bernoullizahlen, siehe 8.3.1
$T(h)$	Trapezsumme mit Schrittweite h, siehe (8.3.5)
$\omega(x)$	Gewichtsfunktion, siehe (8.5.1)
$Q(f)$	Quadraturformel für f, siehe 8.7.2

Literatur

[AW] R. Askey, J.a. Wilson, *Some Basic Hypergeometric Orthogonal Polynomials that Generalize Jacobi Polynomials*, American Mathematical Society, 319th Edition, Providence, 1985

[Bo] C. de Boor, *A Practical Guide to Splines*, Revised Ed., Springer, Berlin et al., 2011

[CMO] R. Caflisch, W. Morokoff, A. Owen, *Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension*, J. Comput. Finance 1, S.27 – 46, 1997

[DH] P. Deuflhard, A. Hohmann, *Numerische Mathematik 1. Eine algorithmisch orientierte Einführung*, de Gruyter Lehrbuch, Berlin et al., 4. Auflage, 2008

[DR] W. Dahmen, A. Reusken, *Numerik für Ingenieure und Naturwissenschaftler*, Springer, Berlin et al., 2. Auflage, 2008

[DKS] J. Dick, F. Y. Kuo, I. H. Sloan, *High-dimensional integration — The quasi-Monte Carlo way*, Acta Numerica 22, S.133 – 288, doi: 10.1017/S0962492913000044, 2013

[FH-SB] R.W. Freund, R.H.W. Hoppe, Stoer/Bulirsch, *Numerische Mathematik 1*, Springer, Berlin et al., 10. Auflage, 2007

[G] M. Giles, *Multilevel Monte Carlo methods*, Acta Numerica 2015, 2015
Matlab codes unter http://people.maths.ox.ac.uk/ gilesm/acta/

[GJ] M. Günther, A. Jüngel, *Finanzderivate mit MATLAB*, Springer, Berlin et al., 2010

[GL] G.H. Golub, C.F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore et al., 3rd Edition, 1996

[H] M. Hanke-Bourgois, *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, Teubner, Stuttgart et al., 1. Auflage, 2002

[HH] G. Hämmerlin, K.-H. Hoffmann, *Numerische Mathematik*, Springer, Berlin et al.,4. Auflage, 1994

[KR] B. Kernigham, D. Ritchie, *Programmieren in C*, Carl Hanser Verlag, München et al., 2. Ausgabe, 1990

[OR] J. M. Ortega, W. C. Rheinholdt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press., 1970

[PTVF] W. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes. The Art of Scientific Computing*, Cambridge University Press, 3rd Edition, 2007

[S] J. Stoer, *Numerische Mathematik 1*, Springer, Berlin et al., 8. Auflage, 1999

[SB] J. Stoer, R. Bulirsch, *Numerische Mathematik 2*, Springer, Berlin et al., 4. Auflage, 2000

[SB2] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, Springer, Berlin et al., 2nd Edition, 1993

[Sch] T. Schmidtmann, *Moderne Methoden der Zeitreihenanalyse mit Anwendungen auf Daten aus dem Finanzwesen*, Bachelorarbeit, Mathematisches Institut, Universität zu Köln., 2014

[Sey] R.U. Seydel, *Tools for Computational Finance*, Springer, Berlin et al., 4th Edition, 2009

[T] E. Zeidler (Ed.) et al., *Teubner Taschenbuch der Mathematik*, Teubner, Stuttgart et al., 1996

[W] J. H. Wikinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford., 1965