Praktische Mathematik I

ausgearbeitet von

Sandra Görke und Simon Jörres

nach einer Vorlesung von

Prof. Dr. Angela Kunoth

im Wintersemester 2002/2003
an der
Rheinischen Friedrich–Wilhelms–Universität Bonn

Prof. Dr. Angela Kunoth
Institut für Angewandte Mathematik
Universität Bonn
Wegelerstr. 6
53115 Bonn

Tel: (0228) 73–3144
Fax: (0228) 73–7527

kunoth@iam.uni-bonn.de
www.iam.uni-bonn.de/∼kunoth

Dieses Dokument wurde mit LaTeX 2_ε gesetzt. LaTeX ist eine freie Typesetting–Software und kann beispielsweise über

`http://www.ctan.org`

bezogen werden. Einige Grafiken wurden mit dem TeX–Makro PiCTeX erzeugt, welches ebenfalls dort erhältlich ist. Dieses Dokument ist in elektronischer Form im PostScript– (ps) sowie im Portable–Document–Format (pdf) erhältlich und kann vom Server des Instituts für Angewandte Mathematik der Universität Bonn

`http://www.iam.uni-bonn.de`

Abteilung Funktionalanalysis und Numerische Mathematik, unter der Adresse

`http://www.iam.uni-bonn.de/∼kunoth/courses.html`

heruntergeladen werden.

Dieses Dokument wird voraussichtlich bis Ende September 2003 weitergepflegt. Fehler und Anmerkungen werden in dieser Zeit gerne unter der e–Mail–Anschrift

`sjoerres@uni-bonn.de` (Simon Jörres)

entgegengenommen.

Inhaltsverzeichnis

Inhalte der Vorlesung

Der Terminus *Praktische Mathematik* ist ein Synonym für *Numerische Mathematik* oder *Numerik*.

Die Ziele der Numerischen Mathematik sind die Konstruktion und das mathematische Verständnis von Algorithmen zur konkreten (zahlenmäßigen) 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 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 *gewöhnlicher* oder *partieller Differentialgleichungen* gebildet.

Bei *Wettervorhersagen* ist es nötig, enorm große *Datenmengen* und *komplizierte Strömungsvorgänge* auf sehr unterschiedlichen Längenskalen zu verarbeiten. Man sucht auch hier approximative Lösungen von gewöhnlichen oder partiellen Differentialgleichungen, was letztendlich wiederum die Lösung von Gleichungssystemen notwendig macht.

Ein ganz anderer Bereich, in dem die Numerik zum Tragen kommt, sind bildgebende Verfahren. Auch hier ist die Verarbeitung und Analyse großer Datenmengen erforderlich. In der *medizinischen Diagnostik*, dort 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 z.B. die Lösung von linearen Gleichungssystemen oder die Berechnung von Eigenwerten behandelt.

Hier werden folgende Inhalte diskutiert:

- **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 das Cholesky–Verfahren für symmetrisch positiv definite Matrizen sowie Bandstrukturen von Matrizen diskutiert.

- **Ausgleichsrechnung (Least–Squares–Approximation):** In diesem Kapitel werden die Verfahren aus dem vorherigen Abschnitt der Vorlesung angewandt, um 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:** Nach den Sätzen den Linearen Algebra sind die Eigenwerte einer Matrix die Nullstellen des charakteristischen Polynoms. Nullstellenberechnung ist aber aus numerischer Sicht ein schlecht konditioniertes Problem. Nach der Diskussion theoretischer Grundlagen und Methoden zu Eigenwertabschätzungen werden daher als numerisch geeignetere Verfahren zwei Iterations– (Power–Iteration und Inverse Power–Iteration) sowie ein auf der QR–Zerlegung basierendes Verfahren vorgestellt.

- **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 in Kapitel 4 für lineare Ausgleichsprobleme gewonnenen Erkenntnisse werden hier derart erweitert, daß sie auf nichtlineare Gleichungssysteme angewandt werden können.

- **Interpolation mit Polynomen:** Bei der Interpolation will man im Gegensatz zur Approximation wie in Kapitel 4 und 7 gegebene (Meß)werte mit Funktionen nicht nur angenähert, sondern exakt reproduziert haben. Die für uns wichtigsten Verfahren sind die Hermite– und 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:** Anstelle der Interpolation mit Polynomen ermöglicht die Interpolation auf der Basis von Splines (stückweise Polynome) zusätzliche Glattheitseigenschaften des Interpolanden. Zentral wird die Darstellung von Splines über B–Spline–Basen sein.
- **Iterative Lösung linearer Gleichungssysteme:** Die direkten Lösungsverfahren aus Kapitel 3 sind in vielen Fällen numerisch zu aufwendig, da die zu lösenden Systeme zu groß oder nur sehr dünn besetzt sind. In solchen Fällen bieten sich iterative Verfahren an. Neben den klassischen Gesamt– und Einzelschrittverfahren sowie der Relaxation werden das Gradienten– und das darauf aufbauende cg–Verfahren besprochen.
- **Numerische Integration:** Zunächst werden klassische Verfahren wie etwa die Trapezregel behandelt. Aus diesen Prinzipien werden die Newton–Cotes–Formeln hergeleitet. Diese sind die Grundlage für die Euler–MacLaurin'sche Summenformel, welche zur Herleitung des Extrapolationsverfahrens benötigt wird. Die Extrapolation ist ein allgemeines Verfahren zur Erhöhung der Genauigkeit. Die Gauss–Quadratur beruht auf gezielter Wahl der Stützstellen, wodurch die Genauigkeit wiederum erhöht wird. Das Kapitel schließt mit kurzen Abhandlungen über schwierige Integranden und zweidimensionale Integration.

Der zweite Vorlesungsteil im Sommersemester wird schwerpunktmäßig die Numerik gewöhnlicher Differentialgleichungen zum Inhalt haben. Im Hauptstudium wird diese Vorlesungsreihe dann durch Vorlesungen über Numerik partieller Differentialgleichungen bzw. Wissenschaftliches Rechnen 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 NumerikerInnen nicht unbedingt zuzutreffen. Vielmehr gilt die Faustregel aus [T], p. 1093:

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

Abschließend seien noch ein paar Bemerkungen zu vorlesungsbegleitender Literatur gegeben. Moderne Lehrbücher der Numerik sind z.B. [DH] und [H], wobei letzteres sehr ausführlich und thematisch umfassend geschrieben ist. 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 auch viele weitere Literaturhinweise und eine große Zahl von praktisch durchgerechneten Beispielen. 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 folgen.

1 Einführung

Das Anforderungsprofil an einen Mathematiker in der Numerik bildet heute die Schnittstelle zwischen der Mathematik und einigen anderen angewandten Wissenschaften. Es ist keine Seltenheit mehr, 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 Vorkommen eines derartigen Modellierungsfehlers selbst in heutiger Zeit ist die *Londoner Millennium–Bridge*, die nach nicht einmal zwei Jahren nach ihrer ersten Öffnung 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 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 *Finanz–Derivaten* wichtig.

1.1 Zwei Beispiele

Nachdem einige Anwendungsbeispiele grob angeschnitten wurden, werden nun zwei Beispiele für numerisches Arbeiten etwas ausführlicher beleuchtet, in denen die Schnittpunkte der Mathematik mit anderen angewandten Wissenschaften konkret dargestellt werden.

Beispiel 1.1.1 (Braunkohleförderung) Bei der Braunkohleförderung muß, bevor man mit der eigentlichen Förderung des Rohstoffvorkommens beginnen kann, ein enormer Abraum von verschiedenen Erdschichten beseitigt werden. Da dies mit hohen Kosten verbunden ist, und ein ökonomischer Abtransport stattfinden soll, braucht man genaue Informationen über das Volumen des Abraums. Um dieses zu berechnen, kommen numerische Methoden zur Anwendung. Dies geschieht in einem Prozeß aus mehreren Schritten.

In einem ersten Schritt findet aus Sicht der Numerik eine Art *Preprocessing* statt, in dem Messungen und Experimente zur Tiefenmessung stattfinden. Hierzu werden z.B. Stereofotoaufnahmen von Flugzeugen aus durchgeführt. Dies sind Verfahren der Photogrammetrie. Um nun numerische Methoden zur Berechnung anstellen zu können, braucht man ein *mathematisches Modell* des Abraums. Dieses wird in einem zweiten Schritt entworfen, indem man z.B. das Volumenintegral

$$\int_V f(x)dx$$

berechnet. Damit ist natürlich auch eine Überprüfung der Existenz und Eindeutigkeit der Lösung dieses Integralwerts verbunden. Ist dies nicht der Fall, wird man das Verfahren modifizieren

müssen. Im nun folgenden dritten Schritt wird ein *konstruktiver numerischer Ansatz* entworfen. Hier sucht man Quadraturformeln zur approximativen Berechnung des Integrals. Diese können etwa von der Gestalt

$$\int_V f(x)dx \approx \sum_{i=1}^{n} a_i f(x_i)$$

sein. Jetzt kommt in Schritt vier die *Realisierung des Algorithmus* auf dem Rechner verbunden mit dessen Programmierung. Hiernach findet im fünften Schritt ein *Postprocessing* als Validierung der Ergebnisse und deren Visualisierung statt. Dabei könnte es sich herausstellen, daß sich die Ergebnisse als für die Praxis ungeeignet erweisen. In einem solch Fall muß man den ganzen Prozeß von vorne beginnen und bei Schritt eins oder zwei wieder einsteigen. ∎

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.

Beispiel 1.1.2 (Optionsbewertung) Als 1973 der Artikel [BS] erschien, in dem eine Formel für die Berechnung des fairen Preises einer Option hergeleitet wurde, war dies für die *Finanzmathematik* ein großer Fortschritt. Basierend auf der BROWN'schen Molekularbewegung, wurde die als BLACK–SCHOLES–Gleichung bekannte partielle Differentialgleichung

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{1.1.3}$$

formuliert, in der $V(S,t)$ für den Wert einer Europäischen Option zur Zeit t, $S(t)$ für den Kurs des der Option zugrundeliegenden Basiswertes, $r > 0$ für den risikofreien Zinssatz und σ für die Volatilität des Kurses S stehen. Für die BLACK–SCHOLES–Gleichung in der Form (1.1.3) ist eine *analytische Lösung* bekannt. Will man jedoch allgemeinere Modelle betrachten, in denen *Transaktionskosten* (z.B. Gebühren oder Steuern) in Höhe von k auftreten, muß man der Gleichung (1.1.3) noch den Term

$$-\sqrt{\frac{2}{\pi}}\frac{k\sigma S^2}{\sqrt{\sigma t}}\left|\frac{\partial^2 V}{\partial S^2}\right|$$

hinzufügen. In diesem Fall ist die Gleichung im allgemeinen nur noch *numerisch* lösbar. Im folgenden soll ein anderer, etwas einfacherer Algorithmus zu Preisberechnung von Optionen vorgestellt werden, dessen Ergebnisse den Werten von BLACK–SCHOLES im wesentlichen entsprechen. Dazu stellen wir zunächst einige Eigenschaften von Optionsscheinen zusammen, die bei der Herleitung des Algorithmus behilflich sein werden. Unsere Darstellung ist an [Se] angelehnt.

Unter einer Option versteht man das *Recht*, eine bestimmte Menge eines bestimmten *Wirtschaftsguts* zu einem *im voraus* festgesetzten Preis *kaufen* (Call–Option) oder *verkaufen* (Put–Option) zu dürfen. Darf dieses Recht, nur zu einem *bestimmten Zeitpunkt* ausgeübt werden, spricht man von einer *Europäischen Option*. Ist die Ausübung des durch die Option verbrieften Rechts in einer bestimmten Frist möglich, so heißt der Vertrag *Amerikanische Option*. Diese Unterscheidung hat allerdings mit den geographischen Bezeichnungen nichts zu tun. Sei T das Laufzeitende eines Optionsvertrages. Dann hat der Inhaber einer Option folgende Handlungsmöglichkeiten:

- die Option am Markt zum Tagespreis verkaufen ($t < T$)
- abwarten
- die Option ausüben ($t \leq T$)
- die Option verfallen lassen ($t \geq T$).

Im letzten Fall, wo das Optionsrecht ungenutzt bleibt, kann man den für die Option gezahlten Preis als eine Art *Versicherungsprämie* interpretieren.

Im folgenden wird der Wert eines Optionsvertrages zum Fälligkeitszeitpunkt T betrachtet (zugrundeliegendes Wertpapier sei eine Aktie). Wir beginnen mit einer *Call–Option*. Sei $0 \leq t \leq T$ die Zeitachse, $S(t)$ der Kurs des zugrundeliegenden Wertpapiers und E der Basispreis. Der Halter wird die Option zum Vertragsende T nur ausüben, falls

$$S(T) > E$$

gilt. Damit gilt für den Wert $V(S(T), T)$ der Option

$$V(S,T) = \begin{cases} 0 & \text{falls } S(T) \leq E \\ S(T) - E & \text{falls } S(T) > E \end{cases}$$

oder

$$V(S,T) = \max\{S(T) - E, 0\} =: (S(T) - E)^+.$$

Die *Put–Option* stellt das Gegenstück zum Call dar. Der Halter einer Put–Option wird in T die Option nur ausüben, falls

$$S(T) < E$$

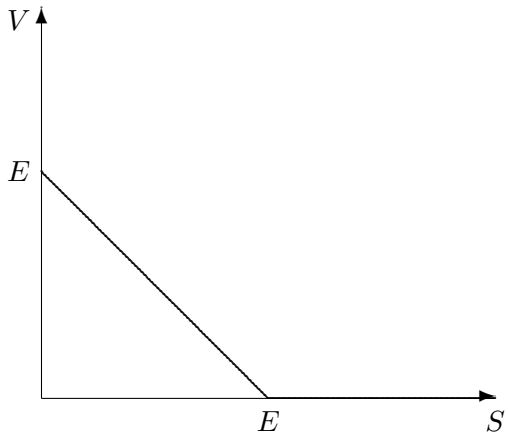
ist. Damit ist der Wert $V(S(T), T)$ eines Put bestimmt durch

$$V(S,T) = \begin{cases} 0 & \text{falls } S(T) \geq E \\ E - S(T) & \text{falls } S(T) < E \end{cases}$$

oder

$$V(S,T) = \max\{E - S, 0\} =: (E - S)^+.$$

Rechts ist der Wert eines Puts in Abhängigkeit vom Kurs des zugrundeliegenden Basiswerts zur Zeit t, wobei E für den Ausübungspreis steht, aufgetragen. Ist der Wert V eines Optionsscheins positiv, so sagt man auch die Option sei *in the money*. Wenn der Wert des Basispapiers gerade den Ausübungspreis hat, so ist die Option *at the money* und sonst im Fall $S < E$ ist ein Call *out of the money*, was ein Put im Fall $E < S$ ist. Bezieht man in ein Schema wie das nebenstehende noch die Zeit t, so ergibt sich eine gekrümmte Fläche, auf der die Optionspreise liegen.



Um nun das mathematische Modell zu formulieren, nehmen wir idealisierend an, daß alle Variablen Werte in $\mathbb{R}$ annehmen. Dies ist praktisch, da man sich so keine Einschränkungen auferlegt, trifft allerdings in der Realität nicht zu, da das wirkliche Verhalten von Optionen eher diskret ist. Wir möchten den in vielen praktischen Fällen wichtigen Wert $V(S(0),0)$, also den Wert V eines europäischen Puts zum „heutigen“ ($t = 0$) Kurs $S(0)$ berechnen. Dann ist es nicht nötig, die oben beschriebene Fläche vollständig zu berechnen, sondern es genügt, ein baumartiges Gitter zu entwerfen. Dieses werden wir als *Binomialbaum* bezeichnen. Wir ersetzen zunächst die kontinuierliche Zeit t durch diskrete Zeitpunkte t_i, indem wir eine Anzahl M von Zeitschritten wählen und $\Delta t := \frac{T}{M}$ setzen. Dann gilt $t_i = i \cdot \Delta t$ für $i = 0, \ldots, M$. Es bezeichne noch $S_i := S(t_i)$. Um einen adäquaten Wert für den Optionspreis zu erhalten, der natürlich vom Zufall abhängt, wird es entscheidend sein, wie der Zufall im Rechner simuliert wird. Dazu machen wir folgende Annahmen

(A1) Ein Kurs S kann sich nach Ablauf von Δt nur zu zwei Kurswerten entwickeln, entweder aufwärts zu Su mit $u > 1$ oder abwärts zu Sd mit $d < 1$.

(A2) Die Wahrscheinlichkeit von u sei p, woraus sich zwangsläufig $1-p$ für die Wahrscheinlichkeit von d ergibt.

(A3) Die erwartete Rendite entspreche dem risikoneutralen Standardzinssatz r, womit die Gleichung

$$\mathsf{E}(S_{i+1}) = S_i \cdot e^{r\Delta t} \tag{1.1.4}$$

gilt.

(A4) Es erfolgt keine Dividendenzahlung.

Die Werte von u, d und p sind zunächst nicht bekannt. Die grundsätzliche Idee der folgenden Überlegungen ist es, die *Erwartungswerte* und *Varianzen* des diskreten und kontinuierlichen Modells gleichzusetzen, womit wir Werte für die drei Unbekannten errechnen können. Aufgrund der ersten beiden Annahmen gilt für den Erwartungswert

$$\mathsf{E}(S_{i+1}) = pS_iu + (1-p)S_id,$$

was gleichgesetzt mit (1.1.4)

$$\mathsf{E}(S_{i+1}) = S_ie^{r\Delta t} = pS_iu + (1-p)S_id$$

oder

$$e^{r\Delta t} = pu + (1-p)d \tag{1.1.5}$$

ergibt, womit wir bereits eine von drei benötigten Gleichungen haben. Als nächstes betrachten wir die Varianzen des kontinuierlichen und diskreten Modells. Für die Varianz des kontinuierlichen Modells gilt

$$\mathsf{Var}(S_{i+1}) = S_i^2 e^{2r\Delta t}(e^{\sigma^2\Delta t} - 1).$$

Auf der anderen Seite haben wir für das diskrete Modell

$$\begin{aligned}\mathsf{Var}(S_{i+1}) &= \mathsf{E}(S_{i+1}^2) - (\mathsf{E}(S_{i+1}))^2 \\ &= p(S_iu)^2 + (1-p)(S_id)^2 - S_i^2(pu + (1-p)d)^2.\end{aligned}$$

Gleichsetzen führt unter Verwendung von (1.1.5) zu

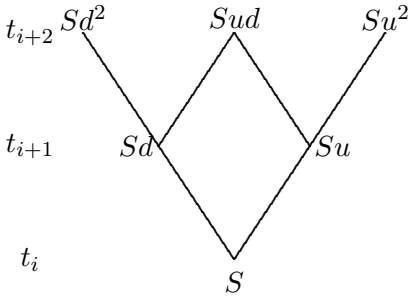
$$e^{2r\Delta t + \sigma^2\Delta t} = pu^2 + (1-p)d^2, \tag{1.1.6}$$

womit wir die zweite Gleichung haben. Als dritte Gleichung setzen wir willkürlich

$$u \cdot d = 1, \tag{1.1.7}$$

was eine gewisse Symmetrie zwischen Kursgewinn und Kursabfall darstellen soll.

Die nebenstehende Skizze zeigt noch einmal unsere Diskretisierung in vereinfachter Form. Die nächste Aufgabe ist nun, das System der drei Gleichungen zu lösen.



Mit $\alpha := e^{r\Delta t}$ folgt aus (1.1.5), (1.1.6) und (1.1.7) durch Elimination

$$0 = u^2 - u(\alpha^{-1} + \alpha e^{\sigma^2 \Delta t}) + 1.$$

Setzen wir noch $2\beta := \alpha^{-1} + \alpha e^{\sigma^2 \Delta t}$, so lautet die Lösung des Gleichungsystems

$$\begin{aligned} \beta &= \frac{1}{2}(e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t}) \\ u &= \beta + \sqrt{\beta^2 - 1} \\ d &= \frac{1}{u} = \beta - \sqrt{\beta^2 - 1} \\ p &= \frac{e^{r\Delta t} - d}{u - d}. \end{aligned}$$

Unser Algorithmus besteht aus zwei Phasen. Zuerst müssen, nachdem die Werte für u und d bekannt sind, die Gitterpunkte berechnet werden. Dies leistet folgender Algorithmus.

$$\begin{aligned} &\text{Berechne für} \quad i = 1, \ldots, M : \\ &\quad \text{Berechne für} \quad j = 1 \ldots, i : \\ &\qquad\qquad S_{ji} := S_0 u^j d^{i-j}. \end{aligned}$$

Hiernach werden in einer Rückwärtsrekursion die konkreten Optionspreise berechnet. Mit $S_{jM} := Su^j d^{M-j}$ gilt also für $j = 0, \ldots, M$ im Fall einer Put–Option für den Startwert $V(S(t_M), t_M) = \max\{E - S(t_M), 0\}$. Analog gilt für die Call–Option $V(S(t_M), t_M) = \max\{S(t_M) - E\}$. Damit berechnen wir für $t_{M-1}, \ldots, 0$ den Optionswert einer europäischen Option $V_{ji} := V(t_i, S_{ji})$ als

$$V_{ji} = e^{-r\Delta t} \cdot (pV_{j+1,i+1} + (1-p)V_{j,i+1}).$$

Bei einer amerikanischen Option muß zusätzlich geprüft werden, ob eine Ausübung nicht sinnvoller ist. Also gilt für eine Call–Option

$$V_{ji} = \max\{(S_{ji} - E)^+, e^{-r\Delta t} \cdot (pV_{j+1,i+1} + (1-p)V_{j,i+1})\},$$

und für eine Put–Option

$$V_{ji} = \max\{(E - S_{ji})^+, e^{-r\Delta t} \cdot (pV_{j+1,i+1} + (1-p)V_{j,i+1})\},$$

Wenn wir dies nun für die Werte $E = 10$, $S = 5$, $r = 0.06$, $\sigma = 0.3$, $T = 1$ für einen europäischen Put implementieren, so erhalten wir die in der nebenstehenden Tabelle eingetragenen Werte. Die Konvergenz der vereinfachten Darstellung gegen den BLACK–SCHOLES–Wert ist deutlich erkennbar. Wer Interesse an weiteren Details hat, sei auf die Bücher [BM] und [L], sowie die Artikel [CR] und [BS] hingewiesen.

M	V(5,0)
4	4.4176454544
8	4.4250702858
16	4.4292454720
32	4.4298558235
64	4.4299254417
128	4.4300422668
256	4.4303789138
1024	4.4304447174
2500	4.4304690361
BLACK–SCHOLES	4.43046477621

■

Diese Beispiele motivierten zwei wesentliche Kriterien für die Güte eines numerischen 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. Seien f und g zwei Funktionen. Dann heißt $f(x) = O(g(x))$, falls

$$\left|\frac{f(x)}{g(x)}\right| \leq c$$

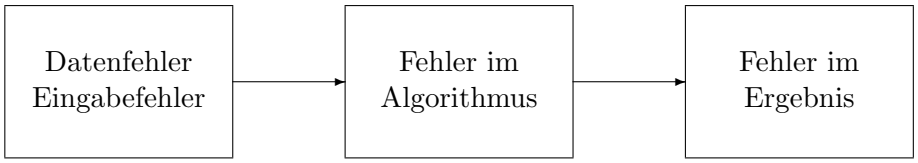
für $x \to x_0$ bzw. $x \to \pm\infty$. Weiter heißt $f(x) = o(g(x))$, falls

$$\frac{f(x)}{g(x)} \to 0$$

für $x \to x_0$ bzw. $x \to \pm\infty$. Ein Beispiel für die Verwendung des „groß O“ ist $\sin x = O(x)$ für $x \to 0$, da $\frac{\sin x}{x} \to 1$ für $x \to 0$. „Klein o“ gilt für $\sin x = o(\sqrt{x})$, $x \to 0$, da $\frac{\sin x}{\sqrt{x}} \to 0$, wenn $x \to 0$. Auch das Restglied der TAYLOR–Entwicklung läßt sich durch „groß O“ ausdrücken. Für die TAYLOR–Entwicklung an der Stelle $x + h$ einer Funktion $f \in \mathcal{C}^k(\mathbb{R})$, $k \geq 2$ mit $0 < h < 1$ gilt $f(x + h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3)$ für $h \to 0$.

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 $f(x_1, x_2) := x_1 x_2$ 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 $a_1^2 > a_2$ und $a_1, a_2 \in \mathbb{R}$. 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

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

Wir nehmen im folgenden an, daß $\|\tilde{x} - x\|$ „klein“ ist, so daß 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

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

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

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

ist. Nun folgt mit der Dreiecksungleichung

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

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

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

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

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

Hier gilt

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

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

Beispiel 2.1.8 Wir betrachten die Addition aus Beispiel 2.1.2. Seien nun

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

Da

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

ist, folgt

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

Das bedeutet

$$\kappa_{\text{rel}} = \left\{ \begin{array}{ll} \leq 1 & \text{falls } x_1, x_2 \text{ gleiches Vorzeichen haben} \\ \gg 1 & \text{falls } x_1 \approx -x_2 \end{array} \right\}.$$

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 die Funktion f nicht explizit bekannt 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 als

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

für ein $\delta > 0$ und sagen

Definition 2.1.9 Das Problem $[f, x, \delta]$ ist gut gestellt (well–posed), falls es eine Konstante $0 \leq L_{\text{rel}} < \infty$ gibt, so daß

$$\frac{\|f(x) - f(\tilde{x})\|}{\|f(x)\|} \leq L_{\text{rel}}(\delta)\frac{\|\tilde{x} - x\|}{\|x\|} \tag{2.1.10}$$

für alle $\tilde{x} \in E_\delta(x)$ gilt, wobei x und $f(x)$ von Null verschieden sein müssen. Ist $[f, x, \delta]$ gut gestellt, so bezeichne $L_{\text{rel}}(\delta)$ die *minimale* Konstante, so daß (2.1.10) 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

$$\kappa_{\text{rel}} := \lim_{\delta \to 0} L_{\text{rel}}(\delta) \tag{2.1.11}$$

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

Man beachte, daß sich (2.1.10) als eine lokale LIPSCHITZ–Bedingung interpretieren läßt.

Bemerkung 2.1.12 Für den Fall, daß f differenzierbar ist, gilt aufgrund des Mittelwertsatzes für die relative Kondition

$$\kappa_{\text{rel}} = \frac{\|x\|}{\|f(x)\|} \cdot \|Df(x)\| \tag{2.1.13}$$

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

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

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

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

Beispiel 2.1.14 Seien $A \in \mathbb{R}^{n \times n}$ nichtsingulär und $b \in \mathbb{R}^n$ gegeben. Gesucht ist die Lösung $x \in \mathbb{R}^n$ von $Ax = b$. Zur Berechnung der relativen Kondition von x betrachten wir das Problem $f(b) = A^{-1}b$. Diese Gleichung ist offenbar linear bezüglich b, und daher gilt für die Ableitung $Df(x) = A^{-1}$. Setzen wir dies in die obige Formel (2.1.13) zur Berechnung der relativen Kondition ein, erhalten wir

$$\kappa_{\text{rel}} = \frac{\|b\|}{\|A^{-1}b\|} \cdot \|A^{-1}\| = \frac{\|Ax\|}{\|x\|} \cdot \|A^{-1}\| \leq \|A\| \cdot \|A^{-1}\|.$$

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

2.2 Maschinenzahlen und Rundungsfehler

Nachdem unsere bisherigen Betrachtungen den einem Problem zugrunde liegenden Daten zugewandt waren, wollen wir nun einen weiteren Problemkreis aufgreifen, der häufig den Ursprung von Fehlern in Rechenresultaten in sich birgt. Wir betrachten die Durchführung von Rechenoperationen auf digitalen Rechnern, in denen es in aller Regel zu *Rundungsfehlern* kommt. Hierzu müssen wir uns zunächst mit der Darstellung von Zahlen in Rechenmaschinen beschäftigen. Auf Rechnern lassen sich nur *endlich viele* Zahlen *exakt* darstellen. Diese heißen *Maschinenzahlen*. Alle anderen für eine Rechnung benötigten Zahlen müssen geeignet approximiert werden. Wir beginnen mit der Darstellung der *natürlichen Zahlen* (unsigned integers).

Sei $b \in \mathbb{N}$ (b für Basis) und $b \geq 2$. Dann hat jede Zahl $x \in \mathbb{N}$ eine Darstellung der Form

$$x = \sum_{i=0}^{m-1} d_i b^i, \tag{2.2.1}$$

wobei $d_i \in \mathbb{N}$ mit $d_i < b$ und $m \in \mathbb{N}$ sind. Für spezielle *Basen* b gibt es eigene Bezeichnungen.

b	Bezeichnung
10	Dezimaldarstellung
2	Binär– oder Dualdarstellung
8	Oktaldarstellung
16	Hexadezimaldarstellung

Beispielsweise ist $42_{\text{dez}} = 101010_{\text{bin}} = 52_{\text{okt}} = 2A_{\text{hex}}$. Rechner arbeiten intern mit $b = 2$. Eine Informationseinheit $d_i \in \{0, 1\}$ heißt *bit*. 8 bit heißen *byte*.

Zur Darstellung von Zahlen fixiert man ein $m \in \mathbb{N}$ (vgl. (2.2.1)) und kann damit die Zahlen $0, 1, 2, \ldots, b^m - 1$ darstellen. Typische Werte für m mit $b = 2$ sind:

m	benötigter Speicherplatz	Zahlbereich	Bitdarstellung von 42_{dez}
8	1 byte	$0, \ldots, 2^8 - 1 (= 255)$	0010 1010
16	2 bytes	$0, \ldots, 2^{16} - 1 (= 65535)$	0000 0000 0010 1010
32	4 bytes	$0, \ldots, 2^{32} - 1 (= 4294967295)$	...
64	8 bytes	$0, \ldots, 2^{64} - 1 (\approx 1.84 \cdot 10^{19})$	...

Nun zur Darstellung der *ganzen Zahlen* (signed integers). Diese erfolgt meist nach der Methode „b–Komplement“: Speichere $x \in \{-2^{m-1}, \ldots, 2^{m-1} - 1\}$ als

$$\left\{ \begin{array}{cc} x & \text{falls } x \geq 0 \\ x + 2^m & \text{falls } x < 0 \end{array} \right\}$$

wie oben ab. Für $m = 8$ gilt dann:

Bitdarstellung	unsigned	signed
0000 0000	0	0
0000 0001	1	1
⋮	⋮	⋮
0111 1111	127	127
1000 0000	128	−128
⋮	⋮	⋮
1111 1111	255	−1

Hierbei werden Rundungen natürlich immer modulo 2^m ausgeführt, was man beim Rechnen berücksichtigen sollte.

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

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

wobei $s \in \{0, 1\}$, $e \in \mathbb{Z}$, $d_i < b$ sind. Diese Darstellung ist keineswegs eindeutig, was das Beispiel $1.\overline{9} = 2$ zeigt. 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 ergo $x = (-1)^s 1.d_1 d_2 \dots d_m \cdot 2^e$ 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, 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)

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

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

Die technischen Daten der standardisierten Datentypen sind in folgender Tabelle zusammengefaßt.

	single precision/float	double precision/double
Größe	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	$-126, \dots, 127$	$-1022, \dots, 1023$
Zahlbereich normalisiert	$1.2 \cdot 10^{-38}, \dots, 3.4 \cdot 10^{38}$	$2.2 \cdot 10^{-308}, \dots, 1.8 \cdot 10^{308}$
kleinste positive denorm. Zahl	$1.4 \cdot 10^{-45}$	$4.9 \cdot 10^{-324}$
Genauigkeit für Dezimalzahlen	$7 - 8$ Stellen	$15 - 16$ Stellen

Rundung und Maschinengenauigkeit

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

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

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*

$$\mathrm{fl}(x) := \left\{ \begin{array}{ll} \pm(\sum_{i=0}^{m} d_i b^{-i})b^e & \text{falls } d_{m+1} < \frac{b}{2} \\ \pm(\sum_{i=0}^{m} d_i b^{-i} + b^{-m})b^e & \text{falls } d_{m+1} \geq \frac{b}{2} \end{array} \right\}, \tag{2.2.3}$$

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*

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

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

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

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: $(x \oplus y) \odot (x \oplus y) \neq (x \odot x) \oplus (x \odot y) \oplus (y \odot x) \oplus (y \odot y)$. Insbesondere spielt also die Reihenfolge der Operationen eine wichtige 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 2.1.8 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 hier verstärkt $a_1^2 \gg a_2$ angenommen sei, so daß 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:

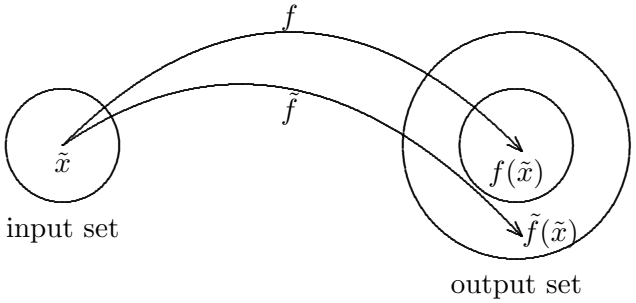
$$\left|\frac{\partial u^*}{\partial y_3}\cdot\frac{y_3}{u^*}\right| = \left|\frac{-a_2}{(a_1+\sqrt{a_1^2-a_2})^2}\cdot\frac{\sqrt{a_1^2-a_2}(a_1+\sqrt{a_1^2-a_2})}{a_2}\right| = \left|\frac{\sqrt{a_1^2-a_2}}{a_1+\sqrt{a_1^2-a_2}}\right| \leq 1.$$

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 vier 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 zweiten Schritt Auslöschung auf. ∎

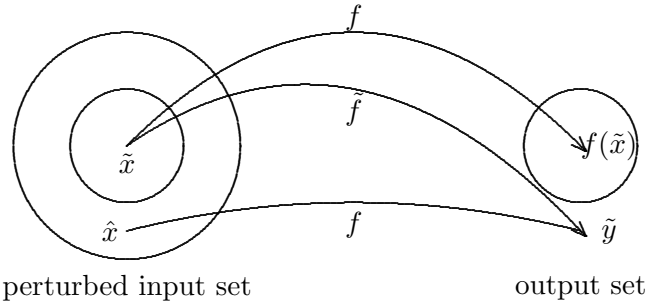
Zwei wesentliche Möglichkeiten, die Akkumulation von Fehlern im Laufe eines Algorithmus einzuschätzen, bilden die *Vorwärts–* und *Rückwärtsfehleranalyse*. Bei der Vorwärtsanalyse (forward analysis) wird die Menge $\tilde{f}(\tilde{x})$ aller durch Eingabefehler und Fehler im Algorithmus gestörten Resultate 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 auftretende 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

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

wobei $|\varepsilon_i| \leq \text{eps}$ 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 $|\delta_i| \leq 2\text{eps}$ für $i = 1, 2$ und $|\delta_i| \leq \text{eps}$ 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 $F_{\text{Rechnung}} \gg \text{eps}$ 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 direkte Lösungsmethoden für lineare Gleichungssysteme. Eine Vielzahl von Problemen in den Naturwissenschaften wird auf das Lösen von linearen Gleichungssystemen zurückgeführt, so z.B. die Diskretisierung von Differentialgleichungen. Zunächst werden grundlegende Begriffe und Bezeichnungen eingeführt. 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

$$Ax = b \quad \text{äquivalent zu} \quad \sum_{j=1}^{n} a_{ij}x_j = b_i, \ \text{für } i = 1, \ldots, n \tag{3.1.1}$$

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

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

(iii) A ist regulär;

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

(v) die Gleichung $Ax = 0$ 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 durch

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

eindeutig bestimmt ist, 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 jeweils $n!$ 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 100 mflop–Rechner ($\hat{=}10^8$ flops/sec) für die Berechnung von x mittels CRAMER'scher Regel für $n = 10, 12, 14, 16, 18, 20$ bräuchte.

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

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, also

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

Analog gilt für eine untere Dreiecksmatrix

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

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 Diagonalmatrix ist, ist der Fall trivial. Hat A obere Dreiecksgestalt, so ist (3.1.1) von der Form

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

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 $i = 1, \ldots, n$ ist. 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 $b \in \mathbb{R}^n$. Berechne für $j = n, \ldots, 1$

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

■

Der *Rechenaufwand* für diesen Algorithmus beträgt für jedes $j = n, n-1, \ldots, 1$ 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

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

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, so daß die resultierenden Einträge der ersten Spalte der Matrix A verschwinden, d.h.

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

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

  `IF` $a_{jj} \neq 0$

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

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

    `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 dieses 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

$$(A, b) = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 4 & 0 & -3 & 1 & -8 \\ 6 & 1 & -1 & 6 & -16 \\ -2 & -5 & 4 & 1 & -12 \end{array}\right).$$

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

$$\begin{array}{c|c} i & l_{i1} \\ \hline 2 & 2 \\ 3 & 3 \\ 4 & -1 \end{array} \quad \rightarrow \quad \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 4 & 8 & -3 & -19 \\ 0 & -6 & 1 & 4 & -11 \end{array}\right).$$

Nun folgt $j = 2$:

$$\begin{array}{c|c} i & l_{i2} \\ \hline 3 & 2 \\ 4 & -3 \end{array} \quad \rightarrow \quad \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 10 & -11 & -41 \end{array}\right).$$

Und schließlich noch $j = 3$:

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

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

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

```
            FOR k = j + 1, ..., n
                                    a_ik ← a_ik − a_ij a_jk
            END
        END
    END
END
```

■

Der Rechenaufwand für diesen Algorithmus ist

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

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

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

und

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

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 $O\left(\frac{1}{2}n^2\right)$ und für die LR–Zerlegung $O\left(\frac{1}{3}n^3\right)$, insgesamt also $O\left(\frac{1}{3}n^3\right)$.

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 $j \leq p \leq n$, so daß

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

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

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

∎

Algebraisch wird die Vertauschung mittels elementarer Permutationsmatrizen P_{ik} dargestellt. Sie entstehen aus der Einheitsmatrix durch Vertauschen der Zeile i mit der Zeile k. Es ist etwa

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

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

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

Beachte

$$\det P_{ik} = \left\{ \begin{array}{ll} 1 & \text{für } i = k \\ -1 & \text{für } i \neq k \end{array} \right\}. \tag{3.3.14}$$

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 $O\left(\frac{1}{3}n^3\right)$. 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 $O\left(\frac{1}{3}n^3\right)$ Operationen im Vergleich zu $O(n!)$ Operationen bei der Anwendung der Leibniz–Regel. (Für $n = 20$ bedeutet dies auf einem 100 mflop–Rechner

0.1 ms gegenüber 16000 Jahren.) 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 $O = \left(\frac{1}{3}n^3\right)$. Der Rest erfolgt über Vorwärts– und Rückwärtseinsetzen mit einem Aufwand von $O = \left(\frac{1}{2}n^2\right)$. Auch die Berechnung der Inversen ist durch das Lösen von Gleichungssystemen möglich. Es gilt

$$PA = LR \text{ genau dann, wenn } A^{-1} = R^{-1}L^{-1}P$$

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 $O\left(\frac{n^3}{3}\right)$ für die LR–Zerlegung, sowie n–mal Vor– und Rückwärtseinsetzen mit je $O\left(\frac{n^2}{2}\right)$, also insgesamt $O = \left(\frac{4n^3}{3}\right)$. In der Numerik wird in der Regel die Inverse einer Matrix *nie explizit* benötigt. Der Ausdruck $x = A^{-1}b$ ist daher 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 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

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

für $1 \le i_1 \le i_l \le n$ 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 Differentialgleichung, siehe Anhang A, 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

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

Für das untere Dreieck $i \geq k$ gilt

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

Dies ist äquivalent zu

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

Für $k = 1$ gilt

$$l_{i1} d_{11} = a_{i1},$$

womit für $i = 1$ $d_{11} = a_{11}$ folgt, da $l_{11} = 1$, und für $i > 1$ $l_{i1} = \frac{a_{i1}}{d_{11}}$. 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

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

und im Fall $i > k$ als

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

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` $k = 1, \dots, n$

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

`END`

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

`ELSE`

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

`FOR` $i = k+1, \dots, n$

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

`END`

`END` ∎

Bemerkung 3.4.10

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

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

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

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 $O(n)$ im Unterschied zu $O(n^2)$ 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 $O(pqn)$ und das Vorwärts– und Rückwärtseinsetzen $O((p+q)n)$. Insbesondere gilt, falls p und q klein sind relativ zu n, daß der Aufwand zu $O(n)$ 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 den 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

$$\begin{aligned} & \left(1 - \kappa_{\|\cdot\|}(A)\frac{\|\Delta A\|}{\|A\|}\right)\|\Delta x\| \\ \leq\ & \kappa_{\|\cdot\|}(A)\left(\frac{\|\Delta A\|}{\|A\|})\|x\| + \frac{\|\Delta b\|}{\|b\|}\right) \\ \leq\ & \kappa_{\|\cdot\|}(A)\left[\frac{\|\Delta A\|}{\|A\|}\|x\| + \frac{\|\Delta b\|}{\|b\|}\|x\|\right]. \end{aligned}$$

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

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

Satz 3.7.4 Für orthogonale Matrizen gilt

i) Ist $Q \in \mathbb{O}_n$, so gilt $|\det Q| = 1$.

ii) Für Q_1 und $Q_2 \in \mathbb{O}_n(\mathbb{R})$ gilt $Q_1Q_2 \in \mathbb{O}_n$.

iii) Es gilt $\|Qx\|_2 = \|x\|_2$ für alle $x \in \mathbb{R}^n$. Geometrisch bedeutet dies, daß orthogonale Matrizen Drehungen oder Spiegelungen beschreiben.

iv) Für jedes $A \in \mathbb{R}^{n\times n}$ und $Q \in \mathbb{O}_n$ gilt $\|A\|_2 = \|QA\|_2 = \|AQ\|_2$.

v) Ebenso gilt $\kappa_2(QA) = \kappa_2(AQ) = \kappa_2(A)$.

vi) Für alle $Q \in \mathbb{O}_n$ gilt $\kappa_2(Q) = 1$. ∎

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) $Q_v^2 = I$, denn es ist

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

iii) Mit i) und ii) gilt: $Q_v \in \mathbb{O}_n(\mathbb{R})$, da $Q_v^TQ_v = Q_v^2 = I$.

iv) $Q_{\alpha v} = Q_v$ für alle reellen $\alpha \neq 0$.

v) $Q_vy = y$ ist äquivalent zu $y^Tv = 0$.

vi) $Q_vv = -v$. ∎

Im folgenden ist es unsere

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

$$Q_vy = \pm\|y\|_2e^1 \tag{3.7.8}$$

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

$$Q_v y = (I - \frac{2}{v^T v} v v^T) y = y - \frac{2}{v^T v}(v^T y) v = c e^1$$

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:

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

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

$$\begin{aligned} & y^T y + 2\alpha y_1 + \alpha^2 - 2(y^T y + \alpha y_2) = 0 \\ \iff \quad & -y^T y + \alpha^2 = 0 \\ \iff \quad & \alpha = \pm \|y\|_2. \end{aligned}$$

Eine sinnvolle Wahl für α ist daher

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

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

$$\alpha \leftarrow \operatorname{sign}(y_1)\|y\|_2$$

und

$$v \leftarrow y + \alpha e^1.$$

■

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 (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

$$Q_2 Q_1 A = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & 0 & & \\ \vdots & \vdots & \tilde{A}^{(3)} & \\ 0 & 0 & & \end{pmatrix}$$

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

$$Q_1 A = \begin{pmatrix} -3 & -\frac{1}{3} \\ 0 & -\frac{2}{3} \\ 0 & -\frac{2}{3} \end{pmatrix}$$

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

$$Q_2 Q_1 A = \begin{pmatrix} -3 & -\frac{1}{3} \\ 0 & \frac{2\sqrt{2}}{3} \\ 0 & 0 \end{pmatrix}.$$

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 $O\left(\frac{2}{3}n^3\right)$ und für $m \gg n$ bei $O\left(mn^2\right)$.

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

Falls $b = a$,

setze $c = 1$ und $s = 0$.

Falls $|b| \geq |a|$,

setze $\tau := \frac{a}{b}$ und $s := \frac{1}{\sqrt{1+\tau^2}}$ sowie $c := s\tau$.

sonst

setze $\tau := \frac{b}{a}$ und $c := \frac{1}{\sqrt{1+\tau^2}}$ sowie $s := c\tau$. ∎

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

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

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

$$\varphi := \varphi_{ik} := \left\{ \begin{array}{cc} 1 & \text{falls } c = 0 \\ \frac{1}{2}\,\text{sign}\,(c)s & \text{falls } |s| < |c| \\ 2\frac{\text{sign}\,(s)}{c} & \text{falls } |c| < |s| \end{array} \right\} \tag{3.7.32}$$

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

Algorithmus 3.7.33 Falls $\varphi = 1$, setze $c := 0$ und $s := 1$. Falls $|\varphi| < 1$ setze $s := 2\varphi$ und $c := \sqrt{1 - s^2}$. Sonst setze $c := \frac{2}{\varphi}$ und $s := \sqrt{1 - c^2}$. ∎

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 $O\left(\frac{4}{3}n^3\right)$, und falls $m \gg n$ von der Ordnung $O\left(2mn^2\right)$. Der Aufwand 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 $O\left(\frac{2}{3}n^3\right)$ und falls $m \gg n$ von der Ordnung $O(m \cdot n^2)$.

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 die 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 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 können bei der GIVENS–Transformation gezielt vorhandene Nullenmuster ausgenutzt werden.

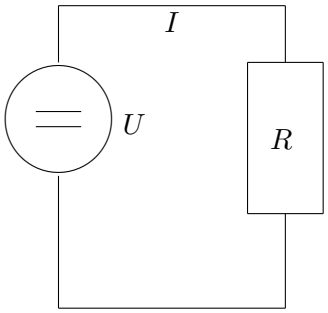
4 Lineare Ausgleichsprobleme

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 OHM'schen Gesetz gilt (zumindest für gewisse Temperaturbereiche)

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



Es sei 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

$$\|g_N - f\|_{L_2}^2 := \int_0^T \left(g_N(t) - f(t)\right)^2 dt = \min \tag{4.1.8}$$

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

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

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

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

zur Bestimmung von g_N, also zur Ermittelung von a_k, b_k. ∎

4.2 Lineare Ausgleichsprobleme — GAUSS'sche 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.

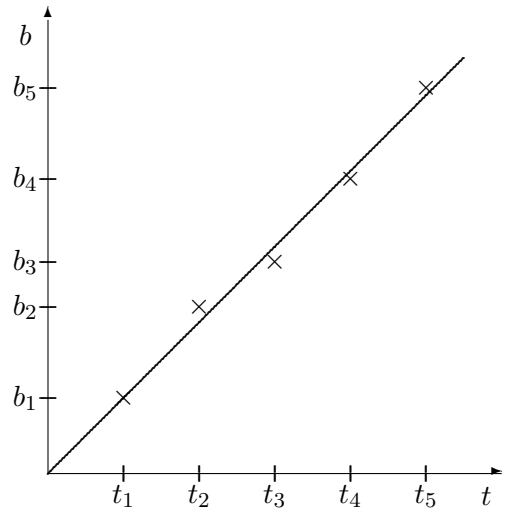
Also gelte

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

an allen betrachteten m Stellen. Wir nehmen weiter an, daß diesem Prozeß eine gewisse Gesetzmäßigkeit zugrunde liege, so daß 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, so daß die x_i so zu bestimmen sind, daß

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

mit *Gewichten* $\omega_i > 0$ wird. Diese Lösungsmethode nennt man GAUSS'sche 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)

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

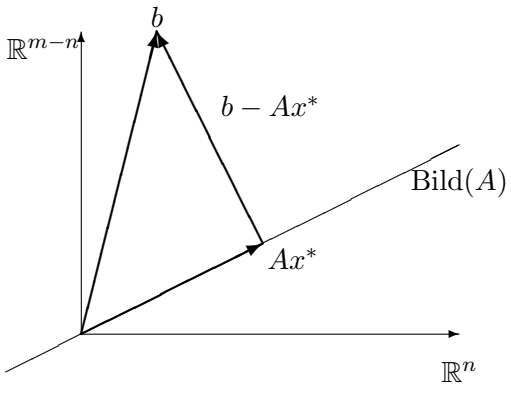
mit $A := (a_{ij})_{\substack{i=1,\ldots,m \\ j=1,\ldots,m}} \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$, was für unsere Aufgabe bedeutet: finde das $x^* \in \mathbb{R}^n$, für das

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

ist. 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 $(\cdot,\cdot)$ ist. 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 $\text{Bild}(A) = \{Ax \in \mathbb{R}^m : x \in \mathbb{R}^n\}$ 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 $Ax - b \perp \text{Bild}(A)$. Unter Ausnutzung der Eigenschaft $\sqrt{(\cdot,\cdot)} = \|\cdot\|_2$ übersetzt sich dies zu

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

für alle $w \in \text{Bild}(A)$.

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

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

was schließlich

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

als gesuchte *Normalengleichungen* ergibt. 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

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 hergeleitete 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 das Skalarprodukt in der rechten Gleichung aufgrund von (4.3.5) verschwindet, gilt

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

und da der erste Summand nichtnegativ ist, folgt

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

und damit die Behauptung. 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 $\hat{u}$ 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

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

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

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

für jedes $i = 1,\ldots,n$. Also $v_n \perp U_n$. Da aber $v_n \in U_n$, muß $v_n = 0$ sein. ∎

Wir fassen die Ergebnisse dieses Abschnitts zusammen als

Satz 4.3.9 Sei $\{\varphi_1,\ldots,\varphi_n\}$ eine Orthonormalbasis von $U_n \in V$. 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),

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

mit $A \in \mathbb{R}^{m\times n}$, $m \geq n$ löst. Insbesondere zieht der Fall $\operatorname{rang} A = n$ 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 $A^+ A = I$ ist, 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 $\operatorname{rang} A = p \leq \min(m, n)$. 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 $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p, 0, \ldots, 0) \in \mathbb{R}^{m\times n}$. Dabei heißen $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$ die *Singulärwerte* von A.

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)

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

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 $\|Ax^* - b\|_2$ minimal ist, und nicht $\|x^*\|_2$. ∎

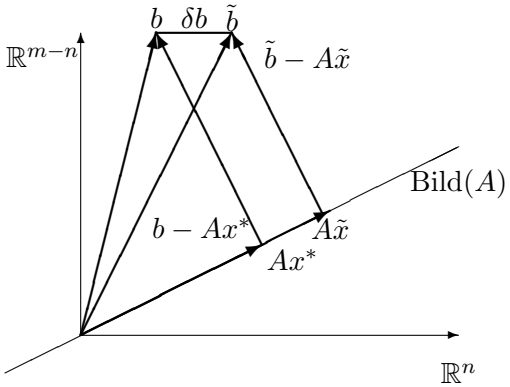
Einen Algorithmus zur praktischen Berechnung werden wir in Abschnitt 4.6 konstruieren. Wir wollen uns zunächst noch ein paar Gedanken zur Kondition dieses Problems machen.

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 Bild (A). Wegen $Ax^* - b \perp \text{Bild}\,(A)$ 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 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) $A^TAx = A^Tb$ 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^TA 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^TA und A^Tb.

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

3.) löse $Ly = A^Tb$ und $L^Tx = D^{-1}y$ durch Vorwärts- und Rückwärtseinsetzen.

∎

Dieser Algorithmus hat aber Nachteile. Für große $m \gg n$ ist die explizite Berechnung von A^TA aufwendig. Dabei besteht zusätzlich die Gefahr von Auslöschungseffekten. Nach Satz 4.5.4 erfolgt die Fehlerverstärkung bei einer Störung in A mit dem Fehler $\kappa_2(A^TA) = \kappa_2(A)^2$, 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 $Q \in \mathbb{R}^{m\times m}$

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \|Ax - b\|_2 &= \min_{x\in\mathbb{R}^n} \|Q(Ax - b)\|_2 \\ &= \min_{x\in\mathbb{R}^n} \|QAx - Qb\|_2 \end{aligned}$$

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) $\min\limits_{x\in\mathbb{R}^n} \|Ax - b\|_2$, und $\|Ax^* - b\|_2 = \|b_2\|_2$. ∎

Jetzt betrachten wir den schwierigeren Fall $p = \operatorname{rang} A < n$. 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 $p = \operatorname{rang} A$, $Q \in \mathbb{R}^{m\times m}$ 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 $LL^T x_2 = W^T v$

7.) setze noch $x_1 := v - W^T x_2$

■

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 $\operatorname{rang} A = n$ 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 wird ein wichtiges Beispiel aus den Anwendungen behandelt, 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 wird die Lösung von u an den *Gitterpunkten* x_j folgendermaßen berechnet: 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 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 $r(x_i) > 0$ sind. 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 die 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 zu einem großen Teil bereits aus der Linearen Algebra bekannt sind.

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 ein Polynom vom Grad n und seine Nullstellen 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, im folgenden Beispiel doch einmal die Eigenwerte einer Matrix aus den Nullstellen des charakteristischen Polynoms berechnet.

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

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

folgt. 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. ∎

Einige weitere Tatsachen aus der Linearen Algebra sind:

- 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 genau dann *ähnlich*, wenn es eine invertierbare Matrix $T \in \mathbb{R}^{n\times n}$ mit der Eigenschaft
$$B = T^{-1}AT$$
gibt.
- 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:
$$\begin{aligned}\det(T^{-1}AT - \lambda I) &= \det(T^{-1}(A - \lambda I)T)\\ &= \det(T^{-1})\det(A - \lambda I)\det(T)\\ &= \det(A - \lambda I)\end{aligned}$$
∎
- 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. ∎

In Zukunft werden zwei Sätze über die Schur'sche Normalform oder Schur–Faktorisierung benötigt. In der linearen Algebra zeigt man, daß nicht jede Matrix diagonalisierbar ist. Aber eine Transformation auf obere Dreiecksgestalt ist immer möglich.

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. $Q^{-1} = \bar{Q}^T =: Q^*$), so daß
$$Q^*AQ = \begin{pmatrix} \lambda_1 & & & * \\ & \ddots & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{pmatrix} =: R. \tag{5.2.6}$$
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 $Q \in \mathbb{R}^{n\times n}$, so daß

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

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

In diesem Unterkapitel wird die Frage behandelt, 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 wieder 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 $\delta \to 0$. ∎

Das Eigenwertproblem kann 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 die Behauptung direkt aus Gleichung (5.3.3) für $p = 2$. ∎

5.4 Eigenwertabschätzungen

Wir erinnern zunächst an ein paar Eigenschaften von Eigenwerten.

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 für das gesamte Spektrum

Korollar 5.4.3 Für das Spektrum einer Matrix A gilt $\sigma(A) \leq \|A\|$. ■

Hieraus sehen wir, daß umgekehrt zu jedem $\varepsilon > 0$ eine Matrixnorm $\|\cdot\|_p$ existiert, so daß die Ungleichung $\|A\|_p \leq \rho(A) + \varepsilon$ gilt, in der $\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}$ bezeichnet. 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: Sei $\lambda \in \sigma(A)$ und v zugehöriger Eigenvektor. Sei i so gewählt, daß $|v_i| = \max_{1 \leq j \leq n} |v_j|$. Aus $Av = \lambda v$ folgt dann

$$\sum_{j=1}^{n} a_{ij} v_j = \lambda v_i \Leftrightarrow \sum_{j \neq i} a_{ij} v_j = (\lambda - a_{ii}) v_i.$$

Also gilt

$$|v_i||\lambda - a_{ii}| = \left| \sum_{j \neq i} a_{ij} v_j \right| \leq \sum_{j \neq i} |a_{ij}||v_j|,$$

und damit

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \frac{|v_j|}{|v_i|} \leq \sum_{j \neq i} |a_{ij}|.$$

■

Die Menge der Eigenwerte kann aber noch 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

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

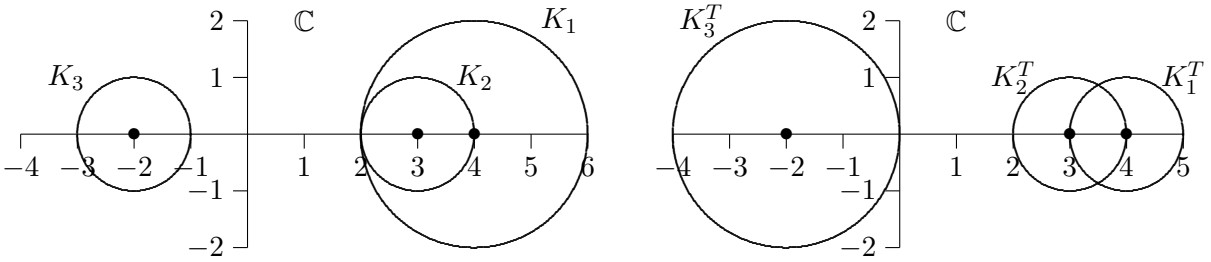
Beweis: Die Aussagen folgen aus 5.4.1(v) zusammen mit (5.4.6). ∎

Beispiel 5.4.10 Wir betrachten die GERSCHGORIN–Kreise der Matrizen

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

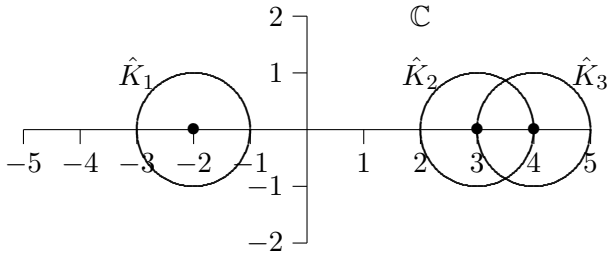
A hat das Spektrum $\sigma(A) = \{3.43 \pm 0.14i, -1.86\}$. Die GERSCHGORIN–Kreise von A und A^T sind:

$$\begin{aligned} K_1 &= \{z : |z-4| \le 2\} & K_1^T &= \{z : |z-4| \le 1\} \\ K_2 &= \{z : |z-3| \le 1\} & K_2^T &= \{z : |z-3| \le 1\} \\ K_3 &= \{z : |z+2| \le 1\} & K_3^T &= \{z : |z+2| \le 2\} \end{aligned}$$



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

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



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

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

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 $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. 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

$$d(S^k, T) \leq \|\alpha_k x^k - v^1\|_2 = |c_1^{-1}| \|r^{(k)}\|_2 = O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \tag{5.5.4}$$

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

$$\begin{aligned}
\lambda^{(k)} &= \lambda_1 \frac{(\alpha_k x^{(k)})^T (\alpha_{k+1} x^{(k+1)})}{\|\alpha_k x^{(k)}\|_2^2} \\
&= \lambda_1 \frac{(v^1 + O(|\frac{\lambda_2}{\lambda_1}|)^k)^T (v^1 + O(|\frac{\lambda_2}{\lambda_1}|)^{k+1})}{\|v^1 + O(|\frac{\lambda_2}{\lambda_1}|)^k\|_2^2} \\
&= \lambda_1 \frac{1 + O(|\frac{\lambda_2}{\lambda_1}|^k)}{1 + O(|\frac{\lambda_2}{\lambda_1}|^k)} \\
&= \lambda_1 \left(1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)\right),
\end{aligned}$$

womit wir

$$\left|\lambda^{(k)} - \lambda_1\right| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right) \tag{5.5.6}$$

erreicht haben.

Bemerkung 5.5.7 Falls A symmetrisch ist, gilt für obige Konvergenz sogar

$$\left|\lambda^{(k)} - \lambda_1\right| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right).$$

■

Bevor nun einen Algorithmus zur Vektoriteration angegeben wird, ein paar Bemerkungen über die *Skalierung der Iterierten*. 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

$$\begin{aligned}
\tilde{y}^{(k+1)} &= A y^{(k)} \\
\lambda^{(k)} &= (\tilde{y}^{(k+1)})^T y^{(k)} \\
y^{(k+1)} &= \frac{\tilde{y}^{(k+1)}}{\|\tilde{y}^{(k+1)}\|_2}
\end{aligned}$$

■

Hierzu sei noch bemerkt, daß unter der Annahme $y^{(0)} = x^{(0)}$ mit vollständiger Induktion nach k

$$y^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|_2} = \frac{A^k x^{(0)}}{\|A^k x^{(0)}\|_2}$$

folgt. 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 wurde 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

$$\lambda_{n-j} = \frac{4}{h^2}\sin^2\left(\frac{1}{2}j\pi h\right)$$

mit $j = 1, \ldots, n-1$ und $h = \frac{1}{n}$ explizit bekannt sind. O.B.d.A. sei die Numerierung hier so gewählt, daß $\lambda_1 > \lambda_2 > \ldots > \lambda_{n-1}$ gilt. Betrachte nun

$$\begin{aligned}\left|\frac{\lambda_2}{\lambda_1}\right| &= \frac{\lambda_2}{\lambda_1} = \frac{\sin^2\left(\frac{1}{2}(n-2)\pi h\right)}{\sin^2\left(\frac{1}{2}(n-1)\pi h\right)} = \frac{\sin^2\left(\frac{1}{2}\pi - \pi h\right)}{\sin^2\left(\frac{1}{2}\pi - \frac{1}{2}\pi h\right)} \\ &= \frac{\cos^2(\pi h)}{\cos^2\left(\frac{1}{2}\pi h\right)} \doteq \frac{(1-\frac{1}{2}(\pi h)^2)^2}{(1-\frac{1}{2}(\frac{1}{2}\pi h)^2)^2} \\ &\doteq \frac{1-\pi^2h^2+\frac{1}{4}\pi^4h^4}{1-\frac{1}{4}\pi^2h^2+\frac{1}{64}\pi^2h^2} \approx 1-\pi^2h^2\end{aligned}$$

Hieran ist ersichtlich, daß man für $h \ll 1$ mit einer sehr langsamen Konvergenz $\lambda^{(k)} \to \lambda_1$ mit einem Faktor $\approx \left|\frac{\lambda_2}{\lambda_1}\right|^2 \doteq 1-\pi^2h^2$ 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 läßt sich 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 in mehreren Fällen gezeigt wurde. 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, so daß

$$|\mu - \lambda_i| < |\mu - \lambda_j| \tag{5.6.1}$$

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. Im nächsten Unterkapitel wird der wichtigste — auf der QR–Zerlegung basierende — Algorithmus zur Bestimmung von Eigenwerten von Eigenvektoren behandelt.

5.7 QR–Verfahren zur Berechnung von Eigenwerten und Eigenvektoren

In Satz 5.3.5 wurde gezeigt, 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:

$$Q^T AQ = \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \tag{5.7.1}$$

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 für $n = 5$ 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, so daß 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 $S_k = \mathrm{diag}(\sigma_1^{(k)}, \ldots, \sigma_n^{(k)})$ mit $|\sigma_i^{(k)}| = 1$, so daß

$$\lim_{k\to\infty} S_{k-1} Q_k S_k = I \tag{5.7.7}$$

und

$$\lim_{k\to\infty} S_k R_k S_{k-1} = \lim_{k\to\infty} S_{k-1} A_k S_k = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) =: D. \tag{5.7.8}$$

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 $A = Y^{-1}DY$ mit $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ 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

$$\lim_{k\to\infty} S_k R_k S_{k-1} = \lim_{k\to\infty} S_{k-1} A_k S_k = \begin{pmatrix} \lambda_1 & & & \\ & \ddots & * & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \tag{5.7.8'}$$

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

$$\lim_{k\to\infty} S_{k-1}^* Q_k S_k = I \tag{5.7.7'}$$

und

$$\lim_{k\to\infty} S_k^* R_k S_{k-1} = \lim_{k\to\infty} S_{k-1}^* A_k S_k = \begin{pmatrix} \lambda_1 & & & \\ & \ddots & * & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \tag{5.7.8"}$$

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 $Q^{-1}AQ = Q^TAQ = D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. Hiermit folgt

$$A^k = Q^T D^k Q \tag{5.7.10}$$

wobei $D^k = \mathrm{diag}(\lambda_1^k, \ldots, \lambda_n^k)$ 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 $\lim\limits_{k\to\infty} E_k = 0$. 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

$$(I + E_k)^T (I + E_k) =: I + F_k \tag{5.7.13}$$

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)

$$A^k = Q^T (\tilde{Q}_k \tilde{R}_k) D^k R = (Q^T \tilde{Q}_k)(\tilde{R}_k D^k R), \tag{5.7.16}$$

[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.

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}

$$\begin{aligned} A^{k+1} &= A A^k = (AP_k)U_k = P_k A_{k+1} U_k \\ &= P_k (Q_{k+1} R_{k+1}) U_k = P_{k+1} U_{k+1} \end{aligned}$$

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 $B = Q^T A Q$ 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 $O(\frac{4}{3}n^3)$ für Teil I und $O(n^2)$ für Teil II.

Bemerkung 5.7.21

- Falls A nichtsymmetrisch ist, transformiere A auf obere HESSENBERG–Gestalt

$$B = Q^T A Q = \begin{pmatrix} * & & & & \\ * & \ddots & & * & \\ & \ddots & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & * & * \end{pmatrix}.$$

  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, so daß 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, so daß $B^T B$ 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 $Q \in \mathbb{O}_n$, so daß

$$PAQ = \begin{pmatrix} B \\ O \end{pmatrix}$$

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 $b_2 \neq 0$, so daß 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 $O(\frac{4}{3}n^3)$ bei (I) und $O(n^2)$ 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 Iterative Verfahren zur Lösung nichtlinearer Gleichungen

6.1 Einführung

Bisher haben wir lediglich Verfahren zur Lösung *linearer* Gleichungssysteme behandelt. Nun gehen wir den Bereich der *nichtlinearen* Gleichungen und deren numerische Behandlung an. Ein einfaches Beispiel für einen solchen Fall ist die Berechnung der Nullstellen der Gleichung

$$f(x) = ax^2 + bx + c,$$

wobei a, b, c fest vorgegeben seien. Die Nichtlinearität dieser Gleichung ist durch das quadratische Auftreten der Variablen x gegeben. Im allgemeinen hat man ein nichtlineares System in n Unbekannten $x_1, \ldots, x_n$ mit r Gleichungen gegeben, also eine Funktion $f : \mathbb{R}^n \to \mathbb{R}^r$, und man sucht die Lösungen der Gleichung

$$f(x) = 0, \tag{6.1.1}$$

wobei

$$f(x) := \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_r(x_1, \ldots, x_n) \end{pmatrix}$$

ist. Hierfür schreibt man auch $f = (f_1, \ldots, f_r)^T$. In dem speziellen Fall $r = n = 1$ hat man es mit einem skalaren Problem zu tun. Ein weiterer Spezialfall ist der Fall, in dem mehr Gleichungen als Unbekannte vorliegen, also $r > n$. Dessen Diskussion wird aber später gesondert erfolgen. Wir behandeln im Augenblick nur den Fall $r = n$, der meistens vorliegt. Literaturgrundlagen für dieses Kapitel sind neben den üblichen Lehrbüchern die Titel [OR] und [DR].

6.2 Kondition des skalaren Nullstellenproblems

Gegeben sei eine Funktion $f : \mathbb{R} \to \mathbb{R}$ und x^* sei eine lokal eindeutige Nullstelle, d.h.

$$f(x^*) = 0. \tag{6.2.1}$$

Sei $\tilde{f}$ eine Störung von f, dergestalt, daß

$$|\tilde{f}(x) - f(x)| \leq \varepsilon \tag{6.2.2}$$

für alle x aus einer Umgebung $U(x^*)$ gilt. Wir fragen nun, welchen Einfluß diese Störung auf die Berechnung der Nullstelle hat. Bei der Analyse dieses Problems müssen wir berücksichtigen, daß der *relative* Fehler zu dessen Auswertung ungeeignet ist, da der Quotient

$$\frac{|\tilde{f}(x) - f(x)|}{|f(x)|}$$

im allgemeinen unbeschränkt für $x \to x^*$ ist, weil $f(x^*)$ verschwindet. Es sei $\tilde{x}^*$ eine Nullstelle von $\tilde{f}$. Dann gilt mit (6.2.2)

$$|f(\tilde{x}^*)| \leq \varepsilon. \tag{6.2.3}$$

Sei weiter m die Vielfachheit dieser Nullstelle x^*. Das bedeutet $f^{(i)}(x^*) = 0$ für alle $i = 0, \ldots, m-1$ und $f^{(m)}(x^*) \neq 0$. Mit einem ξ zwischen x^* und $\tilde{x}^*$ gilt für die TAYLOR–Entwicklung von f an x^*

$$\begin{aligned} f(\tilde{x}^*) &= \sum_{i=0}^{m-1} \frac{(\tilde{x}^* - x^*)^i}{i!} f^{(i)}(x^*) + \frac{(x^* - \tilde{x}^*)^m}{m!} f^{(m)}(\xi) \\ &\approx \frac{(x^* - \tilde{x}^*)^m}{m!} f^{(m)}(x^*). \end{aligned}$$

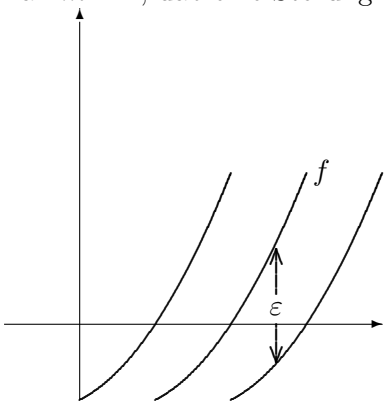
Damit erhalten wir

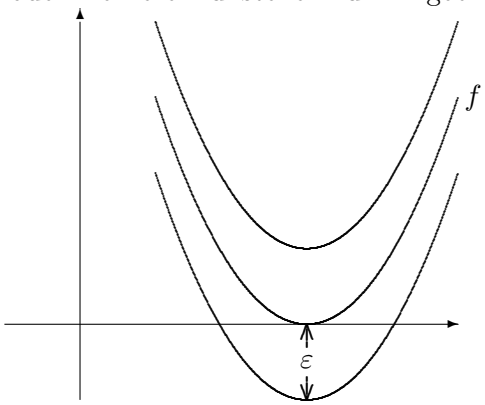
$$\left|\frac{(x^*-\tilde{x}^*)^m}{m!}f^{(m)}(x^*)\right| \approx |f(\tilde{x}^*)| \leq \varepsilon$$

nach (6.2.2) und somit gilt

$$|x^*-\tilde{x}^*| \leq \varepsilon^{\frac{1}{m}}\left|\frac{m!}{f^{(m)}(x^*)}\right|^{\frac{1}{m}}. \tag{6.2.4}$$

Hieraus lesen wir ab, daß das Problem gut konditioniert ist, wenn $m = 1$ ist, und schlecht konditioniert, falls $m > 1$ ist. Dies können wir uns an folgenden Skizzen klarmachen. Links sehen wir den Fall $m = 1$, wo eine Störung von f im Rahmen bleibt. Rechts dagegen gilt für den Fall $m = 2$, daß eine Störung von f keine oder mehrere Nullstellen zum Ergebnis haben kann.





Hierzu betrachten wir noch ein Beispiel

Beispiel 6.2.5 Seien $f(x) := (x-1)^3$ mit der Nullstelle $x^* = 1$ und $\tilde{f}(x) := (x-1)^3 - \varepsilon$ mit der Nullstelle $\tilde{x}^* = 1 + \varepsilon^{\frac{1}{3}}$. Für $\varepsilon = 10^{-12}$ ist

$$|f(x) - \tilde{f}(x)| = 10^{-12}$$

aber

$$|x^* - \tilde{x}^*| = 10^{\frac{-12}{3}} = 10^{-4},$$

was eine enorme Fehlerverstärkung bedeutet. ∎

Bei diesem Beispiel haben noch nicht beachtet, daß die Funktion f zur Lösung des Nullstellenproblems zusätzlich noch ausgewertet werden muß, wobei sich in der Regel noch Rundungsfehler einschleichen werden. Daß dies problematisch werden kann, zeigt das folgende Beispiel aus [DR].

Beispiel 6.2.6 Gegeben sei das Polynom $p(x) = x^3 - 6x^2 + 9x$, das eine doppelte Nullstelle bei $x^* = 3$ hat. Für $i = 0, \ldots, 100$ berechnen wir die Funktionswerte

$$p(3 \pm i * 10^{-9})$$

auf einer Maschine mit der Maschinengenauigkeit eps $\approx 10^{-16}$. Die Resultate sind in der unten stehenden Abbildung geplottet. Man sieht, daß die mit Rundungsfehlern behaftete Funktion $\tilde{p}$ viele Nullstellen im Intervall $[3-10^{-7}, 3+10^{-7}]$ hat, obwohl die Nullstelle $x^* = 3$ lokal eindeutig ist.

∎

6.3 Fixpunktiteration

Da man in aller Regel keine Eliminationstechniken zur Lösung von $f(x) = 0$ zur Verfügung hat, muß man nach anderen Strategien suchen. Hier bieten sich vor allem iterative Verfahren an, die sich Schritt für Schritt einer Nullstelle nähern. Die Grundidee hierbei ist, daß man (6.2.1) für eine Funktion $f : \mathbb{R}^n \to \mathbb{R}^n$ von der Form eines Nullstellenproblems

$$f(x^*) = 0$$

in die Form eines Fixpunktproblems

$$\Phi(x^*) = x^* \tag{6.3.1}$$

überführt. Um die Funktion Φ zu bestimmen, wählt man dabei den Ansatz

$$\Phi(x) := x - g(x)f(x), \tag{6.3.2}$$

wobei $g(x) \in \mathbb{R}^{n\times n}$ und $\det g(x) \neq 0$ für alle x aus einer Umgebung $U(x^*)$ sind. Grundsätzlich führt man die Iteration derart aus, daß man einen Startwert x_0 wählt und dann für $k = 0, 1, 2, \ldots$ die Werte

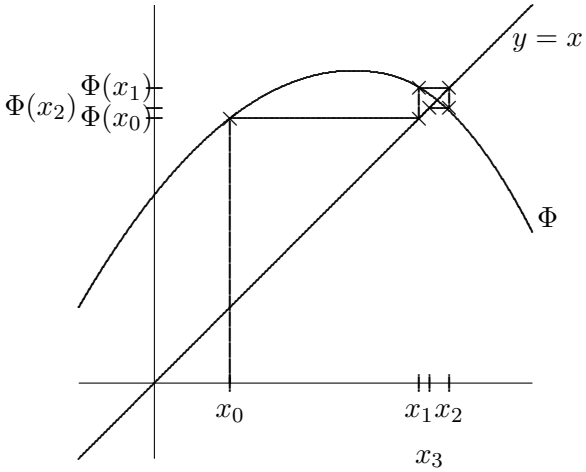
$$x_{k+1} = \Phi(x_k)$$

berechnet. Hierbei hofft man, daß bei geeigneter Wahl von x_0 die Folge x_k für $k \to \infty$ gegen x^* konvergiert. Geometrisch stellen wir uns das für $n = 1$ folgendermaßen vor: Die Aussage, Φ hat einen Fixpunkt, bedeutet, daß Φ die Gerade $y = x$ schneidet.

Hierbei ist zu beachten, daß im Fall $|\Phi'(x)| < 1$ für alle $x \in U := \overline{U(x^*)}$ wegen des Mittelwertsatzes gilt, daß für alle $x, y \in U$ ein ξ existiert mit

$$\begin{aligned} |\Phi(x) - \Phi(y)| &= |\Phi'(\xi)(x-y)| \\ &\leq \max_{t \in U} |\Phi'(t)||x-y|, \end{aligned}$$

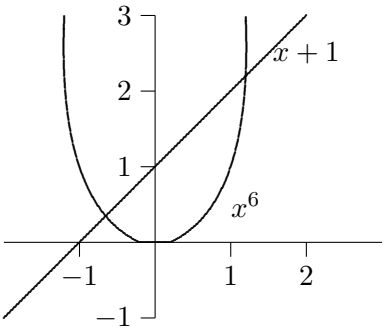
was bedeutet, daß Φ kontrahierend ist. In diesem Fall ist nennt man Φ LIPSCHITZ–stetig und bezeichnet den Term $\max_{t \in U} |\Phi'(t)|$ suggestiv mit L. Im Fall $|\Phi'(x)| > 1$ kann Φ zwar immer noch LIPSCHITZ–stetig sein, jedoch ist Φ dann keine Kontraktion mehr.



Diese Konvergenzeigenschaften wollen wir an einem Beispiel aus [DR] konkret beobachten:

Beispiel 6.3.3 Sei $f(x) := x^6 - x - 1$. Diese Funktion verschwindet genau dann, wenn $x^6 = x + 1$ ist. Aus der Grafik lesen wir ab, daß f *eine* positive Nullstelle im Intervall $[1, 2]$ haben muß. Wir testen zwei Iterationsfunktionen zur Berechnung dieser Nullstelle:

$$\begin{aligned} \Phi_1(x) &= x^6 - 1 \\ \Phi_2(x) &= (x+1)^{\frac{1}{6}}. \end{aligned}$$



Nun gilt $|\Phi_1'(x)| = |6x^5| > 1$ für alle $x \in [1, 2]$ und $|\Phi_2'(x)| = |\frac{1}{6}(x+1)^{\frac{-5}{6}}| \leq \frac{1}{6}$ für alle $x \in [0, 2]$. Mit Φ_2 können wir also eine Kontraktion mit $L = \frac{1}{6}$ erreichen. Die Werte der folgenden Tabelle geben Aufschluß über das Konvergenzverhalten der beiden Iterationsfunktionen.

k	$x_0 = 0.5$ $x_{k+1} = \Phi_2(x_k)$	$x_0 = 0.5$ $x_{k+1} = \Phi_1(x_k)$	$x_0 = 1.13$ $x_{k+1} = \Phi_1(x_k)$	$x_0 = 1.135$ $x_{k+1} = \Phi_1(x_k)$
0	0.50000000	0.50000000	1.13000000	1.14e+00
1	1.06991319	–0.98437500	1.08195175	1.14e+00
2	1.12890836	–0.09016330	0.60415884	1.17e+00
3	1.13420832	–0.99999946	–0.95136972	1.57e+00
4	1.13467844	–0.00000322	–0.25852598	1.38e+01
5	1.13472009	–1.00000000	–0.99970144	6.91e+06
6	1.13472378	0.00000000	–0.00179000	1.09e+41
7	1.13472411	–1.00000000	–1.00000000	1.69e+246

Interessanterweise divergiert bei der Funktion Φ_1 selbst der fast exakte Startwert $x_0 = 1.135$ in sehr starkem Maße. ∎

Die bisherigen, mehr heuristischen Überlegungen dieses Abschnitts werden wir nun mit dem BANACH'schen Fixpunktsatz theoretisch fundieren. Sei X ein linearer normierter Raum mit Norm $\|\cdot\|$ und $E \subseteq X$ eine vollständige Teilmenge von X. Es gelte

$$\Phi : E \to E \tag{6.3.4}$$

und

$$\|\Phi(x) - \Phi(y)\| \leq L\|x - y\| \tag{6.3.5}$$

für alle $x, y \in E$ und $L < 1$. Die Kontraktion Φ ist also auf E LIPSCHITZ–stetig mit einer LIPSCHITZ–Konstanten kleiner eins. Dann gilt

1.) Es existiert genau ein *Fixpunkt* x^* von Φ in E, also

$$\Phi(x^*) = x^*.$$

2.) Für beliebiges $x_0 \in E$ konvergiert

$$x_{k+1} = \Phi(x_k)$$

für $k = 0, 1, 2, \ldots$ gegen den Fixpunkt x^*.

3.) Es gilt die *A–priori–Fehlerabschätzung*

$$\|x_k - x^*\| \leq \frac{L^k}{1 - L}\|x_1 - x_0\|. \tag{6.3.6}$$

4.) Weiter gilt die *A–posteriori–Fehlerabschätzung*

$$\|x_k - x^*\| \leq \frac{L}{1 - L}\|x_k - x_{k-1}\|. \tag{6.3.7}$$

Beweis: Übung. ∎

Zu diesem wichtigen Theorem aus der Analysis sei noch folgendes bemerkt:

a) Ein vollständiger normierter Vektorraum heißt BANACH–Raum. Vollständigkeit bedeutet, daß jede CAUCHY–Folge in dem Raum gegen einen Grenzwert in dem Raum konvergiert. Ist dieser Grenzwert eindeutig, so spricht man auch von HAUSDORFF–Räumen. Standardbeispiele für solche Räume sind die reellen Zahlen, also $X = \mathbb{R}$, in denen jedes abgeschlossene Intervall $E = [a, b] \subset \mathbb{R}$ vollständig ist. Dies kann man verallgemeinern in n Dimensionen, womit auch jede abgeschlossene Teilmenge $E \subset \mathbb{R}^n$ vollständig ist. Der BANACH'sche Fixpunktsatz gilt aber auch für vollständige Teilmengen von unendlichdimensionalen Funktionenräume X.

b) Dieser Satz liefert neben der Existenz– und Eindeutigkeitsaussage über die Lösung einer Fixpunktgleichung auch einen *konstruktiven Algorithmus* sowie Fehlerabschätzungen, wie lange man iterieren muß, um eine gewisse vorgegebene Genauigkeit zu gewährleisten. Will man etwa die Genauigkeit ε, so genügt es wegen der A–priori–Abschätzung (6.3.6), k so groß zu wählen, daß die Gleichung

$$\frac{L^k}{1 - L}\|x_1 - x_0\| \leq \varepsilon,$$

oder nach k umgestellt

$$k \geq \frac{\log\left(\frac{\|x_1 - x_0\|}{\varepsilon(1-L)}\right)}{\log L^{-1}} \tag{6.3.8}$$

erfüllt ist.

Aus dem BANACH'schen Fixpunktsatz ziehen wir für die von uns meistens benötigte Situation noch folgendes

Korollar 6.3.9 Seien $X = \mathbb{R}^n$ und $E \subset X$ abgeschlossen und konvex, sowie $\Phi : E \to E$ stetig differenzierbar mit

$$\max_{x \in E} \|\Phi'(x)\| := L < 1.$$

Man beachte, daß in dieser Gleichung $\Phi'(x)$ natürlich für die JACOBI–Matrix

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

steht. Dann ist Φ eine Kontraktion auf E, und es gelten die Aussagen 1.) bis 4.) aus dem BANACH'schen Fixpunktsatz.

Beweis: Der Beweis läuft über den *Mittelwertsatz*. Dieser macht die Annahme der Konvexität notwendig. Damit hat man dann aber

$$\|\Phi(x) - \Phi(y)\| \leq \max_{\xi \in E} \|\Phi'(\xi)\| \cdot \|x - y\|.$$

■

Wir schließen den Abschnitt noch mit dem folgenden

Beispiel 6.3.10 Das System

$$\begin{aligned} 6x &= \cos x + 2y \\ 8y &= xy^2 + \sin x \end{aligned}$$

besitzt auf $E = [0, 1]^2$ eine eindeutige Lösung. Bestimme diese approximativ bis auf eine Genauigkeit von 10^{-3} bezüglich der Norm $\|\cdot\|_\infty$. ■

6.4 Konvergenzordnung und Fehlerschätzung

Wenn man ein Iterationsverfahren benutzt, um etwa eine Nullstelle einer nichtlinearen Gleichung zu ermitteln, so wünscht man natürlich, daß sich die Iterierten x_k schnell dem Grenzwert x^* annähern. Um ein Maß für die Geschwindigkeit eines Iterationsverfahrens zu erhalten, definieren wir die *Konvergenzordnung*.

Definition 6.4.1 (Konvergenzordnung) Eine konvergente Folge $\{x_k\}_{k \in \mathbb{N}}$ in $\mathbb{R}^n$ mit dem Grenzwert x^* hat die *Konvergenzordnung* p, falls für ein $k_0 \in \mathbb{N}$ für alle $k \geq k_0$ die Abschätzung

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

gilt, wobei $0 < c < 1$ falls $p = 1$, und $0 < c < \infty$ falls $p > 1$ ist. ■

Hierbei kann es natürlich einige Iterationen dauern, bis die Zahl k_0 sichtbar wird. Im Falle $p = 1$ spricht man von *linearer Konvergenz*. Allgemein nennt man im Fall $p > 1$ die Konvergenz *superlinear*, wobei man von *quadratischer Konvergenz* spricht, falls $p = 2$ gilt. Offenbar ist die Konvergenz um so schneller, je höher die Konvergenzordnung ist. Als Faustregel gilt, daß sich die Anzahl der korrekten Dezimalstellen bei jeder Iteration ver–p–facht ab k_0. Dieses Verhalten wollen wir uns konkret anschauen im folgenden Beispiel. Insbesondere stellen wir den Unterschied zwischen linearer und quadratischer Konvergenz zahlenmäßig dar.

Beispiel 6.4.2 Seien $\|x_0 - x^*\| = 0.2$ und $e_k := \|x_k - x^*\|$. Die folgende Tabelle gibt absolute Fehler für lineare und quadratische Konvergenz für zwei verschiedene c bei steigendem k an:

c	p	k	1	2	3	4	5	6
$c = \frac{1}{2}$	$p = 1$	$e_k \leq$	0.1	0.05	0.025	0.0125	0.00625	0.003125
$c = 3$	$p = 2$	$e_k \leq$	0.12	0.0432	0.0056	0.000094	3e-08	2e-15

■

Üblicherweise haben Fixpunktiterationen nur lineare Konvergenzordnungen. Will man Verfahren von höherer Ordnung für skalare Gleichungen konstruieren, kann man wie folgt vorgehen. Nach dem BANACH'schen Fixpunktsatz ist die Approximationsgüte umso besser, je kleiner die LIPSCHITZ–Konstante L ist. Im Hinblick auf Korollar 6.3.9, in dem $L = \max_{\xi \in E} |\Phi'(\xi)|$ war, wird L besonders klein in $U(x^*)$, wenn $\Phi'(x^*) = 0$ ist. Allgemeiner gilt folgender

Satz 6.4.3 Für $\Phi : \mathbb{R} \to \mathbb{R}$ mit $\Phi \in \mathcal{C}^{p+1}(U(x^*))$ und $j = 1, \ldots, p-1$ gelte

$$\Phi^{(j)}(x^*) = 0$$

sowie

$$\Phi^{(p)}(x^*) \neq 0.$$

Für $p = 1$ sei zusätzlich $|\Phi'(x^*)| < 1$. Dann konvergiert

$$x_{k+1} = \Phi(x_k)$$

lokal (d.h. für hinreichend kleines $|x_0 - x^*|$) von genau der Ordnung p.

Beweis: Wir betrachten lediglich den Fall $p > 1$. Für $x_k \in U := \overline{U(x^*)}$ gilt mit TAYLOR–Entwicklung von Φ um x^*

$$\begin{aligned} x_{k+1} &= \Phi(x_k) = \Phi(x^*) + \sum_{j=1}^{p} \frac{(x_k - x^*)^j}{j!} \Phi^{(j)}(x^*) + \frac{(x_k - x^*)^{(p+1)}}{(p+1)!} \Phi^{(p+1)}(\xi_k) \\ &= x^* + \frac{(x_k - x^*)^p}{p!} \Phi^{(p)}(x^*) + \ldots, \end{aligned}$$

wobei die ... für das Restglied stehen. Dies zieht

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} \leq \frac{|\Phi^{(p)}(x^*)|}{p!} + \frac{|x_k - x^*|}{(p+1)!} |\Phi^{(p+1)}(\xi_k)| \tag{6.4.4}$$

nach sich. Setze nun $c := \frac{|\Phi^{(p)}(x^*)|}{p!} + \frac{1}{(p+1)!} \max_{y \in U} |\Phi^{(p+1)}(y)|$ und $\tilde{U} := [x^* - \varepsilon, x^* + \varepsilon]$ mit $0 < \varepsilon < 1$ genügend klein, so daß $\tilde{U} \subset U$ und $c\varepsilon^{p-1} < \frac{1}{2}$ gelten. Falls nun $x_k \in \tilde{U}$ ist, folgt mit (6.4.4):

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} \leq \frac{|\Phi^{(p)}(x^*)|}{p!} + \frac{|x_k - x^*|}{(p+1)!} |\Phi^{(p+1)}(\xi_k)| \leq c.$$

Also gilt

$$\begin{aligned} |x_{k+1} - x^*| &\leq c|x_k - x^*|^p = c|x_k - x^*|^{p-1}|x_k - x^*| \\ &\leq c\varepsilon^{p-1}|x_k - x^*| \\ &< \frac{1}{2}|x_k - x^*|. \end{aligned} \tag{6.4.5}$$

Wähle nun $x_0 \in \tilde{U}$. Dann ist wegen (6.4.5) $x_k \in \tilde{U}$ für alle k und $\lim_{k \to \infty} x_k = x^*$. Also haben wir die lokale Konvergenz bereits gezeigt. (6.4.5) besagt aber auch

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

für alle k, was bedeutet, daß die Konvergenz von der Ordnung p ist. ∎

Bevor wir den Abschnitt schließen, wollen wir noch ein paar Konvergenzbegriffe diskutieren. Konvergiert $x_{k+1} = \Phi(x_k)$ für alle $k = 0, 1, 2, \ldots$ nur für Startwerte $x_0 \in U(x^*)$, so heißt das Verfahren *lokal konvergent*. Konvergiert das Verfahren aber für alle x_0 aus dem Definitionsbereich von Φ, so heißt die Konvergenz *global*. Im allgemeinen sind Verfahren aufgrund der Problemabhängigkeit aber nur lokal konvergent.

6.5 Iterationsmethoden für eine skalare Gleichung

Es sei $f : \mathbb{R} \to \mathbb{R}$ stetig differenzierbar. In diesem Abschnitt beschäftigen wir uns mit dem Lösen des zugehörigen Nullstellenproblems $f(x^*) = 0$.

6.5.1 Bisektionsverfahren

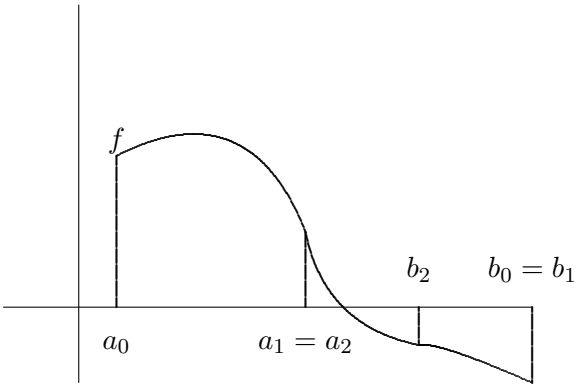
Das *Bisektions– oder Einschließungsverfahren* ist eine vom Prinzip her einfache und robuste Methode zur Lösung der vorgelegten Aufgabe. Es funktioniert nach der folgenden Idee: gilt für $a_0 < b_0$, daß $f(a_0)f(b_0) < 0$ ist, so gilt nach dem Zwischenwertsatz, daß es eine Nullstelle in $[a_0, b_0]$ gibt. Setze nun $x_0 := \frac{1}{2}(a_0 + b_0)$, und berechne $f(x_0)$. Wähle danach ein neues Intervall $[a_1, b_1] := [a_0, x_0]$ oder $[a_1, b_1] := [x_0, b_0]$ so, daß $f(a_1)f(b_1) \leq 0$ gilt.
Hiermit haben wir schon einen

Algorithmus 6.5.1 Für $k = 0, 1, 2, \ldots$ berechne

$$x_k := \frac{1}{2}(a_k + b_k)$$
$$f(x_k)$$

Setze dann

$$\begin{aligned} a_{k+1} &= a_k, \quad b_{k+1} = x_k, \quad \text{falls } f(a_k)f(x_k) \leq 0 \\ a_{k+1} &= x_k, \quad b_{k+1} = b_k, \quad \text{sonst.} \end{aligned}$$



∎

Zur Konvergenzgeschwindigkeit schätzen wir ab

$$|x^* - x_k| \leq \left(\frac{1}{2}\right)^{k+1} |b_0 - a_0|.$$

Also ist die Konvergenz linear. Als Fazit können wir also festhalten: Das *Bisektionsverfahren* ist robust, aber dafür auch relativ langsam.

6.5.2 Das NEWTON–Verfahren

Angenommen, es sei $f \in \mathcal{C}^2(U(x^*))$. Wir wollen ein Fixpunktverfahren von quadratischer Ordnung entwickeln. Nach Satz 6.4.3 gilt im Falle $\Phi'(x^*) = 0$, daß $p = 2$ sein muß. Damit setzen wir an

$$\Phi(x) := x - g(x)f(x),$$

was

$$\Phi'(x^*) = 1 - g'(x^*)f(x^*) - g(x^*)f'(x^*) = 1 - g(x^*)f'(x^*)$$

impliziert. Das bedeutet

$$\Phi'(x^*) = 0$$

ist äquivalent zu

$$g(x^*) = \frac{1}{f'(x^*)}.$$

Falls also $f(x^*) = 0$ und $f'(x^*) \neq 0$ ist, so können wir iterieren

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{6.5.2}$$

für $k = 0, 1, 2, \ldots$. Dieses Verfahren heißt NEWTON–Verfahren und konvergiert lokal quadratisch. Hierzu bemerken wir, daß dieses Verfahren nicht mehr lokal quadratisch konvergiert, wenn f in x^* eine mehrfache Nullstelle hat. Um hier trotzdem die gewünschte Konvergenz zu erreichen, muß man das Verfahren modifizieren zu

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)},$$

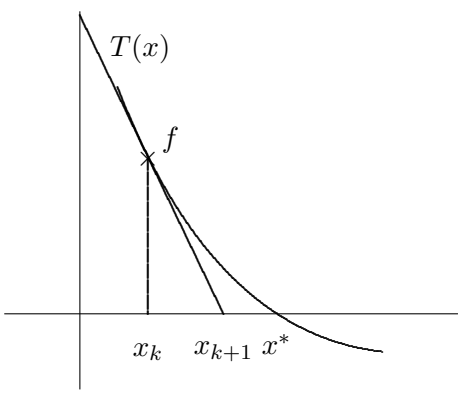
wobei m die entsprechende Vielfachheit der Nullstelle in x^* ist. Hierbei muß man natürlich beachten, daß die Nullstellen von Φ und f eine unterschiedliche Bedeutung haben.

Wir kommen zur *geometrischen Interpretation* des NEWTON–Verfahrens. Es sei x_k eine bekannte Näherung an x^*. Dann liefert die TAYLOR–Entwicklung

$$f(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{(x - x_k)^2}{2}f''(\xi_k). \tag{6.5.3}$$

Hierbei ist $T(x) := f(x_k) + (x - x_k)f'(x_k)$ die Tangente von f an x_k. Diese ist in einer kleinen Umgebung von x_k eine gute Näherung von f. Dementsprechend ist die Nullstelle von T eine gute Näherung an x^*, und die Nullstelle von $T(x_{k+1})$ liefert

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$



Zur Konvergenz beweisen wir noch folgenden

Satz 6.5.4 Sei $f \in \mathcal{C}^2(U(x^*))$ mit $f(x^*) = 0$ und $f'(x^*) \neq 0$. Dann gilt für $x_k \in U(x^*)$ und das NEWTON–Verfahren aus (6.5.3)

$$x_{k+1} - x^* = \frac{1}{2}\frac{f''(\xi_k)}{f'(x_k)}(x_k - x^*)^2$$

mit $\xi_k \in U(x^*)$. Also ist die Konvergenz lokal quadratisch.

Beweis: Setze $x = x^*$ in (6.5.3). Dann gilt

$$0 = f(x^*) = f(x_k) + (x^* - x_k)f'(x_k) + \frac{(x^* - x_k)^2}{2}f''(\xi_k),$$

was äquivalent ist zu

$$-\frac{f(x_k)}{f'(x_k)} + x_k - x^* = \frac{(x^* - x_k)^2}{2}\frac{f''(\xi_k)}{f'(x_k)},$$

woraus die Behauptung mit $\xi_k \in U(x^*)$ und der Definition von x_{k+1} aus (6.5.1) unmittelbar folgt. ∎

6.5.3 NEWTON–ähnliche Verfahren

Oftmals ist die Berechnung von f' aufwendig, insbesondere dann, wenn f nicht explizit gegeben ist. Deshalb werden wir im folgenden Verfahren betrachten, die ohne die explizite Berechnung von f' auskommen. Ein Beispiel hierfür haben wir mit dem *Bisektionsverfahren* in 6.5.1 bereits kennengelernt. Dieses war sehr robust, aber leider auch sehr langsam. Als erstes diskutieren wir das *Sekantenverfahren*. Hierbei ersetzt man f' durch die Sekante durch die Punkte $(x_{k-1}, f(x_{k-1}))$ und $(x_k, f(x_k))$, indem man $S(x)$ definiert durch

$$S(x) := \frac{x - x_{k-1}}{x_k - x_{k-1}}f(x_k) + \frac{x_k - x}{x_k - x_{k-1}}f(x_{k-1}),$$

und die neue Näherung x_{k+1} durch

$$S(x_{k+1}) = 0$$

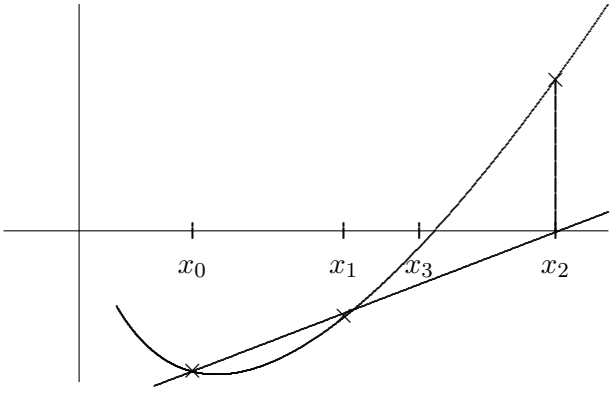
d.h.

$$\begin{aligned} x_{k+1} &= \frac{x_{k-1}f(x_k) - x_k f(x_{k-1})}{f(x_k) - f(x_{k-1})} \\ &= x_k - f(x_k)\left(\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right) \end{aligned} \tag{6.5.5}$$

bestimmt. Dieses Verfahren konvergiert lokal etwa von der Ordnung

$$p \approx 1.6.$$

Geometrisch bedeutet dies, daß man die Sekante durch die jeweiligen Punkte P_k und P_{k+1} an der Stelle $S = 0$ auswertet, und dann $f(S = 0)$ als neuen Punkt P_{k+2} nimmt, um die Sekante durch P_{k+1} und P_{k+2} zu berechnen, usw.. Als wichtigste *Eigenschaften des Sekantenverfahrens* halten wir die schnellere Konvergenz im Vergleich zum Bisektionsverfahren fest. Trotzdem ist es langsamer als das NEWTON–Verfahren und benötigt außerdem *zwei Startwerte*. Dafür kommt es aber ohne Ableitungen aus, und pro Schritt muß man nur eine Funktionsauswertung machen.



Eine weitere, bisher noch nicht in Betracht gezogene Möglichkeit ist die Kombination aus Bisektions– und Sekantenverfahren, die sogenannte *Regula falsi*. Wie bei der Bisektion macht man hier die Annahme, daß zwei Werte a_0, b_0 existieren mit der Eigenschaft

$$f(a_0)f(b_0) < 0.$$

Bei der Bisektion wurden neue Werte in der Mitte des Intervalls durch $x_0 = \frac{1}{2}(a_0+b_0)$ bestimmt, und dann $f(x_0)$ berechnet. Hier bestimmt man den neuen Wert x_0 durch die Sekante durch a_0, b_0 als

$$x_0 = \frac{a_0 f(b_0) - b_0 f(a_0)}{f(b_0) - f(a_0)}.$$

Danach wählt man das neue Intervall als $[a_1, b_1] = [a_0, x_0]$ oder $[a_1, b_1] = [x_0, b_0]$ so, daß $f(a_1)f(b_1) \leq 0$ gilt. Der zugehörige Algorithmus lautet wie folgt:

Algorithmus 6.5.6 Gegeben seien a_0, b_0 mit $f(a_0)f(b_0) < 0$. Für $k = 0, 1, 2, \ldots$ berechne

$$x_k = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$$

sowie

$$f(x_k).$$

Falls $f(x_k)f(a_k) \leq 0$ setze

$$a_{k+1} = a_k, \quad b_{k+1} = x_k.$$

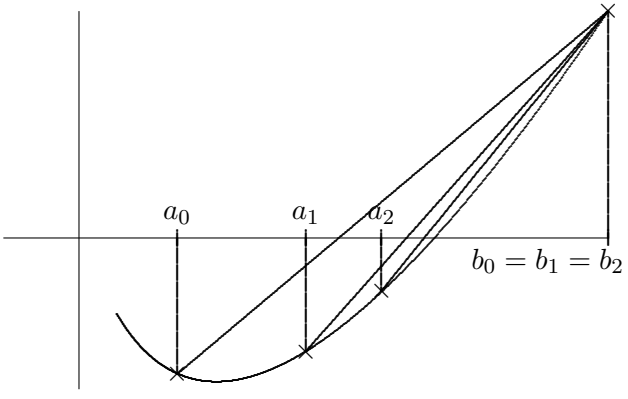
Sonst setze

$$a_{k+1} = x_k, \quad b_{+1} = b_k.$$

■

Hieraus ergibt sich, daß die Nullstelle x^* für alle k in (a_k, b_k) liegt und $a_k \to x^*$ oder $b_k \to x^*$ für $k \to \infty$ geht. Da $x^* \in (a_k, b_k)$, ist diese Methode sehr zuverlässig, und die Konvergenz ist im allgemeinen schneller als bei Bisektion, hat aber dieselbe Konvergenzordnung $p = 1$. Die zugehörige Geometrie zeigt die folgende Skizze:

Zusammenfassend noch folgende Hinweise:



- Häufig benutzt man zunächst ein robustes Verfahren wie Bisektion und danach zur Beschleunigung das NEWTON–Verfahren.
- Mit der Problemstellung ist meist ein f vorgegeben, für das das Problem $f(x) = 0$ zu lösen ist. Ein Lösungsansatz ist eine *Fixpunktiteration* mit geeignet konstruierter Iterationsfunktion Φ. Diese Methode hat im allgemeinen die Konvergenzordnung 1.
- Zur Beschleunigung versucht man generell Verfahren höherer Ordnung zu konstruieren, z.B. das NEWTON–Verfahren. Diese sind im allgemeinen aber nur *lokal* konvergent, d.h. der Erfolg der Iteration hängt wesentlich von der Wahl eines geeigneten Startwertes ab.
- Bisektion und Regula falsi sind sehr zuverlässige Verfahren.

- Das Sekantenverfahren ist eine effiziente Variante des NEWTON–Verfahrens, in der die Berechnung von f' vermieden wird.
- Zur Nullstellenberechnung von Polynomen sollte man das HORNER–Schema verwenden.

6.6 Das NEWTON–Verfahren für Systeme

Nun diskutieren wir das NEWTON–Verfahren zur Lösung von $f(x) = 0$, wobei $f : \mathbb{R}^n \to \mathbb{R}^n$ mit $n > 1$ zweimal stetig differenzierbar sei, $f(x)$ also die Form

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

hat.

6.6.1 Herleitung und Grundlagen

In diesem Kapitel bezeichnen wir den k-ten Iterationsvektor mit

$$x^k = \begin{pmatrix} x_1^k \\ \vdots \\ x_n^k \end{pmatrix} \in \mathbb{R}^n.$$

In Gleichung (6.5.3) haben wir die TAYLOR–Entwicklung für eine skalare Gleichung betrachtet. Für eine Komponente $f_i : \mathbb{R}^n \to \mathbb{R}$ kann man dies auch schreiben als

$$f_i(x) = f_i(x^k) + \sum_{j=1}^{n} (x_j - x_j^k) \frac{\partial f_i}{\partial x_j}(x^k) + O(\|x - x^k\|_2^2).$$

Bezeichnet man die JACOBI–Matrix von f mit

$$Df(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{pmatrix},$$

dann läßt sich dies mit TAYLOR–Entwicklung zusammenfassen als

$$f(x) = f(x^k) + Df(x^k)(x - x^k) + O(\|x - x^k\|_2^2). \tag{6.6.1}$$

Approximiert man nun die Nullstelle von f durch die Nullstelle x^{k+1} der linearen Approximation von f in x^k, also das 1. TAYLOR–Polynom durch

$$0 = f(x^k) + Df(x^k)(x^{k+1} - x^k),$$

dann folgt im falle $\det Df(x^k) \neq 0$

$$x^{k+1} = x^k - (Df(x^k))^{-1} f(x^k). \tag{6.6.2}$$

Dies ist das vektorwertige Analogon zu (6.5.3). Numerisch wird $(Df(x^k))^{-1}$ *nie* explizit berechnet. Der Ausdruck $(Df(x^k))^{-1} f(x^k)$ steht für die Lösung s^k eines linearen Gleichungssystems

$$Df(x^k) s^k = -f(x^k),$$

was man aus (6.6.2) mittels $s^k = x^{k+1} - x^k$ herleitet. Der Algorithmus für unseren Fall lautet

Algorithmus 6.6.3 (NEWTON–Verfahren für Systeme) Für $k = 0, 1, 2, \ldots$ berechne $f(x^k)$ und $Df(x^k)$. Löse für s^k das lineare Gleichungssystem

$$Df(x^k)s^k = -f(x^k) \tag{6.6.4}$$

und setze

$$x^{k+1} = x^k + s^k. \tag{6.6.5}$$

■

Den letzten Schritt (6.6.5) bezeichnet man auch als NEWTON–Korrektur. Je nach Struktur von f verwendet man zur Lösung von (6.6.4) ein QR– oder iteratives–Verfahren. Zur Wahl des Startwerts und der Anzahl der Iterationen kommen wir später. Wir werden nun nochmals Beispiel 6.3.10 vertiefen:

Beispiel 6.6.6 Man löse die System

$$\begin{aligned} f_1(x_1, x_2) &= 6x_1 - \cos x_1 - 2x_2 = 0 \\ f_2(x_1, x_2) &= 8x_2 - x_1 x_2^2 - \sin x_1 = 0. \end{aligned}$$

Hierzu berechnen wir

$$Df(x) = \begin{pmatrix} 6 + \sin x_1 & -2 \\ -x_2^2 - \cos x_1 & 8 - 2x_1 x_2 \end{pmatrix}.$$

Wähle nun als Startwert $x^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Dann gilt

$$f(x^0) = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad \text{und} \quad Df(x^0) = \begin{pmatrix} 6 & -2 \\ -1 & 8 \end{pmatrix}.$$

Nun lösen wir das System

$$\begin{pmatrix} 6 & -2 \\ -1 & 8 \end{pmatrix} s^0 = - \begin{pmatrix} -1 \\ 0 \end{pmatrix}.$$

Als dessen Lösung erhalten wir $s^0 = \frac{1}{46} \begin{pmatrix} 8 \\ 1 \end{pmatrix}$ und damit $x^1 = x^0 + s^0 = \frac{1}{46} \begin{pmatrix} 8 \\ 1 \end{pmatrix}$ usw. ■

Man erwartet wie im skalaren Fall (lokal) *quadratische* Konvergenz. Dies erfordert eine Reihe von Voraussetzungen, die in der Praxis nur sehr schwer zu überprüfen sind. Zur Veranschaulichung des Typs von Voraussetzungen bringen wir hier eine einfache Variante eines Konvergenzsatzes. Dazu benötigen wir die Definition einer *konvexen* Menge Ω in $\mathbb{R}^n$. Eine Menge $\Omega \in \mathbb{R}^n$ heißt konvex, wenn für alle $x, y \in \Omega$ auch

$$[x, y] := \{tx + (1 - t)y : 0 \leq t \leq 1\} \subseteq \Omega$$

ist. Diesen und weitere Sätze zur Konvergenz findet man in [S].

Satz 6.6.7 Sei $\Omega \in \mathbb{R}^n$ offen und konvex. Für ein $x^0 \in \Omega$ gebe es Konstanten $r, \alpha, \beta, \gamma, h > 0$ mit

$$B_r(x^0) := \left\{x \in \mathbb{R}^n : \|x - x^0\| < r\right\} \subseteq \Omega \tag{6.6.8}$$

$$h := \frac{\alpha\beta\gamma}{2} < 1 \tag{6.6.9}$$

$$r := \frac{\alpha}{1 - h}. \tag{6.6.10}$$

Weiter sei die Funktion $f : \Omega \to \mathbb{R}^n$ stetig differenzierbar für alle $x \in \Omega$ und habe die Eigenschaften

(a) Die JACOBI–Matrix $Df(x)$ ist LIPSCHITZ–stetig mit γ, also

$$\|Df(x) - Df(y)\| \leq \gamma \|x - y\|$$

für alle $x, y \in \Omega$.

(b) $(Df(x))^{-1}$ existiert und erfüllt

$$\|(Df(x))^{-1}\| \leq \beta$$

für alle $x \in \Omega$.

(c) Es gilt

$$\|(Df(x^0))^{-1} f(x^0)\| \leq \alpha.$$

Dann gilt:

(A) Ausgehend von x^0 ist jedes x^{k+1}, definiert durch (6.6.2),

$$x^{k+1} := x^k - (Df(x^k))^{-1} f(x^k),$$

für alle $k = 0, 1, 2, \ldots$ wohldefiniert, und $x^k \in B_r(x^0)$ für alle $k \geq 0$.

(B)

$$\lim_{k \to \infty} x^k = x^*$$

existiert und $x^* \in \overline{B_r(x_0)}$ und $f(x^*) = 0$.

(C) Für alle $k \geq 0$ gilt die Fehlerabschätzung

$$\|x^k - x^*\| \leq \alpha \frac{h^{2^k - 1}}{1 - h^{2^k}}.$$

Wegen $0 < h < 1$ ist das NEWTON–Verfahren also mindestens *quadratisch* konvergent. ∎

Bemerkung 6.6.11 (i) Falls Ω beschränkt und $f \in \mathcal{C}^2(\overline{\Omega}, \mathbb{R}^n)$ ist, dann ist Df LIPSCHITZ–stetig, d.h. (a) gilt für ein γ.

(ii) Die Parameter β und γ sind durch das Problem gegeben. Damit also (6.6.8) gilt, muß demnach α genügend klein sein. Wegen (c) beeinflußt dies die Wahl des Startwertes x^0, d.h. x^0 muß schon nahe genug an x^* liegen, denn dann ist $\|f(x^0)\|$ klein.

∎

Bevor wir zum Beweis von Satz 6.6.7 kommen, brauchen wir noch folgendes

Lemma 6.6.12 Sei $\Omega \subseteq \mathbb{R}^n$ konvex. Für alle $x \in \Omega$ existiere $Df(x)$, und es gebe eine Konstante $0 \leq \gamma < \infty$ mit

$$\|Df(x) - Df(y)\| \leq \gamma \|x - y\| \tag{6.6.13}$$

für alle $x, y \in \Omega$. Dann gilt

$$\|f(x) - f(y) - Df(y)(x - y)\| \leq \frac{\gamma}{2} \|x - y\|^2. \tag{6.6.14}$$

Beweis: Definiere für beliebige $x, y \in \Omega$ die Funktion $\Phi : [0,1] \to \mathbb{R}^n$ durch

$$\Phi(t) := f(y + t(x-y)).$$

Da f auf Ω differenzierbar ist, ist auch Φ auf $[0,1]$ differenzierbar mit Ableitung

$$\Phi'(t) = Df(y + t(x-y))(x-y).$$

Für jedes $t \in [0,1]$ folgt daher

$$\begin{aligned} \|\Phi'(t) - \Phi'(0)\| &= \|(Df(y + t(x-y)) - Df(y))(x-y)\| \\ &\leq \|Df(y + t(x-y)) - Df(y)\| \cdot \|x-y\|. \end{aligned}$$

Aufgrund von Voraussetzung (6.6.13) können wir dies nach oben durch

$$\gamma \|y + t(x-y) - y\| \cdot \|x-y\| = \gamma t \|x-y\|^2 \tag{6.6.15}$$

abschätzen. Nach Definition von Φ gilt nun

$$\begin{aligned} f(x) - f(y) - Df(y)(x-y) &= \Phi(1) - \Phi(0) - \Phi'(0) \\ &= \int_0^1 \Phi'(t)dt - \Phi'(0) \\ &= \int_0^1 (\Phi'(t) - \Phi'(0))dt. \end{aligned}$$

Nun gilt mit (6.6.15)

$$\begin{aligned} \|f(x) - f(y) - Df(y)(x-y)\| &\leq \int_0^1 \|\Phi'(t) - \Phi'(0)\|dt \\ &\leq \gamma \|x-y\|^2 \int_0^1 t dt \\ &= \frac{\gamma}{2}\|x-y\|^2 \end{aligned}$$

■

Beweis von Satz 6.6.7: Wir starten mit (A) und führen eine vollständige Induktion nach k durch. Nach Voraussetzung (b) existiert $(Df(x))^{-1}$ für alle $x \in \Omega$. Daher ist die Gleichung

$$x^{k+1} = x^k - (Df(x))^{-1} f(x^k),$$

die wir bereits in (6.6.2) hergleitet haben wohldefiniert. Wir müssen noch zeigen, daß $x^k \in B_r(x^0)$ für alle k gilt. Für $k = 0$ ist dies bereits per definitionem so. Sei $k = 1$. Dann gilt:

$$x^1 = x^0 - (Df(x^0))^{-1} f(x^0) \Leftrightarrow x^1 - x^0 = (Df(x^0))^{-1} f(x^0).$$

Hieraus folgt

$$\|x^1 - x^0\| = \|(Df(x^0))^{-1} f(x^0)\| \leq \alpha \tag{6.6.16}$$

nach Voraussetzung (c). Weiter gilt unter Ausnutzung von (6.6.8)

$$\|x^1 - x^0\| < r.$$

Also ist $x^1 \in B_r(x^0)$. Sei nun $x^j \in B_r(x^0)$ für alle $j = 0, 1, \ldots, k$, mit $k \geq 1$. Wir müssen zeigen, daß auch $x^{k+1} \in B_r(x^0)$ ist. Mit (6.6.2) können wir abschätzen

$$\|x^{k+1} - x^k\| = \|(Df(x^k))^{-1} f(x^k)\| \leq \|(Df(x^k))^{-1}\| \cdot \|f(x^k)\|.$$

Nach Voraussetzung (b) gilt aber

$$\|x^{k+1} - x^k\| \leq \beta \|f(x^k)\| \qquad (*)$$

Weiter ist die Definition von x^k in (6.6.2) äquivalent zu

$$Df(x^{k-1})(x^k - x^{k-1}) + f(x^{k-1}) = 0,$$

was eingesetzt in $(*)$ und nach Lemma 6.6.12

$$\|x^{k+1} - x^k\| \leq \beta \|f(x^k) - f(x^{k-1}) - Df(x^{k-1})(x^k - x^{k-1})\| \leq \beta \frac{\gamma}{2} \|x^k - x^{k-1}\|^2 \qquad (6.6.17)$$

ergibt. Mit Gleichung (6.6.17) zeigen wir durch vollständige Induktion, daß

$$\|x^{k+1} - x^k\| \leq \alpha h^{2^k - 1} \qquad (6.6.18)$$

ist. Der Induktionsanfang $k = 0$ ist klar mit (6.6.19). Zum Induktionsschritt von $k \to k+1$ schätzen wir ab:

$$\begin{aligned} \|x^{k+1} - x^k\| &\leq \frac{\beta\gamma}{2} \|x^k - x^{k-1}\|^2 \\ &\leq \frac{\beta\gamma}{2} \alpha^2 (h^{2^{k-1}-1})^2 \\ &= \frac{\beta\gamma}{2} \alpha\alpha (h^{2^k - 2}) = \alpha h^{2^k - 1}, \end{aligned}$$

woraus (6.6.18) folgt. Aus dieser Gleichung folgt nun aber

$$\begin{aligned} \|x^{k+1} - x^0\| &\leq \|x^{k+1} - x^k\| + \|x^k - x^{k-1}\| + \ldots + \|x^1 - x^0\| \\ &\leq \alpha (h^{2^k - 1} + h^{2^{k-1} - 1} + \ldots + h^7 + h^3 + h + 1) \\ &< \alpha \sum_{i=0}^{\infty} h^i = \frac{\alpha}{1-h} = r. \end{aligned}$$

Also ist $x^{k+1} \in B_r(x^0)$. Weiter mit (B): Aus der Abschätzung (6.6.18) folgt, daß $\{x^k\}_{k \in \mathbb{N}_0}$ eine CAUCHY–Folge ist, denn für $m \geq n$ gilt

$$\begin{aligned} \|x^{m+1} - x^n\| &\leq \|x^{m+1} - x^m\| + \ldots + \|x^{n+1} - x^n\| \\ &\leq \alpha (h^{2^n - 1} + h^{2^{n+1} - 1} + \ldots) \\ &= \alpha h^{2^n - 1} (1 + h^{2^n} + (h^{2^n})^2 + \ldots). \end{aligned}$$

Unter Ausnutzung der geometrischen Reihe erhalten wir

$$\|x^{m+1} - x^n\| < \frac{\alpha h^{2^n - 1}}{1 - h^{2^n}} < \varepsilon \qquad (6.6.19)$$

für genügend großes $n \geq N(\varepsilon)$, da $h < 1$ ist. Daher existiert der Grenzwert

$$\lim_{k \to \infty} x^k = x^*, \qquad (6.6.20)$$

und $x^* \in \overline{B_r(x^0)}$, da nach (A) alle $x^k \in B_r(x^0)$ sind. Den zweiten Teil von (B) ziehen wir gleich nach. Wir schieben hier kurz den Beweis von Behauptung (C) ein: Aus (6.6.19) folgt nämlich auch

$$\lim_{m \to \infty} \|x^{m+1} - x^n\| = \|x^* - x^n\| < \frac{\alpha h^{2^n - 1}}{1 - h^{2^n}},$$

was die Fehlerabschätzung (C) impliziert. Weiter mit (B): Wir müssen noch zeigen, daß der Grenzwert x^* in (6.6.20) Nullstelle von f in $\overline{B_r(x^0)}$ ist. Wegen Voraussetzung (a) gilt für alle $x^k \in B_r(x^0)$:

$$\|Df(x^k) - Df(x^0)\| \leq \gamma \|x^k - x^0\| < \gamma r,$$

und daher ist

$$\begin{aligned} \|Df(x^k)\| &\leq \|Df(x^k) - Df(x^0)\| + \|Df(x^0)\| < \gamma r + \|Df(x^0)\| \\ &\leq \gamma r + \|Df(x^0)\| =: K \end{aligned}$$

Aus (6.6.2) folgt nun

$$\|f(x^k)\| \leq \|Df(x^k)\| \cdot \|x^{k+1} - x^k\| \leq K \|x^{k+1} - x^k\|.$$

Also gilt

$$\begin{aligned} \lim_{k\to\infty} \|f(x^k)\| &\leq K \lim_{k\to\infty} \|x^{k+1} - x^k\| \\ &\leq K \lim_{k\to\infty} \alpha h^{2^k - 1} = 0, \end{aligned}$$

und, da f an $x^* \in \Omega$ stetig ist, auch

$$\lim_{k\to\infty} \|f(x^k)\| = \|f(x^*)\| = 0,$$

also $f(x^*) = 0$, was bedeutet, daß x^* Nullstelle von f ist, und den Beweis beendet. ∎

Unter ähnlichen Voraussetzungen kann man auch die Einzigkeit von x^* in $B_r(x^0)$ zeigen. Dies geschieht für das NEWTON–Verfahren in BANACH–Räumen in folgendem Satz von NEWTON–KANTOROVICH von 1948: Es gelten die Voraussetzungen (a), (c) aus Satz 6.6.7 und anstelle von (b) gelte

(b') $\|(Df(x^0))^{-1}\| \leq \beta$.

Da das nur für den Startwert gilt, ist diese Aussage schwächer. Definiere $h := \alpha\beta\gamma$ und

$$r_{1/2} := \frac{1 \pm \sqrt{1 - 2h}}{h} \alpha.$$

Falls $h \leq \frac{1}{2}$ und $\overline{B_{r_1}(x^0)} \subset \Omega$, gilt: Die Folge $\{x^k\}_{k\in\mathbb{N}}$, definiert durch (6.6.2) bleibt in $B_{r_1}(x^0)$ und konvergiert gegen die einzige Nullstelle von f in $\Omega \cap B_{r_2}(x^0)$. Einen Beweis hierzu findet man z.B. in [OR], Theorem 12.6.2, p. 421.

6.6.2 Hinweise zur praktischen Durchführung des NEWTON–Verfahrens

Oft ist es in der Praxis sehr aufwendig, für jede Iterierte k die Ableitung $Df(x^k)$ zu berechnen. Wenn sich diese Matrix im Laufe der Iteration nur wenig verändert, kann man folgenden Algorithmus anwenden:

Algorithmus 6.6.21 (Vereinfachtes NEWTON–Verfahren) Berechne $Df(x^0) =: A$. Für $k = 0, 1, 2, \ldots$ löse für s^k:

$$As^k = -f(x^k), \tag{6.6.22}$$

und setze dann

$$x^{k+1} = x^k + s^k.$$

∎

Dies hat allerdings den entscheidenden Nachteil, daß man die quadratische Konvergenz verliert. Daher verwendet man in der Praxis eine Mischform, indem man etwa alle drei bis fünf Schritte die Matrix A erneuert. Dies ist vor allem bei schwachen Nichtlinearitäten ein geeignetes Verfahren.

Auswertung der JACOBI–Matrix

Oft sind die Einträge von Df nicht oder nur mit großem Aufwand geschlossen berechenbar. In solchen Fällen wendet man eine numerische Differentiation an, was bedeutet, daß man die Ableitungen durch Differenzenquotienten ersetzt, also

$$\frac{\partial f_i}{\partial x_j}(x) \approx \frac{f_i(x + he^j) - f_i(x)}{h}.$$

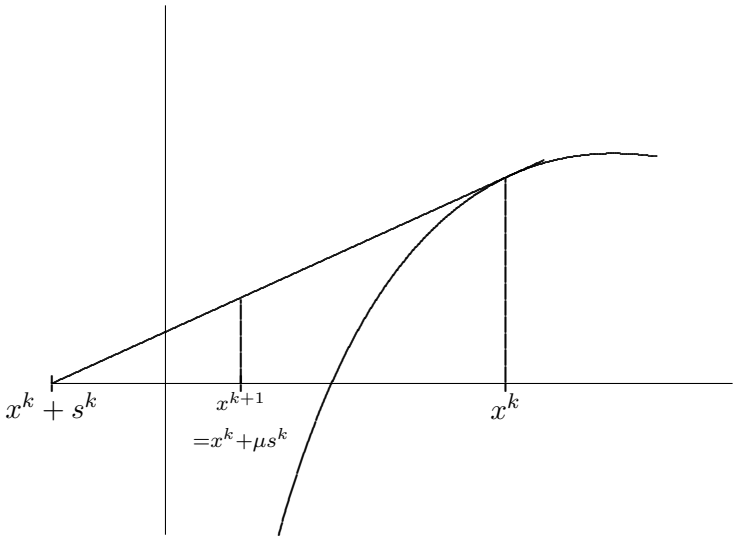
Hierbei hängt die für h anzusetzende Größe vom Problem ab. Bei zu großer Wahl von h, ist die Approximation von Df zu ungenau, was eine langsame Konvergenz zur Folge hat. Wählt man h zu klein, so besteht die Gefahr von Auslöschungen.

Wahl des Startwertes

Hierbei sollte man zur Hauptsache Hintergrundinformationen verwenden. Ist dies nicht möglich, so wendet man das *Homotopieverfahren* an, bei dem man versucht, anstelle von $f(x) = 0$ die Gleichung $f(x, \lambda) = 0$ zu lösen, in der λ für einen Parameter steht, und das Problem dadurch, etwa für $\lambda = 1$, auf ein lineares zurückgeführt wird. Diese Lösung wendet man dann als Startwert für kompliziertere Probleme. Öfters findet man auch in der ursprünglichen Gleichung bereits Parameter, die als solche Homotopieparameter verwendet werden können. Dieses Vorgehen hat allerdings den Nachteil, daß es theoretisch nicht abgesichert ist. Eine Idee, um hier Abhilfe zu schaffen, ist, einen Algorithmus zu finden, der sein Konvergenzverhalten *selbst* beeinflußt. Dies geschieht mit folgendem Grundgedanken: Der Korrekturschritt $s^k = -(Df(x))^{-1} f(x^k)$ des klassischen NEWTON–Verfahrens liefert die Richtung, in der f abnimmt. Oft ist es günstiger, nur einen Teil des Schritts in diese Richtung zu gehen. Man setze etwa

$$x^{k+1} = x^k + \mu s^k \tag{6.6.23}$$

mit passendem $\mu \in (0, 1]$. Dies läßt sich als eine Art *Dämpfung* interpretieren. Graphisch könnte das etwa so aussehen:



Dies wirft natürlich die Frage nach der Wahl von μ auf. Unser Ziel ist es, $f(x) = 0$ als Maß für die Abweichung von $\|\!| f(x^k) |\!\|$ mit einer geeigneten Norm $\|\!| \cdot |\!\|$ zu betrachten. Daher versuchen wir, μ in jedem Schritt so zu wählen, daß

$$\|\!| f(x^0) |\!\| > \|\!| f(x^1) |\!\| > \|\!| f(x^2) |\!\| > \dots \tag{6.6.24}$$

gilt. Zusätzlich ist zu beachten, daß die Lösung x^* von $f(x) = 0$ auch Lösung von $Bf(x) = 0$ für jedes nichtsinguläre $B \in \mathbb{R}^{n\times n}$ ist. Dies läßt sich als eine Art Umskalierung interpretieren, denn die NEWTON–Iteration ist unabhängig von B. Dies macht man sich etwa durch den Monotonietest

$$\|(Df(x^k))^{-1}f(x^{k+1})\|_2 < \|(Df(x^k))^{-1}f(x^k)\|_2 \tag{6.6.25}$$

klar. Denn wenn man in (6.6.25) anstelle von f die umskalierte Funktion Bf verwendet, kürzt sich diese durch die Bildung der Inversen wieder heraus. Weiter ist zu bemerken, daß die rechte Seite in (6.6.25) in der NEWTON–Iteration bereits vorhanden ist. Die linke Seite erfordert noch die Lösung von

$$Df(x^k)v^k = f(x^{k+1}).$$

Dies ist nicht zu viel zusätzlicher Aufwand, wenn man die QR– oder LR–Zerlegung von $Df(x^k)$ bereits kennt. Eine geeignete Norm für $|||\cdot|||$ ist

$$|||f(x^{k+1})||| := \|(Df(x^k))^{-1}f(x^{k+1})\|_2. \tag{6.6.26}$$

Damit haben wir

Algorithmus 6.6.27 (Gedämpftes NEWTON–Verfahren) Setze $k = 0$.

1.) berechne $s = s^k$ über (6.6.4) $Df(x^k)s^k = -f(x^k)$ und setze $\mu = 1$.

2.) Dämpfungsschritt: setze $x = x^k + \mu s^k$ und prüfe, ob

$$|||f(x)||| \le (1 - \frac{\mu}{4})|||f(x^k)|||. \tag{6.6.28}$$

Falls ja, setze

$$x^{k+1} = x^k + \mu s^k$$

und gehe zu 3.).
sonst ersetze μ durch $\frac{\mu}{2}$. Falls noch $\mu > 2^{-10}$, gehe zum Anfang von 2.), sonst Abbruch.

3.) neuer Iterationsschritt: Ist $k >$ vorgegebene Schrittzahl oder $|||f(x^{k+1})||| <$ tol, Ende, sonst setze $k = k + 1$ und gehe zu 1.) zurück.

Zum Ende noch folgende Bemerkungen

a) Dieses Verfahren enthält zwei ineinander geschachtelte Schleifen, eine äußere über k wie beim klassischen NEWTON–Verfahren, und eine innere über $\mu = 1, \frac{1}{2}, \frac{1}{4}, \ldots$. Bei der Implementierung ist es daher wichtig, beide Schleifen zu begrenzen.

b) Es reicht wegen der Rundungsfehler nicht, statt (6.6.28) nur

$$|||f(x^k + \mu s^k)||| < |||f(x^k)|||$$

zu testen.

7 Nichtlineare Ausgleichsrechnung

In Kapitel 4 haben wir in Gleichung (4.1.4) versucht, eine Modellbeschreibung

$$b(t) = \varphi(t, x_1, \ldots, x_n)$$

aus den Daten (t_i, b_i) für $i = 1, \ldots, m$ zurückzugewinnen. Dabei galt üblicherweise $m \geq n$ und $b_i \approx b(t_i)$. Eine Lösungsmethode hierbei war die GAUSS'sche Fehlerquadratmethode, bei der man nach (4.1.5) die Werte x_i so bestimmte, daß galt

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

In Kapitel 4 hing die Modellfunktion φ stets *linear* von den x_i ab, womit wir diese Abhängigkeit auch als

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

schreiben konnten. Damit war die Lösung durch

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$$

mit $b \in \mathbb{R}^n$ gegeben.

Jetzt wollen wir Probleme betrachten, in denen φ nicht linear von den x_i abhängt. Dies wirft natürlich die Frage auf, ob und wenn ja, wie sich ein solches Problem modellieren läßt. Mit $F(x) = F(x_1, \ldots, x_n)$ und $F_i(x_1, \ldots, x_n) := b_i - \varphi(t_i, x_1, \ldots, x_n)$ läßt sich das Problem (4.1.5) in unseren Fall als

$$\|F(x)\|_2^2 = \min_{x \in \mathbb{R}^n} \tag{7.1}$$

schreiben. Nun stellen wir uns folgende Aufgabe: Seien $F \in \mathcal{C}^2(\Omega, \mathbb{R}^m)$ und $\Omega \subset \mathbb{R}^n$ offen. Setze $F(x) := f(x) - b$, wobei $f \in \mathcal{C}^2(\Omega, \mathbb{R}^m)$ und $b \in \mathbb{R}^m$ seien. Finde $x^* \in \Omega$ mit der Eigenschaft

$$\|F(x^*)\|_2 = \inf_{x \in \Omega} \|F(x)\|_2. \tag{7.2}$$

Zur Lösung wenden wir die Strategie Rückführung auf lineare Ausgleichsprobleme an. Sei dazu x^k eine „gute" Startnäherung. Die TAYLOR–Entwicklung von F um x^k liefert

$$F(x) \approx F(x^k) + Df(x^k)(x - x^k).$$

Ersetze nun F in Gleichung (7.2) durch eine lineare Approximation. Finde dazu ein $s^k \in \mathbb{R}^n$, so daß

$$\|F(x^k) + Df(x^k)s^k\|_2 = \min_{x \in \mathbb{R}^n} \|F(x^k) + Df(x^k)x\|_2 \tag{7.3}$$

gilt und setze

$$x^{k+1} = x^k + s^k. \tag{7.4}$$

Hiermit haben wir bereits

Algorithmus 7.5 (GAUSS–NEWTON–Verfahren) 1.) Wähle Startwert x^0 und Toleranz ε.

2.) Für $k = 0, 1, 2, \ldots$
berechne $f(x^k), Df(x^k)$ und

$$\begin{aligned} 2(Df(x^k))^{-T} F(x^k) &= \operatorname{grad}\left(F(x)^T F(x)\right)\big|_{x=x^k} \\ &= \operatorname{grad}\left(\|F(x)\|_2^2\right)\big|_{x=x^k}. \end{aligned}$$

3.) Ist $\|2(Df(x^k))^T F(x^k)\| \leq \varepsilon$, stop.

4.) Löse lineares Ausgleichsproblem (7.3).

5.) Setze $x^{k+1} = x^k + s^k$ und gehe zu 2.) zurück.

■

Zu diesem Algorithmus sei noch angemerkt:

- Das Abbruchkriterium in Schritt 3.) bedeutet, daß man ein lokales Minimum gefunden haben könnte. Dieser kritische Punkt könnte aber trotzdem auch nur ein Sattelpunkt sein.
- Dämpfungsstrategien können wie in Algorithmus 6.6.27 verwendet werden.
- Konvergenzsätze für dieses Verfahren sind aus Satz 6.6.7 ableitbar.
- Ein alternatives Verfahren ist das LEVENBERG–MARQUARDT–Verfahren, welches das Problem eines weit vom Minimum entfernten Iterationswertes x^k aufgreift. Hier ist die Linearisierung s^k sicherlich nicht mehr akzeptabel. In einem solchen Fall löst man das Ersatzproblem
$$\min_{x \in \mathbb{R}^n} \|F(x^k) + Df(x^k)x\|_2$$
unter der Nebenbedingung
$$\|x\|_2 \leq \delta,$$
für ein geeignetes δ. Mittels der LAGRANGE'schen Multiplikatorenregel (Penalty Term) wird dies mit $p > 0$ zu
$$\min_{x \in \mathbb{R}^n} \|F(x^k) + Df(x^k)x\|_2^2 + p\|x\|_2^2.$$
Aus der notwendigen Bedingung für ein lokales Minimum leiten wir somit die Normalengleichung
$$(DF(x^k)^T DF(x^k) + pI)s^k = -DF(x^k)^T F(x^k)$$
ab. Hierbei ergibt sich nun noch das Problem der Anpassung von p an Rangdefekte. Wer sich für weitere Details zum Thema dieses Kapitels interessiert, sei auf die Werke [H] und [DH] verwiesen.

8 Interpolation mit Polynomen

8.1 Vorbemerkungen

In diesem Kapitel wollen wir Probleme der folgenden Art angehen:

Aufgabe 8.1.1 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

$$g_n(x_i) = f_i \tag{8.1.2}$$

für alle $i = 0, \ldots, n$. ∎

Der entscheidende Unterschied zu Kapitel 4 ist die in (8.1.2) geforderte Gleichheit, was die Interpolation von der Approximation trennt. Daher sagt man auch, daß die Funktionswerte in (8.1.2) interpoliert werden. Ein Verfahren hierzu, welches wir auch im folgenden diskutieren werden, ist etwa die LAGRANGE–Interpolation. Wird diese Interpolation mittels Polynomen durchgeführt, ist also $G_n \in \mathcal{P}_n$, so nennt man dies auch *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 9 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 8.1.1 gestellte Aufgabe muß nicht nur auf Funktionswerte beschränkt sein. Es können ebenfalls Ableitungen als Stützwerte vorgeschrieben werden. Hier verwendet man dann die HERMITE–Interpolation.

8.2 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 8.2.1 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

$$\mu_i(f) = \mu_i(g_n) \tag{8.2.2}$$

für alle $i = 0, \ldots, n$ sei. ∎

Ein Beispiel haben wir in (8.1.2) mit $\mu_i(f) = f(x_i)$ bereits gesehen. Dort wurde eine LAGRANGE–Interpolation verlangt. Die Bedingungen für eine HERMITE–Interpolation könnten etwa

$$\begin{aligned} \mu_0(f) &= f(x_0); \quad \mu_1(f) = f'(x_0) \\ \mu_2(f) &= f(x_1); \quad \mu_3(f) = f'(x_1) \end{aligned}$$

lauten. Nun wollen wir die Existenz und Eindeutigkeit der Lösung von 8.2.1 zeigen.

Satz 8.2.3 Sei $\{\varphi_0, \ldots, \varphi_n\}$ eine Basis von G_n. Aufgabe 8.2.1 besitzt genau dann eine eindeutige Lösung, wenn

$$\det(\mu_i(\varphi_j))_{i,j=0,\ldots,n} \neq 0 \tag{8.2.4}$$

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 8.2.1 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 8.2.1 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 8.2.5 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* $\{x^j, j = 0, \ldots, n\}$. Nun gilt mit dieser Basis

$$\mu_i(\varphi_j)_{i,j=0,\ldots,n} = V_n = \begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix}, \tag{8.2.6}$$

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 (8.2.4). 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 8.2.7 (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

$$\mu_j(f) = f^{(l)}(x_j) \tag{8.2.8}$$

mit $l = \max\{r : x_j = x_{j-r}\}$ und hinreichend glattem f auf $[x_0, x_n]$ gegeben. Dann existiert ein eindeutiges Polynom

$$P_n(x) := P(f|x_o, \ldots, x_n)(x) \in \mathcal{P}_n$$

mit der Eigenschaft

$$\mu_j(P_n) = \mu_j(f) \tag{8.2.9}$$

für alle $j = 0, \ldots, n$.

Man interpretiert die Interpolation von Ableitungen formal durch zusammenfallende Stützstellen.

Beweis: Die Aussage folgt aus dem Fundamentalsatz der Algebra. ∎

8.3 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

$$\mu_i(f) = f(x_i). \tag{8.3.1}$$

Wir beginnen mit der LAGRANGE–Darstellung:

Lemma 8.3.2 Das (eindeutige) Interpolationspolynom $P_n(x) := P(f|x_0, \ldots, x_n) \in \mathcal{P}_n$, das

$$\mu_j(P_n) = P_n(x_j) = f(x_j) \tag{8.3.3}$$

für alle $j = 0, \ldots, n$ erfüllt, läßt sich explizit als

$$P_n(x) = \sum_{j=0}^{n} f(x_j) l_{jn}(x) \tag{8.3.4}$$

schreiben, wobei

$$l_{jn}(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} \tag{8.3.5}$$

für $j = 0, \ldots, n$ die LAGRANGE–Fundamentalpolynome sind.

Beweis: Aufgrund der gewählten Darstellung sind die $l_{jn} \in \mathcal{P}_n$ für alle $j = 0, \ldots, n$. Weiter gilt $l_{jn}(x_i) = \delta_{ij}$. Daraus folgt

$$P_n(x_i) = \sum_{j=0}^{n} f(x_j) l_{jn}(x_i) = f(x_i).$$

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 (8.3.4) 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

$$P(f|x_0, \ldots, x_n)(x) = \sum_{j=0}^{n} a_j x^j \tag{8.3.6}$$

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 n 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 8.3.7 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

$$P_n(x) = P_{n-1}(x) + \delta_n\omega_n(x) \tag{8.3.8}$$

mit

$$\omega_n(x) := \prod_{i=0}^{n-1}(x - x_i) \in \mathcal{P}_n \tag{8.3.9}$$

und

$$\delta_n := \frac{f(x_n) - P_{n-1}(x_n)}{\omega_n(x_n)}. \tag{8.3.10}$$

■

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 8.2.7, 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 (8.3.8) von f und den x_i abhängt. Daher hat sich auch die Schreibweise

$$\delta_n = [x_0,\ldots,x_n]f \quad \text{oder auch} \quad \delta_n = f[x_0,\ldots,x_n] \tag{8.3.11}$$

eingebürgert. Eine wiederholte Anwendung der Argumentation aus Lemma 8.3.7 liefert uns die NEWTON'sche Interpolationsformel

$$P(f|x_0,\ldots,x_n)(x) = \sum_{i=0}^{n}([x_0,\ldots,x_i]f)\omega_i(x) \tag{8.3.12}$$

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 8.3.13 (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 8.3.14 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 (8.3.12) in (8.3.13) ein und führe einen Koeffizientenvergleich durch. ∎

Aus Gleichung (8.3.11) wissen wir

$$[x_i]f = f(x_i). \tag{8.3.15}$$

Damit erhalten wir das folgende Schema zur Berechnung der dividierten Differenzen:

	0	1	2	3	$\cdots$
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	$[x_3]f$				
$\vdots$					

(8.3.16)

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 8.3.17 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
$$[x_0,\ldots,x_n](gh)=\sum_{i=0}^{n}([x_0,\ldots,x_i]g)([x_i,\ldots,x_n]h).$$

■

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 (8.3.16). Zu dessen Auswertung verwende das Horner–Schema. Damit haben wir folgenden

Algorithmus 8.3.18 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 (8.3.16).

2.) Setze $p = \delta_n$ für $k = n-1, \ldots, 0$ und berechne

$$p = \delta_k + (x - x_k)p.$$

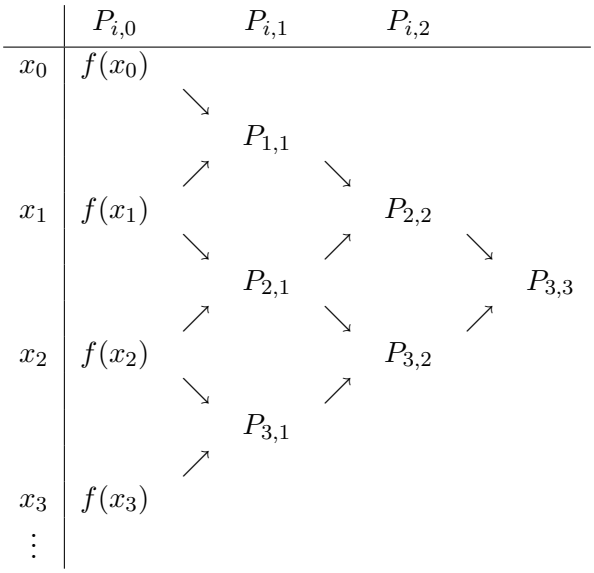
Dann ist $P_n(x) = p$.

■

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 8.3.13: 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 8.3.13 gilt dann

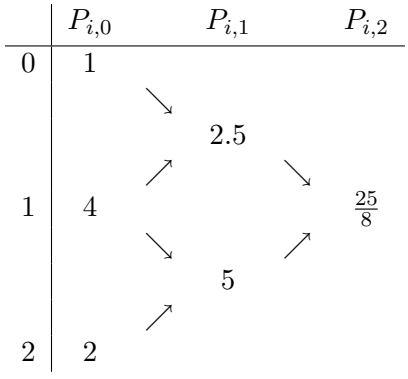
$$P_{i,k} = \frac{x - x_{i-k}}{x_i - x_{i-k}} P_{i,k-1}(x) + \frac{x_i - x}{x_i - x_{i-k}} P_{i-1,k-1}(x),$$

was die Neville'sche Interpolationsformel ist. Damit haben wir das Schema von Neville und Aitken:



(8.3.19)

Dies entspricht einem Aufwand von etwa n^2 Multiplikationen 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 $f(2) = 2$.

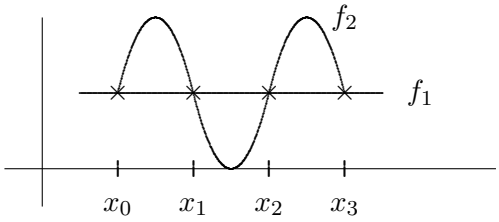


Für die lineare Interpolation, also $p \in \mathcal{P}_1$ benötigt man mit $n = 1$ 2 Stützstellen und für die kubische mit $p \in \mathcal{P}_3$ und $n = 3$ 4 Stützstellen.

8.4 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 8.3.17(c), daß

$$[x_0, \ldots, x_n]f = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Eine zentrale Abschätzung liefert der folgende

Satz 8.4.1 Sei $f \in \mathcal{C}^{n+1}([x_0, x_n])$. Dann existiert ein $\xi \in [x_0, x_n]$ mit

$$f(x) - P(f|x_0, \ldots, x_n)(x) = \omega_{n+1}(x)\frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{8.4.2}$$

Insbesondere gilt

$$\|f - P(f|x_0, \ldots, x_n)\|_\infty \leq \|\omega_{n+1}\|_\infty \frac{1}{(n+1)!}\|f^{(n+1)}\|_\infty, \tag{8.4.3}$$

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 (8.4.3) 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 (8.4.3). 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 8.4.4 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.

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)T_m(x)dx = \left\{ \begin{array}{ll} 0 & \text{falls } n \neq m \\ \pi & \text{falls } n = m = 0 \\ \frac{\pi}{2} & \text{falls } n = m \neq 0 \end{array} \right\}.$$

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 8.4.5 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*

$$\text{co}\,(x_0, \ldots, x_n, x),$$

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 8.4.6 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.

8.5 Grenzen der Polynominterpolation

Auf den ersten Blick suggeriert Satz 8.4.1, 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 $\|\cdot\|_\infty$.

9 Splinefunktionen

9.1 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 Spline 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

$$\int_a^b \frac{S''(x)}{(1+S'(x)^2)^{3/2}}dx$$

minimal wird. Unter Vernachlässigung des Nenners müssen wir eine interpolierende Funktion finden, für die

$$S(x_i) = f_i \tag{9.1.1}$$

für $i = 1, \ldots, n$ und $x_i \in [a, b]$ gilt, und zusätzlich das Glattheitsmaß

$$\int_a^b S''(x)^2 dx \tag{9.1.2}$$

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 in (9.1.1) unter der Bedingung (9.1.2) 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.

9.2 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 $\Delta = \{\tau_i\}_{i=0,\ldots,l+1}$ mit $l+2$ paarweise verschiedenen Knoten

$$a = \tau_0 < \tau_1 < \ldots < \tau_{l+1} = b \tag{9.2.1}$$

den *Splineraum*

$$S_{k,\Delta} = \{S \in \mathcal{C}^{k-2}([a, b]) : S\big|_{[\tau_i,\tau_{i+1})} \in \Pi_k \quad \text{für alle } i = 0, \ldots, l\}. \tag{9.2.2}$$

Im Falle $k = 4$ spricht man von *kubischen Splines*, welche den Polynomen vom Grad drei und $S \in \mathcal{C}^2$ entsprechen. Ist $k = 2$, so nennt man die zugehörigen Splines *linear*. Dies sind offenbar die Polynome vom Grad eins und $S \in \mathcal{C}^0$. Sie entsprechen den Polygonzügen.

Wir gehen nun der Frage der Dimension von $S_{k,\Delta}$ nach. Wir geben uns auf $[a, \tau_1)$ 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 $k-1$ 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 $i = 1, \ldots, l$ gewählt hat gilt

Bemerkung 9.2.3 Es gilt dim $S_{k,\Delta} = k + l$, und $S_{k,\Delta}$ ist ein linearer Raum über $\mathbb{R}$. ∎

Wir machen weiter mit der Frage nach einer Basis für $S_{k,\Delta}$. Dazu stellen wir zunächst fest, daß $\mathcal{P}_{k-1} \subset S_{k,\Delta}$, also $\Pi_k \subset S_{k,\Delta}$. Für *abgebrochene Potenzen* gilt

$$(x-\tau_i)_+^{k-1} := \left\{ \begin{array}{cc} (x-\tau_i)^{k-1} & \text{falls } x > \tau_i \\ 0 & \text{sonst} \end{array} \right\}. \tag{9.2.3a}$$

Damit haben wir die Frage beantwortet.

Satz 9.2.4 Die Menge

$$\{x^i \quad \text{für} \quad i = 0, \ldots, k-1 \quad \text{und} \quad (x-\tau_j)_+^{k-1} \quad \text{für} \quad j = 1, \ldots, l\} \tag{9.2.5}$$

bildet eine Basis für $S_{k,\Delta}$.

Beweis: Nach 9.2.3 gilt: dim $S_{k,\Delta} = k + l$. In (9.2.5) sind $k + l$ Funktionen. Diese erzeugen mit den Argumenten von oben den Raum $S_{k,\Delta}$. Wir müssen noch zeigen, daß diese linear unabhängig sind. Dazu sei $S \in S_{k,\Delta}$ mit

$$S(x) = \sum_{i=0}^{k-1} a_i x^i + \sum_{j=1}^{l} b_j (x-\tau_j)_+^{k-1} = 0 \tag{9.2.6}$$

für alle $x \in [a, b]$. Wir haben 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 $r = 1, \ldots, l$ und τ_r^+, τ_r^- als rechts– bzw. linksseitige Grenzwerte auf S an. Dann gilt

$$0 = \mu_r(S) = \mu_r \left(\sum_{i=0}^{k-1} a_i x^i \right) + \sum_{j=1}^{l} b_j \mu_r((x-\tau_j)_+^{k-1}) = b_r.$$

Aufgrund der Differenz und der $(k-1)$–ten Ableitung auf $\mathcal{P}_{k-1}$ verschwindet der erste μ_r–Term in obiger Gleichung. Also folgt $S(x) = 0$ für alle $x \in [a, b]$, also $a_i = 0$, für $i = 0, \ldots, k-1$.

∎

Allerdings ist die in (9.2.5) angegebene Basis für praktische Zwecke ungeeignet, denn die Basisfunktionen sind nicht lokal. Weiter sind für Werte τ_i und τ_{i+1} nahe beieinander abgebrochene Potenzen

$$(x-\tau_i)_+^{k-1}, \quad (x-\tau_{i+1})_+^{k-1}$$

fast linear abhängig, d.h. die Auswertung eines Splines S in der Darstellung bezüglich (9.2.5) und (9.2.6) ist schlecht konditioniert bezüglich Störungen in den Koeffizienten b_i. Weiter haben die Koeffizienten keine geometrische Bedeutung. Daher wird es im weiteren Aufgabe sein, eine geeignete Basis für $S_{k,\Delta}$ zu konstruieren.

9.3 B–Splines

Zur Motivation beginnen wir mit einem Beispiel.

Beispiel 9.3.1 Seien $k = 1$ und

$$S_{1,\Delta}\{S : S\big|_{[\tau_i,\tau_{i+1})} \in \Pi_1, \quad \text{für} \quad i = 0,\dots,l\}$$

stückweise konstant. Eine Basisfunktion für diesen Raum könnte die charakteristische Funktion

$$\chi_{[\tau_i,\tau_{i+1})}(x) = \left\{ \begin{array}{ll} 1 & \text{falls } x \in [\tau_i,\tau_{i+1}) \\ 0 & \text{sonst} \end{array} \right\}$$

sein. Für den Fall $k = 2$ und

$$S_{2,\Delta} = \{S \in \mathcal{C}^0([a,b]) : S\big|_{[\tau_i,\tau_{i+1})} \in \Pi_2 \quad \text{für} \quad i = 0,\dots,l\}$$

könnte eine Basis aus Hutfunktionen bestehen. ∎

Verallgemeinerungen dieser Basen sind die folgenden, siehe [Bo] für eine umfassende Darstellung.

Definition 9.3.2 Für eine Knotenfolge $\Delta = \{\tau_i\}_{i=0,\dots,l+1}$ mit der Eigenschaft (9.2.1) definieren wir die *B–Splines* $N_{i,k}(x)$ der Ordnung k bezüglich $\tau_i,\dots,\tau_{i+k}$ für $k = 1,\dots,l$ und $i = 0,\dots,l-k+1$ rekursiv durch

$$\begin{aligned} N_{i,1}(x) &= \chi_{[\tau_i,\tau_{i+1})}(x) = \left\{ \begin{array}{ll} 1 & \text{falls } x \in [\tau_i,\tau_{i+1}) \\ 0 & \text{sonst} \end{array} \right\} \\ N_{i,k}(x) &= \frac{x-\tau_i}{\tau_{i+k-1}-\tau_i} N_{i,k-1}(x) + \frac{\tau_{i+k}-x}{\tau_{i+k}-\tau_{i+1}} N_{i+1,k-1}(x) \end{aligned} \tag{9.3.3}$$

∎

B–Splines haben folgende nützliche

Eigenschaften 9.3.4 Für die in (9.3.3) definierten B–Splines gilt

(a) (lokaler Träger) $\operatorname{supp} N_{i,k} \subseteq [\tau_i,\tau_{i+k}]$;

(b) (Nichtnegativität) $N_{i,k}(x) \geq 0$ für alle $x \in [a,b]$,

(c) $N_{i,k}(x)$ ist ein stückweises Polynom von der Ordnung k bezüglich $[\tau_j,\tau_{j+1})$.

∎

Die rekursive Darstellung in (9.3.3) ist günstig für die praktische Auswertung, allerdings ist eine geschlossene Darstellung für theoretische Zwecke oft geeigneter. Diese wollen wir nun bestimmen.

Lemma 9.3.5 Die in (9.3.3) definierten B-Splines lassen sich als

$$N_{i,k}(x) = (\tau_{i+k} - \tau_i)\left([\tau_i,\dots,\tau_{i+k}](\cdot - x)_+^{k-1}\right) \tag{9.3.6}$$

geschlossen darstellen, wobei $[\tau_i,\dots,\tau_{i+k}]f$ die durch (8.3.14) definierten dividierten Differenzen sind.

Beweis: Wir führen den Beweis über Induktion nach k. Für den Induktionsanfang sei $k = 1$. Es gilt

$$\begin{aligned} N_{i,1}(x) &= (\tau_{i+1} - \tau_i)\left([\tau_i, \tau_{i+1}](\cdot - x)_+^0\right) \\ &= (\tau_{i+1} - \tau_i)\frac{[\tau_{i+1}](\cdot - x)_+^0 - [\tau_i](\cdot - x)_+^0}{\tau_{i+1} - \tau_i} \\ &= (\tau_{i+1} - x)_+^0 - (\tau_i - x)_+^0 \\ &= \chi_{(-\infty,\tau_{i+1})} - \chi_{(-\infty,\tau_i)} \\ &= \chi_{[\tau_i,\tau_{i+1})}. \end{aligned}$$

Sei die Aussage für $k-1$ bereits bewiesen. Wir führen den Induktionsschritt von $k-1 \to k$ durch. Dazu formen wir $N_{i,k}$ in die Darstellung (9.3.6) um. Es ist

$$\begin{aligned} N_{i,k}(x) &= (\tau_{i+k} - \tau_i)[\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-1} \\ &= (\tau_{i+k} - \tau_i)[\tau_i, \ldots, \tau_{i+k}]\left((\cdot - x)(\cdot - x)_+^{k-2}\right) \\ &= (\tau_{i+k} - \tau_i)\left[\sum_{j=i}^{i+k}[\tau_i, \ldots, \tau_j](\cdot - x)[\tau_j, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2}\right] \\ &= (\tau_{i+k} - \tau_i)\Big[[\tau_i](\cdot - x)[\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} \\ &\qquad + [\tau_i, \tau_{i+1}](\cdot - x)[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} + \ldots\Big] \\ &= (\tau_{i+k} - \tau_i)\Big[(\tau_i - x)[\tau_i, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} \\ &\qquad + \frac{(\tau_{i+1} - x) - (\tau_i - x)}{\tau_{i+1} - \tau_i}[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2}\Big] \\ &= (\tau_{i+k} - \tau_i)\left[(\tau_i - x)\left(\frac{[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} - [\tau_i, \ldots, \tau_{i+k-1}](\cdot - x)_+^{k-2}}{\tau_{i+k} - \tau_i}\right)\right. \\ &\qquad \left. + [\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2}\right] \\ &= (\tau_i - x)\left([\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} - [\tau_i, \ldots, \tau_{i+k-1}](\cdot - x)_+^{k-2}\right) \\ &\qquad + (\tau_{i+k} - \tau_i)[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} \\ &= (\tau_i - x + \tau_{i+k} - \tau_i)[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} - (\tau_i - x)[\tau_i, \ldots, \tau_{i+k-1}](\cdot - x)_+^{k-2} \\ &= (\tau_{i+k} - x)[\tau_{i+1}, \ldots, \tau_{i+k}](\cdot - x)_+^{k-2} - (\tau_i - x)[\tau_i, \ldots, \tau_{i+k-1}](\cdot - x)_+^{k-2} \\ &= \frac{x - \tau_i}{\tau_{i+k-1} - \tau_i}N_{i,k-1}(x) + \frac{\tau_{i+k} - x}{\tau_{i+k} - \tau_{i+1}}N_{i+1,k-1}(x) \end{aligned}$$

■

Bemerkung 9.3.7 Obige Rekursionsformeln (9.3.3) lassen sich genauso angeben, wenn Stützstellen 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 9.3.8 Die Rekursionsformeln für die Ableitungen von B–Splines erhält man direkt aus der expliziten Darstellung (9.3.6):

$$\begin{aligned} N'_{i,k}(x) &= (k-1)(\tau_{i+k}-\tau_i)\left([\tau_i,\ldots,\tau_{i+k}](\cdot-x)_+^{k-2}\right)(-1) \\ &= -(k-1)(\tau_{i+k}-\tau_i)\left(\frac{[\tau_{i+1},\ldots,\tau_{i+k}](\cdot-x)_+^{k-2}-[\tau_i,\ldots,\tau_{i+k-1}](\cdot-x)_+^{k-2}}{\tau_{i+k}-\tau_i}\right) \\ &= -(k-1)\left[\frac{1}{\tau_{i+k}-\tau_{i+1}}N_{i+1,k-1}(x)-\frac{1}{\tau_{i+k-1}-\tau_i}N_{i,k-1}(x)\right] \\ &= (k-1)\left[\frac{N_{i,k-1}(x)}{\tau_{i+k-1}-\tau_i}-\frac{N_{i+1,k-1}(x)}{\tau_{i+k}-\tau_{i+1}}\right], \end{aligned}$$

d.h. die Berechnung der Ableitungen reduziert sich auf die Auswertung einer Linearkombination von B–Splines niedrigerer Ordnung. ∎

Auswertung von Splinefunktionen

Auf der Basis von B–Splines lassen sich nun Splinefunktionen angeben. Betrachte dazu die erweiterte Knotenfolge

$$T := \{\theta_i\}_{i=1,\ldots,n+k} \tag{9.3.9}$$

mit $\theta_i < \theta_{i+k}$ für $i = 1,\ldots,n$, also

$$\theta_1 = \ldots = \theta_k = a < \theta_{k+1} \le \ldots \le \theta_n < b = \theta_{n+1} = \ldots = \theta_{n+k}.$$

Definiere dazu das lineare Erzeugnis der B–Splines auf T

$$\mathcal{N}_k(T) = \mathcal{N}_{k,T} := \operatorname{span}\{N_{i,k} : i = 1,\ldots,n\}, \tag{9.3.10}$$

d.h. jedes Element $S \in \mathcal{N}_k(T)$ besitzt eine Darstellung

$$S(x) = \sum_{i=1}^{n} c_i N_{i,k}(x) \tag{9.3.11}$$

für $x \in [a,b]$. Durch Einsetzen der Rekursionsformel (9.3.3) bekommt man daraus

$$S(x) = \sum_{i=r+1}^{n} c_i^{[r]}(x) N_{i,k-r}(x), \tag{9.3.12}$$

wobei

$$c_i^{[r]} = \left\{ \begin{array}{cc} c_i & \text{falls } r = 0 \\ \frac{x-\theta_i}{\theta_{i+k-r}-\theta_i}c_i^{[r-1]}(x) + \frac{\theta_{i+k-r}-x}{\theta_{i+k-r}-\theta_i}c_{i-1}^{[r-1]}(x) & \text{falls } r > 0 \\ 0 & \text{falls } \theta_{i+k-r} = \theta_i \end{array} \right\}. \tag{9.3.13}$$

Speziell für den Fall $r = k-1$ folgt für $x \in [\theta_i, \theta_{i+1})$ aus (9.3.12)

$$N_{i,k-r}(x) = N_{i,1}(x) = \chi_{[\theta_i,\theta_{i+1})}$$

und damit

$$S(x) = c_i^{[k-1]}(x) \tag{9.3.14}$$

für $x \in [\theta_i, \theta_{i+1})$. Zur rekursiven Berechnung der $c_i^{[r]}(x)$ bietet sich ein NEVILLE–artiges Schema an:

$$\begin{array}{ccccccccc}
c_{i-k+1} & & & & & & & & \\
 & \searrow & & & & & & & \\
c_{i-k+2} & \rightarrow & c_{i-k+2}^{[1]}(x) & & & & & & \\
 & \searrow & & \searrow & & & & & \\
c_{i-k+3} & \rightarrow & c_{i-k+3}^{[1]}(x) & \rightarrow & c_{i-k+3}^{[2]} & & & & \\
\vdots & & & & & & & & \\
 & \searrow & & \searrow & & \cdots & \searrow & & \\
c_i & \rightarrow & c_i^{[1]}(x) & \rightarrow & c_i^{[2]}(x) & \cdots & \rightarrow & c_i^{[k-1]} &
\end{array} \tag{9.3.15}$$

Der Aufwand zur Auswertung von S entspricht dem zur Auswertung *eines* B–Splines nach (9.3.3). Wir müssen noch zeigen, daß die $N_{i,k}$ tatsächlich eine Basis für den Splineraum 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

$$\begin{aligned}
(x-\sigma)^{k-1} &= \sum_{i=1}^{n} \prod_{j=1}^{k-1} (\theta_{i+j} - \sigma) N_{i,k}(x) \\
&= \sum_{i=1}^{n} \varphi_{i,k}(\sigma) N_{i,k}(x)
\end{aligned} \tag{9.3.16}$$

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

$$1 = \sum_{i=1}^{n} 1 \cdot N_{i,1}(x)$$

gilt. Wir führen den Induktionsschritt von $k-1 \rightarrow k$ aus. Dazu nehmen wir an, daß die Behauptung für $r \leq k-1$ gilt. Betrachte die rechte Seite in (9.3.16):

$$\begin{aligned}
\sum_{i=1}^{n} \varphi_{i,k}(\sigma) N_{i,k}(x) &= \sum_{i=2}^{n} c_i^{[1]}(x) N_{i,k-1}(x) \\
&= \sum_{i=2}^{n} \left(\frac{x-\theta_i}{\theta_{i+k-1}-\theta_i} c_i + \frac{\theta_{i+k-1}-x}{\theta_{i+k-1}-\theta_i} c_{i-1} \right) N_{i,k-1}(x) \\
&= \sum_{i=2}^{n} \left(\frac{x-\theta_i}{\theta_{i+k-1}-\theta_i} \prod_{j=1}^{k-1} (\theta_{i+j}-\sigma) \right. \\
&\qquad \left. + \frac{\theta_{i+k-1}-x}{\theta_{i+k-1}-\theta_i} \prod_{j=1}^{k-1} (\theta_{i-1+j}-\sigma) \right) N_{i,k-1}(x) \\
&= \sum_{i=2}^{n} \prod_{j=1}^{k-2} (\theta_{i+j}-\sigma) \left(\frac{x-\theta_i}{\theta_{i+k-1}-\theta_i} (\theta_{i+k-1}-\sigma) \right. \\
&\qquad \left. + \frac{\theta_{i+k-1}-x}{\theta_{i+k-1}-\theta_i} (\theta_i-\sigma) \right) N_{i,k-1}(x) \\
&= (x-\sigma) \sum_{i=2}^{n} \varphi_{i,k-1}(\sigma) N_{i,k-1}(x) \\
&= (x-\sigma)^{k-1}.
\end{aligned}$$

■

Einige Konsequenzen hieraus sind

Korollar 9.3.17 Die Menge der Polynome vom Grad höchstens $k-1$ auf $[a,b]$ ist eine Untermenge von span $\{N_{i,k}, i = 1, \ldots, n\}$ 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:

$$\begin{aligned}
\left(\frac{d}{d\sigma} \right)^{\ell} (x-\sigma)^{k-1} \big|_{\sigma=0} &= \left((k-1)\cdots(k-l)(x-\sigma)^{k-l-1}(-1)^l \right) \big|_{\sigma=0} \\
&= (k-1)\cdots(k-l) x^{k-l-1}(-1)^l \\
&= \sum_{i=1}^{n} \varphi_{i,k}^{(l)}(0) N_{i,k}(x)
\end{aligned}$$

Aus der letzten Identität folgt mit $m = k-l-1$ folgende Darstellung

$$x^m = \frac{(-1)^{k-m-1}}{(k-1)\cdots(m+1)} \sum_{i=1}^{n} \varphi_{i,k}^{(k-m-1)}(0) N_{i,k}(x) \tag{9.3.18}$$

für Monome mit $m = 0, \ldots, k-1$, d.h. die Monome lassen sich als Linearkombination von $N_{,k}$ darstellen. Da $\mathcal{P}_{k-1}$ sich wiederum als Linearkombination von Monomen darstellen läßt, folgt die Behauptung. ■

Speziell für $m = 0$ folgt aus (9.3.18)

$$\sum_{i=1}^{n} N_{i,k}(x) = 1$$

für alle $x \in [a, b]$, was bedeutet, daß die B–Splines eine *Zerlegung der Eins* bilden. Aus Satz 9.2.4 wissen wir, daß die Menge

$$\{x^i, i = 0, \ldots, k-1, \quad \text{und} \quad (x - \tau_j)_+^{k-1}, j = 0, \ldots, l\}$$

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

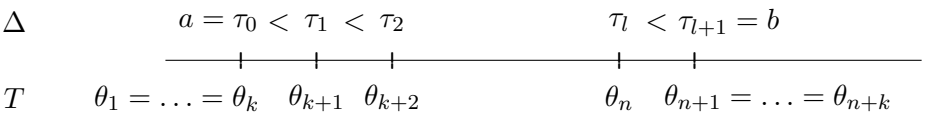
Satz 9.3.19 Die $N_{i,k}$ sind lokal linear unabhängig, d.h. gilt

$$\sum_{i=1}^{n} c_i N_{i,k}(x) = 0$$

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 (9.3.18) und Korollar 9.3.17 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 $S_{k,\Delta}$ bilden. Dazu müssen wir die Stützstellen und Knoten geeignet wählen. In Erinnerung an (9.2.1) ergibt sich das folgende Schema:



Damit haben wir

Satz 9.3.20 Mit Δ, T wie oben gilt

$$S_{k,\Delta} = \mathcal{N}_{k,T},$$

d.h. die B–Splines der Ordnung k bilden eine Basis für den Raum der Splinefunktionen $S_{k,\Delta}$.

Beweis: Wegen 9.3.4 (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

$$\mathcal{N}_{k,T} \subseteq S_{k,\Delta}.$$

Aufgrund der lokalen linearen Unabhängigkeit der $N_{i,k}$ gilt

$$\dim \mathcal{N}_{k,T} = n = k + l = \dim S_{k,\Delta},$$

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 9.3.21 Sei $S_{k,\mu,\Delta} = \mathcal{N}_{k,T}$. Ist $\mu_j = 1$ für alle $j = 1, \ldots, l$, so gilt

$$S_{k,\mu,\Delta} = S_{k,\Delta}.$$

■

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 9.3.22 (Stabilität der B–Spline Basis) Man kann zeigen, daß

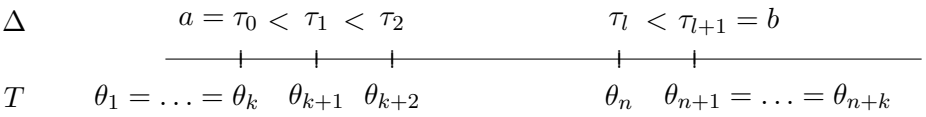
$$C_k \|\underline{c}\|_\infty \leq \left\| \sum_{i=1}^{n} c_i N_{i,k} \right\|_{\infty,[a,b]} \leq \|\underline{c}\|_\infty.$$

D.h. die B–Splines bilden eine *unkonditionell stabile Basis* für $S_{k,\Delta}$, denn unabhängig von der Knotenfolge T läßt sich $S = \sum c_i N_{i,k}$ 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. ■

9.4 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



Mit $\mathcal{N}_{k,T} = \operatorname{span}\{N_{i,k}, i = 1, \ldots, n\}$ 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 9.4.1 Sei $T = \{\theta_i\}_{i=1,\ldots,n+k}$ wie oben und seien $x_1 < \ldots < x_n \in [a, b]$ Stützstellen. Das Problem

$$\text{Finde zu Daten } f_1, \ldots, f_n \text{ ein } S \in \mathcal{N}_{k,T} \text{ mit } S(x_i) = f_i \text{ für } i = 1, \ldots, n \tag{9.4.2}$$

besitzt genau dann eine eindeutige Lösung, wenn

$$x_i \in (\theta_i, \theta_{i+k}) \tag{9.4.3}$$

für alle $i = 1, \ldots, n$, d.h. wenn in den Träger jedes B–Splines genau eine Stützstelle fällt.

Beweis: Nach Satz 8.2.3 ist die Interpolationsbedingung genau dann eindeutig lösbar, wenn $\det\left((N_{i,k}(x_j))_{i,j=1,\ldots,n}\right) \neq 0$ 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, denn es gilt

$$x_i \in (\theta_i, \theta_{i+k}) \quad \Longleftrightarrow \quad N_{i,k}(x_i) > 0.$$

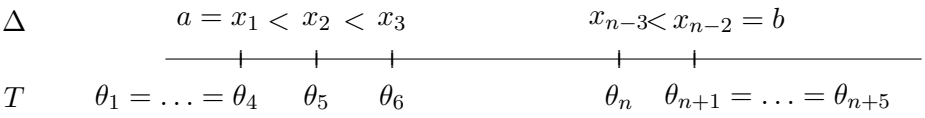
■

Bemerkung 9.4.4 Man kann sogar zeigen, daß im 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 $A\underline{c} = \underline{f}$ 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 $A\underline{c} = \underline{f}$ her:

Kubische Spline–Interpolation mit B–Splines

Für $k = 4$ wandelt sich unser Schema zu:



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:

$$S(\theta_i) = f(\theta_i) \tag{9.4.2'}$$

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 9.4.5 Zu jedem $f \in \mathcal{C}^2([a,b])$ existiert ein eindeutiger Spline $S = I_4 f \in \mathcal{N}_{4,T}$, so daß für $i = 4, \ldots, n+1$

$$\begin{aligned} (I_4 f)(\theta_i) &= f(\theta_i), \\ (I_4 f)'(a) &= f'(a) \\ (I_4 f)'(b) &= f'(b), \end{aligned} \tag{9.4.6}$$

gilt. Des weiteren erfüllt $I_4 f$ die *Extremaleigenschaft*

$$\|(I_4 f)''\|_2 \le \|g''\|_2$$

für alle $g \in \mathcal{C}^2([a,b]) \cap L_2([a,b])$, die auch (9.4.6) erfüllen. Weiter gelten die Fehlerabschätzungen

$$\|f - I_4 f\|_{\infty,[a,b]} \leq \frac{5}{384} h^4 \|f^{(4)}\|_{\infty,[a,b]} \tag{9.4.7}$$

für $f \in \mathcal{C}^4([a,b])$, wobei $h = \max_i |\theta_{i+1} - \theta_i|$ ist.

Beweis: Die Existenz und Eindeutigkeit zeigt man mit Satz 8.2.3. Die Extremaleigenschaft findet man etwa in [DH], Kapitel 7.4. Ein Beweis der Fehlerabschätzungen steht in [SB2], 2.4. ∎

Zusätzlich beachte man noch, daß das Einfügen von Stützstellen in (9.4.7) bewirken kann, daß h kleiner wird und damit die Abschätzung genauer. Weiter ist (9.4.7) unabhängig von der Lage der Stützstellen im Unterschied zur Polynominterpolation, bei der die rechte Seite von (8.4.3)

$$\|\omega_n\|_\infty \frac{1}{n} \|f^{(n)}\|_\infty$$

lautete.

Berechnung der vollständigen kubischen Spline–Interpolation mit B–Splines

Im folgenden wird die Berechnung von $S = I_4 f$ mit den Interpolationsbedingungen (9.4.2')

$$S(\theta_i) = f(\theta_i)$$

für $i = 4, \ldots, n+1$ und

$$S'(a) = f'(a), \quad S'(b) = f'(b)$$

durchgeführt. Die gesuchte Funktion $S \in \mathcal{N}_{4,T}$ wird durch die Lösung des Gleichungssystems

$$S(\theta_j) = \sum_{i=1}^{n} c_i N_{i,k}(\theta_j) = f(\theta_j) \tag{9.4.8}$$

für $j = 4, \ldots, n+1$ und zwei Ableitungsbedingungen ermittelt. Nach der Rekursionsformel gilt $N_{j,4}(\theta_4) = 0$ für $j = 2, 3, \ldots$ und $N_{j,4}(\theta_{n+1}) = 0$ für $j = n-1, n-2, \ldots$ (d.h. die B–Splines $N_{j,4}$ für $j \neq 1$ und $j \neq n$ verschwinden am Rand). Da die B–Splines wie in Kapitel 9.3 gezeigt, eine Partition der Eins bilden, gilt $N_{1,4}(\theta_4) = 1$ und $N_{n,4}(\theta_{n+1}) = 1$. Daher gilt für die Koeffizienten c_1 und c_n in der Darstellungsformel (9.4.8)

$$\begin{aligned} S(\theta_4) &= c_1 = f(\theta_4) \\ S(\theta_{n+1}) &= c_n = f(\theta_{n+1}). \end{aligned}$$

Somit müssen in (9.4.8) nur noch die $n-2$ Koeffizienten $c_2, \ldots, c_{n-1}$ berechnet werden. Mit gleicher Argumentation schließt man aus den Bedingungen an die Ableitung und aus den Rekursionsformeln in Bemerkung 9.3.8 für die Ableitung von B–Splines

$$\begin{aligned} c_2 &= \frac{f'(a) - f(a) N'_{1,4}(a)}{N'_{2,4}(a)} \\ c_{n-1} &= \frac{f'(b) - f(b) N'_{n,4}(b)}{N'_{n-1,4}(b)}, \end{aligned}$$

so daß nur noch die $n-4$ Koeffizienten $\tilde{\underline{c}} := (c_3, \ldots, c_{n-2})^T$ zu bestimmen sind. D.h. löse das Gleichungssystem

$$\tilde{A}\tilde{\underline{c}} = \tilde{f},$$

wobei

$$\tilde{A} = \begin{pmatrix} N_{3,4}(\theta_5) & N_{4,4}(\theta_5) & & 0 & \\ N_{3,4}(\theta_6) & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & N_{n-2,4}(\theta_{n-1}) \\ & 0 & & N_{n-3,4}(\theta_n) & N_{n-2,4}(\theta_n) \end{pmatrix}$$

und $\tilde{f} = (f_3, \ldots, f_{n-2})^T$ mit $f_3 = f(\theta_5) - c_2 N_{2,4}(\theta_5)$, $f_j = f(\theta_{j+2})$ für $j = 4, \ldots, n-3$ und $f_{n-2} = f(\theta_n - c_{n-1} N_{n-1,4}(\theta_n)$.

9.5 Mehrdimensionale Splines

Die Theorie zu mehrdimensionalen Splines läßt sich nicht annähernd so einheitlich entwickeln und darstellen wie die univariate Splinetheorie.

Als Beispiel wird hier der einfachste Fall von Tensorproduktansätzen auf Rechteckgittern für zwei Variablen x, y diskutiert. Dazu seien zwei Knotenfolgen $T = \{\theta_i\}_{i=1,\ldots,n+k}$ für x und $U = \{u_j\}_{j=1,\ldots,m+k}$ für y für die Intervalle $[a, b]$ und $[c, d]$ gegeben, d.h. Knotenpunkte (θ_i, u_j) für $i = 1, \ldots, n+k$ und $j = 1, \ldots, m+k$. Setze nun

$$N_{i,j}(x, y) := N_{i,k,T}(x) N_{j,k,T}(y),$$

womit ein Splineraum

$$\mathcal{N}_{k,T,U} := \operatorname{span}\{N_{i,j}(x, y) : \ i = 1, \ldots, n; \quad j = 1, \ldots, m\}$$

mit Dimension nm definiert wird. Die Interpolationsaufgabe lautet nun: Gegeben seien Stützstellen (x_r, y_s) und Daten f_{rs} für $r = 1, \ldots, n$ und $s = 1, \ldots, m$. Finde

$$S(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} N_{ij}(x, y) \in \mathcal{N}_{k,T,U},$$

so daß $S(x_r, y_s) = f_{rs}$ gilt. Dies führt auf ein lineares Gleichungssystem

$$A\underline{c} = \underline{f} \tag{9.5.1}$$

mit $A \in \mathbb{R}^{nm \times nm}$ aus

$$\begin{aligned} & \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} N_{ij}(x_r, y_s) = f_{rs} \\ \iff & \sum_{j=1}^{m} \left(\sum_{i=1}^{n} c_{ij} N_{i,k,T}(x_r) \right) N_{j,k,U}(y_s) = f_{rs} \end{aligned}$$

für $r = 1, \ldots, n$ und $s = 1, \ldots, m$. Setzt man nun $a_j^r := \sum_{i=1}^{n} c_{ij} N_{i,k,T}(x_r)$, so kann man die Lösung dieses Systems auf eindimensionale Probleme wie folgt reduzieren. Bestimme zunächst für alle $r = 1, \ldots, n$ die Koeffizienten a_j^r als Lösung des linearen Gleichungssystems

$$\sum_{j=1}^{m} a_j^r N_{j,k,U}(y_s) = f_{rs}$$

für $s = 1, \ldots, m$. D.h. man hat n eindimensionale Probleme vom Rang m zu lösen. Löse dann noch für $j = 1, \ldots, m$

$$\sum_{i=1}^{n} c_{ij} N_{i,k,T}(x_r) = a_j^r$$

für $r = 1, \ldots, n$, also m Probleme vom Rand n. Insgesamt braucht man also $mO(n) + nO(m)$ Operationen, da die einzelnen Systeme aus Bandmatrizen bestehen, die jeweils in $O(n)$ bzw. $O(m)$ Operationen gelöst werden können. Zwar ist $A \in \mathbb{R}^{nm \times nm}$ auch dünn besetzt, aber die Verwendung von direkten Lösern würde ein *fill–in* erzeugen und man könnte das Gleichungssystem nicht mehr in $O(nm)$ Operationen lösen.

Allgemein werden Spline–Interpolationen häufig auf Dreiecksgittern angewandt. Hier gilt jedoch

$$N_{i,j}(x, y) \neq \tilde{N}_i(x)\tilde{N}_j(y),$$

so daß die Lösung des Systems in (9.5.1) mit iterativen Methoden ermittelt werden muß.

10 Iterative Lösung linearer Gleichungssysteme

10.1 Motivation und Einführung

Zur Lösung des Systems

$$Ax = b$$

mit nichtsingulärem $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$ wurden in Kapitel 3 mit LR–, QR– und CHOLESKY–Zerlegung drei direkte Verfahren vorgestellt. Diese sind bedingt durch ihren Rechenaufwand von jeweils rund $O(n^3)$ Multiplikationen besonders geeignet für vollbesetze Matrizen ohne besondere Struktur, aber dennoch nur für verhältnismäßig kleine Matrizen. Für sehr große und darüberhinaus noch dünn besetzte Matrizen sind „billigere“ Verfahren mit einem optimalen Rechenaufwand von $O(n)$ zur Ermittelung der n Unbekannten wünschenswert. Solche Matrizen entstehen beispielsweise bei der mehrdimensionalen Spline–Interpolation auf Δ–Gittern oder bei der Diskretisierung partieller Differentialgleichungen für $d > 1$. Typisch hierfür sind Matrizen der Form

$$A = \begin{pmatrix} * & & & * & & 0 \\ & * & & & \ddots & \\ & & \ddots & 0 & & * \\ * & & 0 & \ddots & & \\ & \ddots & & & * & \\ 0 & & * & & & * \end{pmatrix}.$$

Oftmals müssen lineare Gleichungssysteme nur bis zu einer bestimmten Genauigkeit gelöst werden, die z.B. durch die Diskretisierung einer Differentialgleichung bestimmt ist, so daß sich auch aus diesem Grunde iterative Verfahren anbieten.

10.2 Klassische Iterationsverfahren

Zur Entwicklung eines iterativen für lineare Gleichungssysteme schreibt man ähnlich wie im nichtlinearen Fall die Gleichung

$$Ax = b \tag{10.2.1}$$

mit nichtsingulärem $A \in \mathbb{R}^{n\times n}$ als Fixpunktgleichung

$$x = x + C(b - Ax) =: \Phi(x) \tag{10.2.2}$$

um, wobei $C \in \mathbb{R}^{n\times n}$ nichtsingulär geeignet zu wählen ist. Damit erhält man für $k = 0, 1, 2, \ldots$ eine Iteration der Form

$$x^{k+1} = \Phi(x^k) = x^k + C(b - Ax^k) = (I - CA)x^k + Cb. \tag{10.2.3}$$

Mit der Notation $\rho(M) := |\lambda_{\max}(M)|$ für den Spektralradius einer Matrix $M \in \mathbb{R}^{n\times n}$ gilt folgender

Satz 10.2.4 Das Verfahren (10.2.3) konvergiert genau dann für jeden Startwert x^0 gegen die Lösung von (10.2.1), wenn

$$\rho(I - CA) < 1 \tag{10.2.5}$$

gilt.

Beweis: Zur Notwendigkeit von (10.2.5) nehmen wir an, es gäbe einen Eigenwert λ von $I-CA$ mit $|\lambda| \geq 1$. Wähle den Startwert x^0 nun gerade so, daß $x^0 - x$ Eigenvektor zu λ ist. Mit (10.2.2) und (10.2.3) folgt dann

$$\begin{aligned} x^{k+1} - x &= (I-CA)(x^k - x) = \dots \\ &= (I-CA)^{k+1}(x^0 - x) \\ &= \lambda^{k+1}(x^0 - x), \end{aligned}$$

was

$$\|x^{k+1} - x\| = |\lambda|^{k+1}\|x^0 - x\|$$

impliziert. Ist nun $|\lambda| \geq 1$, so kann die Folge $\{x^k\}_k$ nicht konvergieren. Für die Umkehrung sei daran erinnert, daß zu jeder Matrix M und jedem $\varepsilon > 0$ stets eine Vektornorm konstruiert werden kann, so daß für die zugehörige Matrixnorm $\|\cdot\|_\varepsilon$

$$\|M\|_\varepsilon \leq \rho(M) + \varepsilon$$

gilt. Wählt man speziell $\varepsilon = \frac{1}{2}(1-\rho(I-CA)) > 0$, so erhält man

$$\begin{aligned} \|I-CA\|_\varepsilon &\leq \rho(I-CA) + \frac{1}{2}(1-\rho(I-CA)) \\ &= \frac{1}{2} + \frac{1}{2}\rho(I-CA) =: \tilde{\rho} < 1. \end{aligned}$$

Damit gilt für jeden Startwert x^0 aber

$$\begin{aligned} \|x^{k+1} - x\|_\varepsilon &= \|(I-CA)^{k+1}(x^0 - x)\|_\varepsilon \leq \|(I-CA)^{k+1}\|_\varepsilon \|x^0 - x\|_\varepsilon \\ &\leq \|I-CA\|_\varepsilon^{k+1}\|x^0 - x\|_\varepsilon \leq \tilde{\rho}^{k+1}\|x^0 - x\|_\varepsilon \\ &\to 0 \text{ für } k \to \infty. \end{aligned}$$

∎

Bemerkung 10.2.6 Da $\Phi'(x) = I - CA$ ist, ist mit (10.2.5) gerade sichergestellt, daß Φ eine Kontraktion ist, so daß der zweite Teil des Beweises auch aus dem BANACH'schen Fixpunktsatz aus Kapitel 6.3 folgt.

Korollar 10.2.7 Falls irgendeine Vektornorm existiert, so daß für die zugehörige Matrixnorm $\|\cdot\|$

$$\|I-CA\| < 1 \tag{10.2.8}$$

gilt, konvergiert (10.2.3) für jeden Startwert x^0. Ferner gelten die *A–priori–Fehlerabschätzung*

$$\|x^k - x\| \leq \frac{\|I-CA\|^k}{1-\|I-CA\|}\|x^1 - x^0\|$$

und die *A–posteriori–Fehlerabschätzung*

$$\|x^k - x\| \leq \frac{\|I-CA\|}{1-\|I-CA\|}\|x^k - x^{k-1}\|.$$

∎

(10.2.5) und (10.2.8) verlangen, daß C die Inverse von A approximiert, denn für $C = A^{-1}$ wäre man $I - CA = 0$. In diesem Fall wäre man sofort fertig. Aber die Berechnung von A^{-1} ist aufwendiger als die Lösung von (10.2.1). Es geht also um folgenden Balanceakt: finde eine Matrix C so, daß C die Matrix A^{-1} genügend gut approximiert, andererseits aber die Berechnung und Anwendung von nur einen Aufwand von $O(n)$ erfordert. Dies ist schwierig, da im allgemeinen auch für dünn besetzte Matrizen A die Inverse A^{-1} vollbesetzt ist. Verschiedene Wahlen von C führen auf unterschiedliche Methoden.

Gesamtschrittverfahren (JACOBI–Verfahren)

Sei $a_{ii} \neq 0$ für $i = 1, \ldots, n$. Wähle

$$C = (\mathrm{diag}(a_{11}, \ldots, a_{nn}))^{-1} =: D^{-1}.$$

Mit dieser Wahl lautet die Iterationsvorschrift (10.2.3)

$$x^{k+1} = (I - D^{-1}A)x^k + D^{-1}b. \tag{10.2.9}$$

Für eine etwas andere Schreibweise zerlege A in

$$A = D + L + U, \tag{10.2.10}$$

wobei L eine untere und U eine obere Dreiecksmatrix bezeichnen. Damit läßt sich (10.2.9) als

$$x^{k+1} = -D^{-1}(L + U)x^k + D^{-1}b \tag{10.2.11}$$

bzw. in Komponenten

$$x_i^{k+1} = a_{ii}^{-1} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^k \right) \tag{10.2.11a}$$

für $i = 1, \ldots, n$ schreiben.

Alle Komponenten von x^{k+1} lassen sich unabhängig voneinander parallel aus x^k ermitteln. Daher heißt dieses Verfahren Gesamtschrittverfahren. Ersetzt man L, D und U durch Blockmatrizen, so ergibt sich sofort die Blockversion des JACOBI–Verfahrens.

Die *Konvergenz des Gesamtschrittverfahrens* benötigt Eigenschaften von A.

Satz 10.2.12 Ist A strikt diagonaldominant (starkes Zeilensummenkriterium), d.h.

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \tag{10.2.13}$$

für $i = 1, \ldots, n$, so konvergiert (10.2.11) für jeden Startwert x^0. Eine analoge Aussage gilt für das starke Spaltensummenkriterium.

Beweis: Aus (10.2.11) folgt

$$\begin{aligned} \|I - CA\|_\infty &= \|I - D^{-1}A\|_\infty \\ &= \|D^{-1}(L + U)\|_\infty \\ &= \max_i \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| < 1. \end{aligned}$$

Nun folgt die Behauptung mit Korollar 10.2.7 und $\|\cdot\| = \|\cdot\|_\infty$. ∎

Die strikte Diagonaldominanz in Satz 10.2.12 ist eine sehr starke Forderung, die selbst bei der der Diskretisierung einfacher Differentialgleichungen wie der POISSON–Gleichung

$$-\Delta u = f$$

nicht erfüllt wird (vergl. die Diskretisierungsmatrix in Anhang A). In diesem Fall kann man den Satz aber auf „$\geq$“ in (10.2.13) abschwächen.

Einzelschrittverfahren (Gauss–Seidel–Verfahren)

Alternativ kann man die Matrix C aber auch in Dreiecksgestalt wählen, womit das System ebenfalls einfach aufzulösen bleibt. Wähle dazu

$$C = (D + L)^{-1}.$$

Dann gilt

$$\begin{aligned} x^{k+1} &= (I - CA)x^k + Cb \\ &= (I - (D + L)^{-1}A)x^k + (D + L)^{-1}b \\ &= (I - (D + L)^{-1}(D + L + U))x^k + (D + L)^{-1}b \\ &= -(D + L)^{-1}Ux^k + (D + L)^{-1}b, \end{aligned}$$

was äquivalent zu

$$(D + L)x^{k+1} = -Ux^k + b \tag{10.2.14}$$

oder

$$Dx^{k+1} = -Lx^{k+1} - Ux^k + b \tag{10.2.14a}$$

ist. Prozedural angewandt in Komponentenschreibweise bedeutet dies

$$\sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + a_{ii}x_i^{k+1} = -\sum_{j=i+1}^{n} a_{ij}x_j^k + b_i$$

oder äquivalent dazu

$$x_i^{k+1} = (a_{ii})^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\right) \tag{10.2.14b}$$

für $i = 1, \ldots, n$. Im Unterschied zum Gesamtschrittverfahren werden hier also schon berechnete Komponenten von x^{k+1} bei der Berechnung der restlichen Komponenten miteinbezogen. Dieses Verfahren heißt *Einzelschritt–* oder Gauss–Seidel–Verfahren.

Für die Konvergenz dieses Verfahrens gilt

Satz 10.2.15 Das Einzelschrittverfahren (10.2.14) konvergiert für jeden Startwert x^0, falls die Matrix A symmetrisch positiv definit ist.

Beweis: Einen Beweis findet man in [DH], p. 266. ∎

Hierzu sei noch bemerkt, daß das Einzelschrittverfahren ebenfalls für jeden Startwert x^0 konvergiert, falls A strikt diagonaldominant ist. Dieses Verfahren ist ein Spezialfall einer allgemeineren Methode, die im folgenden hergeleitet wird.

Relaxationsverfahren

Die Konvergenz des Gesamtschrittverfahrens (10.2.11) und des Einzelschrittverfahrens (10.2.14) hängt vom Spektralradius $\rho(I - CA)$ ab. D.h. je kleiner $\rho(I - CA)$ ist, desto schneller konvergiert das jeweilige Verfahren. Für jedes Verfahren gibt es aber eine einfache Beschleunigung, nämlich

die *Relaxation*, indem man mit einem Parameter ω eine Konvexkombination aus *alten* und *neuen* Iterierten bildet. Anstelle von

$$x^{k+1} = (I - CA)x^k + Cb =: Mx^k + Cb \tag{10.2.3}$$

bilde

$$x^{k+1} := \omega(Mx^k + Cb) + (1 - \omega)x^k.$$

Hierbei entspricht die erste Klammer obiger Gleichung den neuen Iterierten und die zweite den alten. Verschiedene Wahlen von ω haben verschiedene Bezeichnungen: Im Fall $\omega > 1$ spricht man von *Überrelaxation* (overrelaxation) und im Fall $\omega < 1$ von *Unterrelaxation*. Für $\omega > 1$ heißt das Verfahren allgemein *SOR–Verfahren* (successive overrelaxation). ω heißt hierbei *Relaxations–* oder *Dämpfungsparameter*. Eine typische Wahl ist $C = (D + L)^{-1}$ (wie beim Einzelschritt–/Gauss–Seidel–Verfahren) und eine Konvexkombination aus Iterierten in der Form von (10.2.14a), so daß die Iterationvorschrift des SOR–Verfahrens

$$Dx^{k+1} = \omega(-Lx^{k+1} - Ux^k + b) + (1 - \omega)Dx^k \tag{10.2.16}$$

bzw. in Komponentenschreibweise

$$a_{ii}x_i^{k+1} = \omega\left(-\sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k + b_i\right) + (1 - \omega)a_{ii}x_i^k \tag{10.2.16a}$$

lautet.

Satz (von Ostrowski–Reich) Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit. Dann konvergiert das SOR–Verfahren (10.2.16) für $0 < \omega < 2$. Im Fall $\omega = 1$ entspricht das SOR–Verfahren dem Gauss–Seidel–Verfahren.

Beweis: Einen Beweis findet man in [SB2], p. 581. ∎

SSOR–Verfahren

Eine Variante des SOR–Verfahrens ist das symmetrische SOR–Verfahren, das SSOR–Verfahren. In (10.2.16a) werden die Komponenten von x^{k+1} „von oben nach unten“ durchlaufen. Dies kann man auch umkehren: Ein durchlaufen der Iteration jeweils in Vorwärts– und Rückwärtsrichtung ergibt

$$\begin{aligned} Dx^{k+1/2} &= \omega(-Lx^{k+1/2} - Ux^k + b) + (1 - \omega)Dx^k \\ Dx^{k+1} &= \omega(-Lx^{k+1/2} - Ux^{k+1} + b) + (1 - \omega)Dx^{k+1/2}. \end{aligned} \tag{10.2.17}$$

Eine Konvergenzaussage ist mit dem Satz von Ostrowski–Reich ableitbar. Der Vorteil dieses Verfahrens ist, daß der Aufwand von k SSOR–Zyklen nicht $2k$ SOR, sondern nur $k + \frac{1}{2}$ Zyklen SOR entspricht. Weiter führt das Verfahren auf eine symmetrische Iterationsmatrix. Bemerkungen zur Güte der einzelnen Verfahren werden später gegeben.

10.3 Gradientenverfahren

Im folgenden sei die Matrix $A \in \mathbb{R}^{n\times n}$ stets symmetrisch positiv definit. Solche Matrizen entstehen typischerweise bei der Diskretisierung elliptischer partieller Differentialgleichungen. Eine besondere Rolle spielt in diesem Fall das (p)cg–Verfahren (preconditioned conjugate gradient). Vorher wird allerdings das allgemeine Gradientenverfahren behandelt.

Lemma 10.3.1 Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit. Dann gilt: $x^* \in \mathbb{R}^n$ löst die Gleichung $Ax = b$ genau dann, wenn x^* Minimum von

$$f(x) = \frac{1}{2}x^T Ax - x^T b \tag{10.3.2}$$

ist.

Beweis: Nach Voraussetzung ist $f \in \mathcal{C}^2(\mathbb{R}^n)$. Die notwendige Bedingung für ein lokales Minimum lautet

$$\nabla f(x) = 0 \Longleftrightarrow Ax - b = 0.$$

Also ist $Ax - b$ ein kritischer Punkt. Da $D^2 f(x) = A$ nach Voraussetzung positiv definit ist, ist dieses lokale Minimum sogar global, und die notwendige Bedingung ist auch hinreichend.

∎

Das *Gradientenverfahren* besteht nun darin, das Minimum

$$\min_{x\in\mathbb{R}^n} f(x) \tag{10.3.3}$$

einer Funktion $f \in \mathcal{C}^1(\Omega)$ mit $\Omega \subset \mathbb{R}^n$ offen zu finden. Algorithmisch entspricht dies

Algorithmus 10.3.4 (Allgemeines Gradientenverfahren mit vollständiger linearer Suche) Sei $\Omega \subset \mathbb{R}^n$ offen und $f \in \mathcal{C}^1(\Omega)$. Wähle $x_0 \in \Omega$. Für $k = 0, 1, 2, \ldots$

1.) Bestimmung der Richtung: berechne die „Abstiegsrichtung“

$$d_k = -\nabla f(x_k).$$

2.) Liniensuche: suche das Minimum von f auf der Linie

$$\{x_k + \alpha d_k,\ \alpha \geq 0\} \cap \Omega.$$

Das (lokale) Minimum werde bei $\alpha = \alpha_k$ angenommen. Setze dann

$$x_{k+1} = x_k + \alpha_k d_k.$$

∎

Zur Abstiegsrichtung $-\nabla f(x_k)$ und praktischen Durchführung dieses Verfahrens kommen wir hinten. Zunächst betrachten wir die quadratische Funktion f in (10.3.2). Hier gilt speziell

$$d_k = b - Ax_k \tag{10.3.5}$$

$$\alpha_k = \frac{d_k^T d_k}{d_k^T A d_k}, \tag{10.3.6}$$

denn setze

$$\begin{aligned}\tilde{f}(\alpha) &:= f(x_k + \alpha d_k)\\ &= \frac{1}{2}(x_k + \alpha d_k)^T A(x_k + \alpha d_k) - (x_k + \alpha d_k)^T b\\ &= \frac{1}{2}(x_k^T A x_k + 2\alpha d_k^T A x_k + \alpha^2 d_k^T A d_k) - x_k^T b - \alpha d_k^T b.\end{aligned}$$

Dann gilt

$$\begin{aligned}\tilde{f}'(\alpha) &= d_k^T A x_k + \alpha d_k^T A d_k - d_k^T b\\ &= d_k^T (A x_k - b) + \alpha d_k^T A d_k\\ &= -d_k^T d_k + \alpha d_k^T A d_k\\ &\stackrel{!}{=} 0,\end{aligned}$$

womit

$$\alpha_k = \frac{d_k^T d_k}{d_k^T A d_k}$$

folgt.

Für die Abstiegsrichtung ist ausgehend von $f(x_k)$ ein d zu bestimmen, so daß

$$f(x_k + \beta d) \leq f(x_k)$$

ist. D.h., finde ein d mit $\|d\| = 1$, so daß die Richtungsableitung $D_d f(x_k) = (\nabla f(x_k))^T d$ minimal ist. Die Wahl

$$d = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$$

liefert die Richtung des steilsten Abstiegs. Da $D_d f = \langle \nabla f, d\rangle = \cos\theta \|\nabla f\| \|d\|$, ist im Fall $\nabla f \neq 0$ die Ableitung $D_d f$ minimal, falls $\theta = 0$, die Richtungsableitung also maximal, wenn ∇f und d dieselbe Richtung haben.

Zur praktischen Durchführung des Gradientenverfahrens bemerken wir noch, daß der zweite Schritt nur näherungsweise ausgeführt wird. Man bestimmt z.B. einen Punkt $x_k + t d_k$, in dem die Richtungsableitung klein, etwa

$$|d_k^T \nabla f(x_k + t d_k)| < \frac{1}{4}|d_k^T \nabla f(x_k)|$$

ist. Hierfür kann man folgende *globale Konvergenzaussage* zeigen: es konvergiert wenigstens eine Teilfolge gegen $x^* \in \Omega$ mit $\nabla f(x^*) = 0$, oder die Folge konvergiert gegen den Rand von Ω. (x^* kann allerdings auch nur lokales Minimum oder Sattelpunkt sein; daher sei im folgenden f stets quadratisch angenommen.)

Bemerkung 10.3.7 Die Funktion $f(x) = \frac{1}{2}x^T A x - x^T b$ ist quadratisch auf $\mathbb{R}^n$, falls A symmetrisch positiv definit ist. In diesem Fall konvergiert das Gradientenverfahren für jeden Startwert gegen das Minimum von f.

Konvergenzgeschwindigkeit

Als Maß für den Abstand betrachte die *Energienorm*

$$\|x\|_A := \sqrt{x^T A x}, \tag{10.3.8}$$

wohldefiniert, da jede symmetrisch positiv definite Matrix A ein Skalarprodukt $(x, y)_A := x^T A y$ definiert. Für das Gradientenverfahren in (10.3.4) gilt

Satz 10.3.9 Sei A symmetrisch positiv definit und $\kappa = \kappa_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$ ihre spektrale Kondition. Dann liefert das Gradientenverfahren (10.3.4) für $f(x) = \frac{1}{2}x^T Ax - x^T b$ eine Folge $\{x_k\}_{k=0,1,2,\ldots}$ (x_0 als Startwert) mit

$$\|x_k - x^*\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x_0 - x^*\|_A, \tag{10.3.10}$$

wobei x^* die Lösung von $Ax = b$ bezeichne.

Beweis: Einen Beweis findet man etwa in [Br]. ∎

Ist κ sehr groß, dann ist wegen

$$\frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1} \approx 1 - \frac{2}{\kappa}$$

die Konvergenzrate nahe bei 1 und die Konvergenz damit sehr langsam.

Geometrische Erklärung

Seien $a \gg 1$ und $A = \begin{pmatrix} 1 & 0 \\ 0 & a \end{pmatrix}$ mit $\kappa(A) = a$. D.h. mit $x = (y, z)^T$ gilt

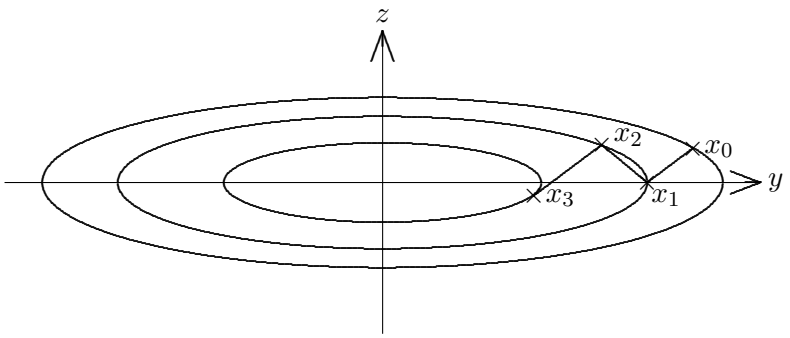
$$f(y, z) = \frac{1}{2}(y^2 + az^2).$$

Wähle als Startvektor $x_0 = (y_0, z_0)^T = (a, 1)^T$. Für die Abstiegsrichtung gilt dann

$$d_0 = -\nabla f(x_0) = -\left.\begin{pmatrix} y \\ az \end{pmatrix}\right|_{(y,z)=x_0} = a\begin{pmatrix} -1 \\ -1 \end{pmatrix}.$$

Weiter bekommt man $x_k = \begin{pmatrix} y_k \\ z_k \end{pmatrix}$ mit $y_{k+1} = \rho y_k$ und $z_{k+1} = -\rho z_k$, wobei $\rho = \frac{a-1}{a+1}$ ist. Also ist $d_k = -\nabla f(x_k)$ zwar Richtung des steilsten Abstiegs, aber schon für sehr kleine α kann dies für $f(x_k + \alpha d_k)$ ganz nahe liegen, so daß man α_k sehr klein wählen muß.

Die Höhenlinien von f sind in die Ebene projiziert stark gestreckte Ellipsen:



Das Verfahren ergibt, wie die Skizze zeigt, eine Art Zick–Zack–Kurs, in dem d_k und d_{k+1} zwar orthogonal bzgl. des EUKLIDISCHEN Skalarprodukts, aber fast parallel bzgl. $(\cdot, \cdot)_A$ sind. Dies liegt an den Eigenschaften von A, was zeigt, daß man diese mit in die Richtungssuche einbeziehen sollte. Dies wird im nächsten Abschnitt über das cg–Verfahren der Fall sein.

10.4 cg–Verfahren

Das Verfahren der konjugierten Gradienten/cg–Verfahren (conjugate gradient) wurde 1952 von Hestenes und Stiefel entwickelt und hat eine wachsende Bedeutung seit den 1970'er Jahren, bedingt durch effiziente Vorkonditionierer. Der Grundgedanke bei diesem Verfahren ist der, im Gradientenverfahren aus dem vorigen Abschnitt aufeinanderfolgende Richtungen so zu wählen, daß sie orthogonal bzgl. $(\cdot,\cdot)_A$ sind. Unsere Herleitung ist an [Br] angelehnt.

Im folgenden sei die Matrix A stets symmetrisch positiv definit. Zwei Vektoren $x, y \in \mathbb{R}^n$ heißen *A–orthogonal* oder *A–konjugiert*, falls $(x,y)_A = \langle x, Ay\rangle = x^T Ay = 0$ ist. Sind x, y A–konjugiert und gilt $x \not\equiv 0, y \not\equiv 0$, so sind x, y linear unabhängig, da A als positiv definit vorausgesetzt ist.

Seien nun $d_0, \ldots, d_{n-1}$ konjugierte Richtungen, d.h. $d_i \in \mathbb{R}^n$ und $(d_i, d_j)_A = 0$ für $i \neq j$. Dann besitzt $x^* = A^{-1}b$ die Darstellung

$$x^* = \sum_{k=0}^{n-1} \gamma_k d_k. \tag{10.4.1}$$

Für die Berechnung der γ_k gilt

$$\begin{aligned}(d_i, x^*)_A &= \left(d_i, \sum_{k=0}^{n-1} \gamma_k d_k\right)_A \\ &= \sum_{k=0}^{n-1} \gamma_k (d_i, d_k)_A \\ &= \gamma_i (d_i, d_i)_A,\end{aligned}$$

womit

$$\gamma_i = \frac{(d_i, x^*)_A}{(d_i, d_i)_A} = \frac{\langle d_i, b\rangle}{(d_i, d_i)_A} = \frac{d_i^T b}{d_i^T A d_i} \tag{10.4.2}$$

folgt. D.h. die Entwicklungskoeffizienten in (10.4.1) lassen sich direkt aus b, d_i berechnen.

Lemma 10.4.3 Seien $d_0, \ldots, d_{n-1} \in \mathbb{R}^n$ konjugierte Richtungen. Sei $x_0 \in \mathbb{R}^n$ ein beliebiger Startvektor und definiere für $k \geq 0$

$$x_{k+1} = x_k + \alpha_k d_k \tag{10.4.4}$$

mit

$$\alpha_k = -\frac{r_k^T d_k}{d_k^T A d_k} \tag{10.4.5}$$

und $r_k := Ax_k - b = \nabla f(x_k)$ als *Residuum*. Dann gilt: die Folge $\{x_k\}_{k\in\mathbb{N}_0}$ liefert nach (höchstens) n Schritten die Lösung $x_n = A^{-1}b$.

Beachte, daß das Verfahren in (10.4.4) noch nicht praktisch verwendbar ist, da die konjugierten Richtungen $d_0, \ldots, d_{n-1}$ im Gegensatz zum eigentlichen Gradientenverfahren noch nicht bekannt sind. Theoretisch könnte man versuchen, die d_i *vor* dem Start des Algorithmus zu berechnen, z.B. mit dem Gram–Schmidt–Verfahren. Dies ist aber viel zu aufwendig. Später werden wir sehen, daß $m \ll n$ Schritte für die Iteration ausreichen und die d_i direkt mitberechnet werden.

Beweis: Eine Rekursion von (10.4.4) ergibt

$$x_k = x_{k-1} + \alpha_{k-1} d_{k-1} = x_{k-2} + \alpha_{k-2} d_{k-2} + \alpha_{k-1} d_{k-1} = \ldots = x_0 + \sum_{j=0}^{k-1} \alpha_j d_j. \tag{10.4.6}$$

Sei nun x^* Lösung von $Ax = b$. Wir zeigen, daß $x^* = x_n$. Seien dazu $d_0, \ldots, d_{n-1}$ konjugiert (also linear unabhängig). Es gilt

$$x^* - x_0 = \sum_{j=0}^{n-1} \beta_j d_j$$

mit (irgendwelchen) $\beta_j \in \mathbb{R}$, für die

$$\begin{aligned} \frac{d_i^T A(x^* - x_0)}{d_i^T A d_i} &= \sum_{j=0}^{n-1} \beta_j \frac{d_i^T A d_j}{d_i^T A d_i} \\ &= \beta_i \frac{d_i^T A d_i}{d_i^T A d_i} \\ &= \beta_i \end{aligned}$$

gilt und weiterhin auch

$$\beta_i = \frac{d_i^T (Ax^* - Ax_0)}{d_i^T A d_i} = \frac{d_i^T (b - Ax_0)}{d_i^T A d_i} = \frac{-d_i^T r_0}{d_i^T A d_i}.$$

Als nächstes zeigen wir, daß $\beta_i = \alpha_i = \frac{-r_i^T d_i}{d_i A d_i}$ gilt. Dies ist aber der Fall wegen

$$\begin{aligned} \beta_i &= -\frac{d_i^T (Ax_0 - b)}{d_i^T A d_i} \\ &= -\frac{d_i^T (Ax_0 - b + A\sum_{j=0}^{i-1} \alpha_j d_j)}{d_i^T A d_i} \\ &= -\frac{d_i^T (A(x_0 + \sum_{j=0}^{i-1} \alpha_j d_j) - b)}{d_i^T A d_i} \\ &= -\frac{d_i^T (Ax_i - b)}{d_i^T A d_i} \\ &= -\frac{d_i^T r_i}{d_i^T A d_i} \\ &= \alpha_i. \end{aligned}$$

D.h. also

$$x^* - x_0 = \sum_{j=0}^{n-1} \alpha_j d_j \Longleftrightarrow x^* = x_0 + \sum_{j=0}^{n-1} \alpha_j d_j = x_n \tag{10.4.5a}$$

nach Definition der Iteration in (10.4.4). ∎

Korollar 10.4.7 Seien $d_0, \ldots, d_{n-1} \in \mathbb{R}^n$ konjugierte Richtungen. Für jedes in (10.4.4) mit (10.4.5) definierte x_k mit Startvektor x_0 gilt

$$f(x_k) = \min_{x \in V_k} f(x_0 + x), \tag{10.4.8}$$

wobei

$$V_k = \operatorname{span}\{d_0, \ldots, d_{k-1}\}. \tag{10.4.9}$$

Insbesondere gilt

$$d_i^T r_k = 0 \tag{10.4.10}$$

für $i < k$, d.h. das k–te Residuum ist orthogonal zu V_i. ∎

Im Unterschied zum Gradientenverfahren aus (10.3.4), bei dem man zunächst die Richtung $\tilde{d}_k = -\nabla f(x_k)$ und dann das Minimum von f auf der Linie $\{x_k + \tilde{\alpha}\tilde{d}_k,\ \tilde{\alpha} \geq 0\}$, welches bei $\tilde{\alpha} = \tilde{\alpha}_k$ angenommen wurde, bestimmte, wird hier durch die konjugierten Richtungen $d_0, \ldots, d_{k-1}$ das Minimum von f bzgl. des *affinen* Teilraums

$$\left\{ x \in \mathbb{R}^n :\ x = x_0 + \sum_{j=0}^{k-1} \gamma_j d_j,\ \gamma_j \in \mathbb{R} \right\}$$

angenommen. Beim Gradientenverfahren ist der affine Teilraum jeweils nur die Gerade $\{x :\ x_k + \tilde{\alpha}\tilde{d}_k\}$.

Beweis von Korollar 10.4.7: Es ist zu zeigen, daß x_k die Funktion f über dem affinen Teilraum V_k minimiert, also $f(x_k) = \min\limits_{x \in V_k} f(x_0 + x)$, und Gleichung (10.4.10) $d_i r_k = 0$ für $i < k$ stimmt. Zur ersten Aussage betrachte

$$\begin{aligned} \tilde{f}(\alpha_0, \ldots, \alpha_{k-1}) := f(x_k) &= f\left(x_0 + \sum_{j=0}^{k-1} \alpha_j d_j \right) \\ &= \frac{1}{2} x_k^T A x_k - b^T x_k, \end{aligned}$$

was

$$\nabla \tilde{f}(\alpha_0, \ldots, \alpha_{k-1}) = \begin{pmatrix} d_0^T r_0 + \alpha_0 d_0^T A d_0 \\ \vdots \\ d_{k-1}^T r_{k-1} + \alpha_{k-1} d_{k-1}^T A d_{k-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

nach sich zieht. (Dies gilt auch beim Gradientenverfahren, allerdings müssen dort die $\tilde{d}_i$ nicht linear unabhängig sein.) Da die d_i nach Voraussetzung für alle $i = 0, \ldots, k-1$ konjugiert sind, minimiert x_k die Funktion f über $x_0 + V_k = \{x_0 + x,\ x \in V_k\}$. Zum Beweis von (10.4.10) gilt

$$\begin{aligned} d_i^T r_{i+1} &= d_i^T (A x_{i+1} - b) \\ &= d_i^T (A(x_i + \alpha_i d_i) - b) \\ &= d_i^T (A x_i - b) + \alpha_i d_i^T A d_i \\ &= d_i^T r_i + \alpha_i d_i^T A d_i \\ &= 0 \end{aligned}$$

nach Wahl von α_i. Also gilt (10.4.10) für $i = k-1$ und für $k = 1$, da $d_0^T r_1 = 0$. Den Rest zeigen wir per vollständiger Induktion nach k. Sei also $d_i^T r_k = 0$ richtig für $i < k$. Die Gültigkeit für $k+1$ ist zu zeigen. Nach Definition gilt

$$x_{k+1} - x_k = \alpha_k d_k,$$

was

$$\begin{aligned} r_{k+1} - r_k &= A x_{k+1} - b - (A x_k - b) \\ &= A(x_{k+1} - x_k) \\ &= \alpha_k A d_k \end{aligned}$$

impliziert. Damit gilt aber

$$d_i^T (r_{k+1} - r_k) = \alpha_k d_i^T A d_k = 0$$

für $k > i$; insbesondere ist (10.4.10) also auch für $k+1$ richtig. ∎

Aus Lemma 10.4.3 läßt sich ein Algorithmus entwickeln, sobald man die konjugierten Richtungen d_i berechnen kann. Tatsächlich berechnet man die d_i aus $\nabla f(x_i)$ durch Addition einer Korrektur. Dies ist die entscheidende Verwandtschaft zwischen cg– und Gradientenverfahren.

In Lemma 10.3.1 haben wir gezeigt, daß x^* genau dann die Gleichung $Ax = b$ löst, wenn es die Funktion $f(x) = \frac{1}{2}x^T Ax - x^T b$ minimiert. Für das Gradientenverfahren aus (10.3.4) erhält man mit beliebigem Startwert $x_0 \in \mathbb{R}^n$ als erste Richtung

$$d_0 := -\nabla f(x_0) = b - Ax_0 =: r_0$$

als Residuum des Startwerts. Nehme nun für $k = 0, 1, 2, \ldots$ als d_{k+1} denjenigen Anteil des Residuums $-\nabla f(x_{k+1}) = b - Ax_{k+1}$, der zu $d_0, \ldots, d_k$ konjugiert ist. Setze also an

$$r_{k+1} = -d_{k+1} + \sum_{j=0}^{k} \beta_j d_j \Longleftrightarrow d_{k+1} = -r_{k+1} + \sum_{j=0}^{k} \beta_j d_j.$$

Stelle dabei sicher, daß $r_{k+1} \notin \operatorname{span}\{d_0, \ldots, d_k\}$ ist, denn sonst ist $d_{k+1} = 0$ möglich. Es läßt sich sogar erreichen, daß $\beta_j = 0$ für alle $j = 0, \ldots, k-1$ ist:

Lemma 10.4.11 Sei $d_0 = b - Ax_0 = -r_0$. Für $k = 0, 1, 2, \ldots$ setze

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{10.4.12}$$

solange $r_k \neq 0$ und

$$d_{k+1} = -r_{k+1} + \beta_k d_k. \tag{10.4.13}$$

Dann gilt: die d_j sind konjugiert, d.h. $d_i^T A d_j = 0$ für $i \neq j$, und

$$d_i^T r_j = -r_i^T r_j = 0 \tag{10.4.14}$$

für $i < j$.

Beachte: Sobald $r_k = 0 = Ax_k - b$ gilt, ist man fertig. $d_i^T r_j = 0$ galt für $i < j$ schon in (10.4.10).

Beweis: Wir zeigen mit vollständiger Induktion über k, daß die d_{k+1} gemäß (10.4.13) konjugiert sind. Für den Induktionsanfang sei $k = 0$. Wegen (10.4.10) und der Wahl von d_0 gilt

$$0 = d_0^T r_1 = -r_0^T r_1.$$

Also sind d_0 und r_1 linear unabhängig und man kann ansetzen

$$d_1 = -r_1 + \gamma d_0.$$

Nun gilt mit α_0 gemäß (10.4.5)

$$\begin{aligned}
\alpha_0 d_0^T A d_1 &= \alpha_0 d_0^T A(-r_1 + \gamma d_0) \\
&= -\alpha_0 d_0^T A r_1 + \gamma \alpha_0 d_0^T A d_0 \\
&= -\alpha_0 d_0^T A r_1 - \gamma d_0^T r_0 \\
&= -\alpha_0 d_0^T A r_1 - r_0^T r_1 + \gamma r_0^T r_0 \\
&= -(r_0 + \alpha_0 A d_0)^T r_1 + \gamma r_0^T r_0 \\
&= -r_1^T r_1 + \gamma r_0^T r_0,
\end{aligned}$$

d.h. d_0 und d_1 sind genau dann konjugiert, also $d_0^T A d_1 = 0$, wenn

$$\gamma = \frac{r_1^T r_1}{r_0^T r_0} = \beta_1$$

gilt, was den Induktionsanfang beendet. Induktionsannahme: Seien $d_0, \ldots, d_k$ konjugiert. Wir haben zu zeigen, daß dann auch $d_0, \ldots, d_{k+1}$ mit d_{k+1} gemäß (10.4.12) und (10.4.13) konjugiert sind. Mit α_k aus (10.4.5) gilt wieder

$$\begin{aligned} \alpha_k d_k^T A d_{k+1} &= \alpha_k d_k^T A(-r_{k+1} + \beta_k d_k) \\ &= -\alpha_k d_k^T A r_{k+1} + \beta_k \alpha_k d_k^T A d_k \\ &= -(r_k + \alpha_k A d_k)^T r_{k+1} - \beta_k r_k^T d_k \\ &= -r_{k+1}^T r_{k+1} + \beta_k r_k^T r_k. \end{aligned}$$

Also ist d_{k+1} genau dann zu d_k konjugiert, wenn

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

gilt. Es bleibt zu zeigen, daß $r_k^T r_{k+1} = 0 (= -r_k^T d_k)$ gilt. Dies aber folgt aus

$$\begin{aligned} r_{k+1}^T r_k &= r_{k+1}^T(\beta_{k-1} d_{k-1} - d_k) \\ &= \beta_{k-1} r_{k+1}^T d_{k-1} - r_{k+1}^T d_k \\ &= 0. \end{aligned}$$

∎

Nun können wir den Algorithmus des cg–Verfahrens angeben.

Algorithmus (cg–Verfahren): Gegeben seien eine Matrix $A \in \mathbb{R}^{n \times n}$ symmetrisch positiv definit und $b \in \mathbb{R}^n$. Gesucht sei die Lösung x^* der Gleichung $Ax = b$, die gleichzeitig die Funktion $f(x) = \frac{1}{2} x^T A x - x^T b$ minimiert.

- Wähle $x_0 \in \mathbb{R}^n$ und setze

$$d_0 := -\nabla f(x_0) = b - A x_0 =: -r_0.$$

- Für $k = 0, 1, 2, \ldots$ berechne dann

$$\begin{aligned} \alpha_k &= \frac{r_k^T r_k}{d_k^T A d_k} \\ x_{k+1} &= x_k + \alpha_k d_k \\ r_{k+1} &= r_k + \alpha_k A d_k. \end{aligned}$$

  Falls x_{k+1} genügend genau (oder $r_{k+1} = 0$) breche ab. Sonst setze

$$\begin{aligned} \beta_k &= \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \\ d_{k+1} &= -r_{k+1} + \beta_k d_k. \end{aligned}$$

■

Beachte die besondere Form von α_k in diesem Algorithmus:

$$\alpha_k = \frac{-r_k^T d_k}{(d_k,d_k)_A} = \frac{-r_k^T(-r_k+\beta_{k-1}d_{k-1})}{(d_k,d_k)_A} = \frac{r_k^T r_k}{(d_k,d_k)_A} - \beta_{k-1}(r_k^T d_{k-1}) = \frac{r_k^T r_k}{(d_k,d_k)_A}.$$

Eine etwas andere Sichtweise des cg–Verfahrens ist die folgende: Nach (10.4.5a) besitzt die exakte Lösung x^* von $Ax = b$ die Darstellung

$$\begin{aligned} x^* &= x_0 + \sum_{j=0}^{n-1} \alpha_j d_j \\ &= x_0 + \sum_{j=0}^{k-1} \alpha_j d_j + \sum_{j=k}^{n-1} \alpha_j d_j \\ &= x_k + \sum_{j=k}^{n-1} \alpha_j d_j, \end{aligned}$$

was äquivalent zu

$$x^* - x_k = \sum_{j=k}^{n-1} \alpha_j d_j$$

ist. Damit gilt weiter

$$\begin{aligned} (x^* - x_k, d_i)_A &= \left(\sum_{j=k}^{n-1} \alpha_j d_j, d_i \right)_A \\ &= \sum_{j=k}^{n-1} \alpha_j (d_j, d_i)_A \\ &= 0 \end{aligned}$$

für $i < k$, was $(x^* - x_k, v_k)_A = 0$ für alle $v_k \in V_k = \operatorname{span}\{d_0, \ldots, d_{k-1}\}$ impliziert. Dies ist nach Satz 4.3.3 äquivalent[3] zu

$$\|x^* - x_k\|_A = \min_{x \in V_k} \|(x^* - x_0) - x\|_A,$$

d.h. das cg–Verfahren erzeugt, da die d_i konjugiert sind, eine A–orthogonale Projektion auf den Teilraum V_k. Das Gradientenverfahren produzierte ebenfalls eine in $x_0 + V_k$ liegende Lösung, diese hatte allerdings nicht minimale Energienorm.

Konvergenzgeschwindigkeit

Satz 10.4.15 Für das cg–Verfahren gilt mit jedem beliebigen Startwert $x_0 \in \mathbb{R}^n$

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \right)^k \|x_0 - x^*\|_A, \tag{10.4.16}$$

wobei $\kappa = \kappa_2(A)$ die spektrale Kondition von A bezeichnet.

[3] In Kapitel 4 wurden bei der Behandlung linearer Ausgleichsprobleme orthogonale Projektionen auf einen Unterraum $U_n \subset V$ eines Vektorraums V betrachtet, und zu gegebenem $v^* \in V$ ein $u_n \in U_n$ gesucht, welches die Gleichung

$$\|v^* - u_n\| = \min_{u_n \in U_n} \|v^* - u\|$$

erfüllte.

Beweis: Einen Beweis findet man in [DH], p. 278f.. ∎

Gleichung (10.4.16) besagt, wie der Anfangsfehler im *ungünstigsten* Fall reduziert wird, d.h. wenn k so gewählt, ist daß die rechte Seite kleiner als die Toleranzgrenze gewählt ist, so gilt natürlich

$$\|x_k - x^*\|_A \leq \text{ tol.}$$

Vorkonditionierung

Anstelle das cg–Verfahren auf die Gleichung

$$Ax = b$$

anzuwenden, kann man auch das vorkonditionierte pcg–Verfahren (preconditioned conjugate gradient) auf die Gleichung

$$BAx = Bb$$

anwenden, wobei $B \approx A^{-1}$ entspricht. Dabei wird der Wert für die Kondition in (10.4.16) zu $\kappa = \kappa_2(BA)$. Typische Wahlen von B sind z.B.

$$B = (D - \omega L)^{-1},$$

wie beim SSOR–Verfahren, wobei $A = D + L + U$ gelte. Weiterhin kann man aber auch eine „unvollständige“ Cholesky–Zerlegung der Matrix A derart durchführen, daß man nur ein Band um die Diagonale nach der Cholesky–Methode zerlegt, so daß $A \approx LL^T$ gilt, und dann

$$B = (LL^T)^{-1}$$

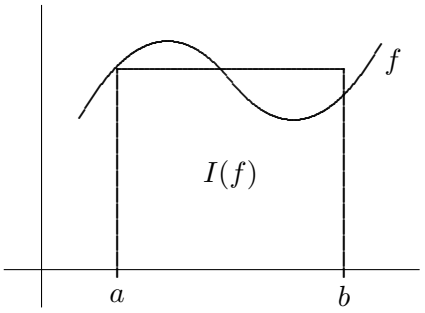
setzt. Dies Verfahren heißt ILU–Verfahren. Auf diese Art und Weise kann man die Anwendungsmöglichkeiten des Gradienten–/cg–Verfahrens auch auf Fälle erweitern, in denen die Matrix A nicht positiv definit ist.

11 Numerische Integration

Integrale sind analytisch oft nicht geschlossen berechenbar, was eine *Numerische Berechnung* eines RIEMANN–Integrals der Form

$$\int_a^b f(x)dx$$

notwendig macht. Unter einer solchen *Numerischen Integration* oder *Quadratur* versteht man die Aufgabe, den Flächeninhalt $I(f)$ unter dem Graphen von f durch die Berechnung eines Quadrats zu approximieren. Bei mehrdimensionalen Integranden spricht man demgemäß von *Kubatur*.



11.1 Einleitung und klassische Methoden

Im folgenden sei die Aufgabe gestellt, das Integral

$$I(f) := \int_a^b f(x)dx$$

mit einer meist stückweise glatten und stetigen Funktion $f \in \mathcal{C}([a,b])$ zu berechnen. Die grundsätzliche Idee hierbei ist, die Funktion f durch eine Funktion $\tilde{f}$ zu ersetzen, so daß $I(\tilde{f})$ berechenbar wird. Zur Abschätzung der *Kondition* dieser Quadraturaufgabe seien $I := \int_a^b f(x)dx$ und $\tilde{I} := \int_a^b \tilde{f}(x)dx$ mit einem gestörten Integranden $\tilde{f}$. Es gilt

$$\begin{aligned} |I - \tilde{I}| &= \left| \int_a^b (f(x) - \tilde{f}(x))dx \right| \\ &\leq \int_a^b |f(x) - \tilde{f}(x)|dx \\ &\leq (b-a) \sup_{x\in[a,b]} |f(x) - \tilde{f}(x)| \\ &= (b-a)\|f - \tilde{f}\|_{\infty,[a,b]}, \end{aligned}$$

d.h. die absolute Kondition des Integrationsproblems ist gut. Für den relativen Fehler gilt

$$\frac{|I-\tilde{I}|}{|I|} \leq (b-a)\frac{\|f-\tilde{f}\|_\infty \|f\|_\infty}{|\int_a^b f(x)dx| \|f\|_\infty} = \frac{\int_a^b \|f\|_\infty dx}{|\int_a^b f(x)dx|}\frac{\|f-\tilde{f}\|_\infty}{\|f\|_\infty} =: \kappa_{\text{rel}}\frac{\|f-\tilde{f}\|_\infty}{\|f\|_\infty}.$$

Falls f also stark oszilliert, kann

$$\left| \int_a^b f(x)dx \right| \ll (b-a)\|f\|_\infty$$

gelten, also $\kappa_{\text{rel}} \gg 1$ sein und damit die relative Kondition sehr schlecht sein.

Klassische Integrationsverfahren

Zur Berechnung von $I(f) = \int_a^b f(x)dx$ kann man so vorgehen:

1. Unterteile das Intervall $[a, b]$ für $k = 1, \ldots, n$ in n Teilintervalle $[t_{k-1}, t_k]$ mit den äquidistanten Stützstellen $t_k = a + kh$ und $h = \frac{b-a}{n}$.
2. Approximiere f auf jedem Teilintervall $[t_{k-1}, t_k]$ durch eine einfach zu integrierende Funktion g_k und benutze den Zusammenhang

$$\sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} g_k(x)dx \approx \sum_{k=1}^{n} \int_{t_{k-1}}^{t_k} f(x)dx = \int_a^b f(x)dx. \tag{11.1.1}$$

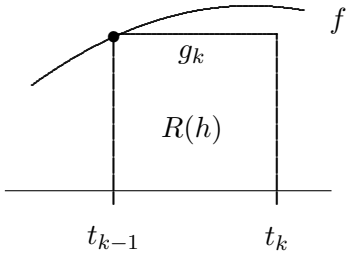
Es werden nun drei derartige Verfahren vorgestellt.

Beispiel 11.1.2 (Rechteckregel) Eine mögliche Wahl ist $g_k(x) := f(t_{k-1})$ für $x \in [t_{k-1}, t_k]$. Dann ist

$$R(h) := h \sum_{k=1}^{n} f(t_{k-1})$$

ein Näherungswert für das Integral $I(f)$. Mit TAYLOR–Entwicklung um t_{k-1}

$$f(x) = f(t_{k-1}) + (x - t_{k-1})f'(\xi),$$



($\xi \in [t_{k-1}, t_k]$) gilt für die Approximationsgüte

$$\begin{aligned} hf(t_{k-1}) &= \int_{t_{k-1}}^{t_k} f(t_{k-1})dx \\ &= \int_{t_{k-1}}^{t_k} f(x)dx + O(h^2), \end{aligned}$$

was

$$\begin{aligned} R(h) &= h \sum_{k=1}^{n} f(t_{k-1}) \\ &= \sum_{k=1}^{n} \left(\int_{t_{k-1}}^{t_k} f(x)dx + O(h^2) \right) \\ &= \int_a^b f(x)dx + nO(h^2) \\ &= \int_a^b f(x)dx + O(h) \\ &= I(f) + O(h) \end{aligned}$$

ergibt. D.h. der Fehler $I(f) - R(h)$ geht für $h \to 0$ von der Ordnung $O(h)$ gegen Null und die Rechteckregel ist ein *Verfahren erster Ordnung.* ∎

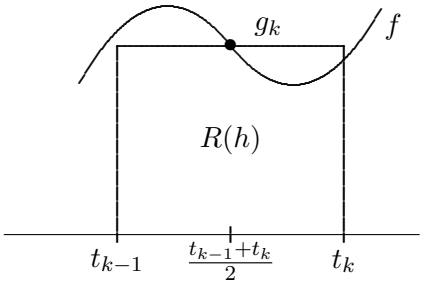
Beispiel 11.1.3 (Mittelpunktsregel) Wähle $g_k(x) := f\left(\frac{t_{k-1}+t_k}{2}\right)$ für $x \in [t_{k-1}, t_k]$, was eine Näherung

$$R(h) = h \sum_{k=1}^{n} f\left(\frac{t_{k-1} + t_k}{2}\right)$$

für das Integral $I(f)$ ergibt. Man kann wie oben mit TAYLOR–Entwicklung um t_{k-1} zeigen, daß für $f \in \mathcal{C}^2([t_{k-1}, t_k])$

$$hf\left(\frac{t_{k-1}+t_k}{2}\right) = \int_{t_{k-1}}^{t_k} f(x)dx - \frac{f''(\xi_k)}{24}h^3$$

für ein $\xi_k \in [t_{k-1}, t_k]$ gilt. Damit ist



$$\begin{aligned} R(h) &= h\sum_{k=1}^{n} f\left(\frac{t_{k-1}+t_k}{2}\right) \\ &= \sum_{k=1}^{n}\int_{t_{k-1}}^{t_k} f(x)dx - \sum_{k=1}^{n}\frac{f''(\xi_k)}{24}h^3 \\ &= \int_a^b f(x)dx - \sum_{k=1}^{n}\frac{f''(\xi_k)}{24}h^3 \\ &=: I(f) - E_h(f). \end{aligned}$$

Für den Diskretisierungsfehler (Verfahrensfehler) erhält man somit

$$|E_h(f)| \le \frac{h^3}{24}\sum_{k=1}^{n}\left|f''(\xi_k)\right| \le \frac{h^3}{24}n\|f''\|_\infty = \frac{h^2}{24}(b-a)\|f''\|_\infty.$$

Das Verfahren ist also *zweiter Ordnung*, da der Fehler für $h \to 0$ mit h^2 gegen Null geht. ∎

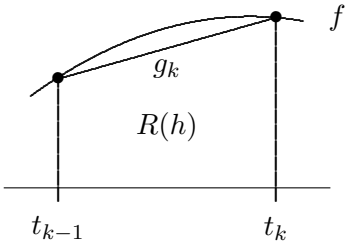
Sehr bekannt ist auch die folgende Methode:

Beispiel 11.1.4 (Trapezregel) Ersetze f auf $[t_{k-1}, t_k]$ durch die lineare Interpolation von $f(t_{k-1})$ und $f(t_k)$, d.h. f auf $[a, b]$ durch einen Polygonzug bzw. linearen Spline.
Wähle dazu in (11.1.1)

$$g_k(x) := \frac{x - t_{k-1}}{h}f(t_k) + \frac{t_k - x}{h}f(t_{k-1})$$

für $x \in [t_{k-1}, t_k]$. Dann gilt

$$\int_{t_{k-1}}^{t_k} g_k(x)dx = h\frac{1}{2}(f(t_{k-1}) + f(t_k)),$$



und die $I(f)$ approximierende Fläche ist ein Trapez. Für die *Trapezsumme* gilt

$$R(h) = h\left[\frac{1}{2}(f(a)+f(b)) + \sum_{k=1}^{n-1} f(t_{k-1})\right].$$

Man kann wie bei der Mittelpunktsregel zeigen, daß für $f \in \mathcal{C}^2([t_{k-1}, t_k])$ und $\xi_k \in [t_{k-1}, t_k]$

$$h\frac{1}{2}(f(t_{k-1}) + f(t_k)) = \int_{t_{k-1}}^{t_k} f(x)dx + \frac{f''(\xi_k)}{12}h^3$$

und damit für den Diskretisierungsfehler

$$|E_h(f)| = |I(f) - R(h)| \leq \frac{h^3}{12} \sum_{k=1}^{n} |f''(\xi_k)| \leq \frac{h^2}{12}(b-a)\|f''\|_\infty$$

gilt, das Verfahren also von *zweiter Ordnung* ist. ∎

11.2 Die NEWTON–COTES–Formeln

Im folgenden stehe $[c,d]$ statt $[t_{k-1}, t_k]$ für ein typisches Teilintervall. Obige Beispiele waren von folgendem Typ: Seien $x_0, \ldots, x_m \in [c,d]$ verschieden. Approximiere f durch das *Interpolationspolynom* $P(f|x_0, \ldots, x_m)(x) \in \mathcal{P}_m$ vom Grad m (vgl. Kapitel 8), d.h.

$$I(f) = \int_c^d f(x)dx \approx \int_c^d P(f|x_0, \ldots, x_m)(x)dx =: I_m(f). \tag{11.2.1}$$

Konkret waren dies $m = 0$ und $x_0 = c$ bei der Rechteckregel, $m = 0$ und $x_0 = \frac{c+d}{2}$ bei der Mittelpunktsregel sowie $m = 1$ und $x_0 = c, x_1 = d$ bei der Trapezregel. Da P ein Polynom ist, sind die Integrale einfach zu berechnen. Weiter gilt

Bemerkung 11.2.2 Sei $I_m(f)$ durch (11.2.1) definiert. Dann gilt für jedes Polynom $Q \in \mathcal{P}_m$ vom Grad m

$$I_m(Q) = I(Q) = \int_c^d Q(x)dx.$$

Man sagt, die Quadraturformel ist *exakt* vom Grad m.

Beweis: Wegen der Eindeutigkeit der Polynominterpolation (vgl. Satz 8.2.7) gilt $Q(x) = P(Q|x_0, \ldots, x_m)(x)$ für jedes $Q \in \mathcal{P}_m$. Weiter liefert die Integration gleicher Integranden das gleiche Integral. ∎

Zur *praktischen Auswertung* von $I_m(f)$ benötigt man eine Darstellung von $I_m(f)$. Die Darstellung des Interpolationspolynoms $P(f|x_0, \ldots, x_m)(x)$ mittels der LAGRANGE–Fundamentalpolynome aus (8.3.5) liefert das folgende

Lemma 11.2.3 Es gibt Gewichte $c_0, \ldots, c_m$, so daß $I_m(f)$ in (11.2.1) die Form

$$I_m(f) = h \sum_{j=0}^{m} c_j f(x_j)$$

mit $h = d - c$ erhält. Die c_j sind durch

$$c_j = \frac{1}{h} \int_c^d \prod_{\substack{k=0 \\ k \neq j}}^{m} \frac{x - x_k}{x_j - x_k} dx$$

bestimmt.

Beweis: Übung. ∎

Wählt man nun die Stützstellen x_j *äquidistant*, d.h.

$$x_j = c + \frac{j}{m}h$$

mit $h = d - c$, dann läßt sich $I_m(f)$ in der Form

$$I_m(f) = h \sum_{j=0}^{m} c_j f(c + \xi_j h) \tag{11.2.4}$$

schreiben, wobei die normierten Stützstellen $\xi_j = \frac{j}{m}$ und die Gewichte c_j unabhängig von $[c, d]$ sind. Dies sind die NEWTON–COTES–Formeln.

Die folgende Tabelle gibt für verschiedene Werte von m zugehörige Stützstellen, Gewichte, Fehlerabschätzungen und Konvergenzordnungen an:

m	Verfahren	Stützstellen ξ_j	Gewichte c_j	Diskretisierungsfehler $I_m(f) - \int_c^d f(x)dx$	Ordn. auf $[a, b]$
0	Rechtecksregel	0	1	$O(h^2)$	1
0	Mittelpunktsregel	$\frac{1}{2}$	1	$\frac{1}{24}h^3 f''(\xi)$	2
1	Trapezregel	$0, 1$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{12}h^3 f''(\xi)$	2
2	SIMPSON–Regel	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{2}{3}, \frac{1}{6}$	$\frac{1}{90}\left(\frac{1}{2}h\right)^5 f^{(4)}(\xi)$	4
3	$\frac{3}{8}$–Regel/pulcherrima	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$\frac{3}{80}\left(\frac{1}{3}h\right)^5 f^{(4)}(\xi)$	4
4	MILNE–Regel	$0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$\frac{8}{945}\left(\frac{1}{4}h\right)^7 f^{(6)}(\xi)$	6

Für $m > 6$ treten negative Gewichte auf, so daß die Formeln numerisch unbrauchbar werden. Um das Integral nun praktisch auf ganz $[a, b]$ zu berechnen, unterteilt man das Intervall wie oben für $k = 1, \ldots, n$ in Teilintervalle $[t_{k-1}, t_k]$ mit $t_k = a + kh$ und $h = \frac{b-a}{n}$ und wiederholt den obigen Prozeß auf jedem dieser Intervalle.

Beispiel 11.2.5 Eine Anwendung der SIMPSON–Regel auf jedem Teilintervall $[c, d] = [t_{k-1}, t_k]$ ergibt die summierte SIMPSON–Regel

$$R(h) = \int_a^b f(x)dx + E_h(f)$$

mit

$$R(h) = \frac{h}{6}\left[f(t_0) + 4f\left(\frac{t_0 + t_1}{2}\right) + 2f(t_1) + 4f\left(\frac{t_1 + t_2}{2}\right) + \ldots + f(t_n)\right]$$

und

$$|E_h(f)| = |R(h) - I(f)| \leq \frac{h^4}{2880}(b - a)\|f^{(4)}\|_\infty.$$

∎

Man beachte noch, daß die h–Potenz im Restglied für m und $m+1$ gleich ist, falls m gerade ist.

11.3 Die EULER–MACLAURIN'sche Summenformel

Zur Abschätzung des Diskretisierungsfehlers bei den NEWTON–COTES–Formeln wurden die über die TAYLOR–Entwicklung hergeleiteten Identitäten

$$hf\left(\frac{c+d}{2}\right) = \int_c^d f(x)dx - \frac{f''(\xi)}{24}h^3$$

mit $\xi \in [c,d]$ für die Mittelpunktsregel und

$$h\frac{1}{2}(f(c)+f(d)) = \int_c^d f(x)dx + \frac{f''(\xi)}{12}h^3$$

für die Trapezregel verwendet. Diese sind Spezialfälle der EULER–MACLAURIN'schen Summenformel. Diese wiederum ist zur Herleitung der *Extrapolation* im nächsten Kapitel an sich nützlich. Für die Herleitung der EULER–MACLAURIN'sche Summenformel werden die BERNOULLI–Zahlen benötigt, welche wiederum über die BERNOULLI–Polynome definiert werden. Sei zunächst $[c,d] = [0,1]$.

Bemerkung 11.3.1 Die durch die Folge von Polynomen $\{B_k(x)\}_{k\geq 0}$ definierten Polynome

$$\begin{aligned}
B_0(x) &:= 1 \\
B_1(x) &:= x - \frac{1}{2} \\
B_2(x) &:= x^2 - x + \frac{1}{6} \\
B_3(x) &:= x^3 - \frac{3}{2}x^2 + \frac{1}{2}x \\
B_4(x) &:= x^4 - 2x^3 + x^2 - \frac{1}{30} \\
&\vdots
\end{aligned}$$

heißen BERNOULLI–Polynome. Sie sind allgemein durch die Rekursion

$$B'_{k+1}(x) = (k+1)B_k(x)$$

für $k = 1, 2, \ldots$ definiert, die die B_{k+1} allerdings nur bis auf eine Konstante festlegt. Diese Konstante ist so gewählt, daß

$$B_{2k+1}(0) = B_{2k+1}(1) = 0$$

für alle $k > 0$ gilt (mehr in [S]). Man nennt $B_k := B_k(0)$ die BERNOULLI–Zahlen ($\neq 0$ für gerades $k > 1$):

$$B_0 = 1, \quad B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_6 = \frac{1}{42}, \quad B_8 = -\frac{1}{30}, \quad \ldots$$

■

Damit gilt für $g \in \mathcal{C}^{2s+2}([0,1])$ und $\xi \in (0,1)$ die EULER–MACLAURIN'sche Summenformel

$$\int_0^1 g(x)dx = \frac{g(0)}{2} + \frac{g(1)}{2} + \sum_{k=1}^{s} \frac{B_{2k}}{(2k)!}\left(g^{(2k-1)}(0) - g^{(2k-1)}(1)\right) - \frac{B_{2s+2}}{(2s+2)!}g^{(2s+2)}(\xi). \quad (11.3.2)$$

Beweisidee: (Einen vollständigen Beweis findet man etwa in [S].) Forme $\int_0^1 g(x)dx$ durch partielle Integration sukzessive um und verwende die Rekursionsformel für die BERNOULLI–Polynome, d.h.

$$\int_0^1 g(x)dx = \int_0^1 B_0(x)g(x)dx = B_1(x)g(x)\Big|_0^1 - \int_0^1 B_1(x)g'(x)dx.$$

Nun läßt sich das Integral auf der rechten Seite durch

$$\int_0^1 B_1(x)g'(x)dx = \frac{1}{2}B_2(x)g(x)\Big|_0^1 - \frac{1}{2}\int_0^1 B_2(x)g''(x)dx$$

ausdrücken, wobei wieder $B'_{k+1}(x) = (k+1)B_k(x)$ für $k = 1, 2, \ldots$ gilt, usw. ∎

Nun werden wir eine entsprechende Darstellung für $\int_0^n g(x)dx$ und danach eine für $\int_a^b f(x)dx$ herleiten. Dies wird dann einen Zugang zu Extrapolationsverfahren liefern. Durch wiederholte Anwendung von (11.3.2) auf Integrale in den Grenzen $i-1$ und i und Summation über $i = 1, \ldots, n$ erhält man mit $\xi_i \in (i-1, i)$ und $\xi \in (0, n)$

$$\begin{aligned}
\int_0^n g(x)dx &= \sum_{i=1}^n \int_{i-1}^i g(x)dx \\
&= \frac{g(0)}{2} + g(1) + \ldots + g(n-1) + \frac{g(n)}{2} \\
&\qquad + \sum_{k=1}^s \frac{B_{2k}}{(2k)!}\left(g^{(2k-1)}(0) - g^{(2k-1)}(n)\right) - \frac{B_{2s+2}}{(2s+2)!}\sum_{i=1}^n g^{(2s+2)}(\xi_i) \\
&= \frac{g(0)}{2} + g(1) + \ldots + g(n-1) + \frac{g(n)}{2} \\
&\qquad + \sum_{k=1}^s \frac{B_{2k}}{(2k)!}\left(g^{(2k-1)}(0) - g^{(2k-1)}(n)\right) - \frac{B_{2s+2}}{(2s+2)!} n g^{(2s+2)}(\xi), \quad (11.3.3)
\end{aligned}$$

denn es gilt

$$\min_i g^{(2s+2)}(\xi_i) \le \frac{1}{n}\sum_{i=1}^n g^{(2s+2)}(\xi_i) \le \max_i g^{(2s+2)}(\xi_i),$$

was die Existenz eines $\xi \in (\min_i \xi_i, \max_i \xi_i)$ für ein stetiges $g^{(2s+2)}$ mit der Eigenschaft

$$g^{(2s+2)}(\xi) = \frac{1}{n}\sum_{i=1}^n g^{(2s+2)}(\xi_i)$$

impliziert. Für ein beliebiges Intervall $[a, b]$ mit Bruchstellen $t_k = a + kh$ und $h = \frac{b-a}{n}$ für $k = 0, \ldots, n$ leiten wir die (11.3.3) entsprechende Summenformel mittels Variablensubstitution wie folgt her: setze $g(x) = f(a + xh)$. Dann gilt

- $\int_0^n g(x)dx = \int_0^n f(a+xh)dx = \int_a^b f(t)\frac{dt}{h} = \frac{1}{h}\int_a^b f(t)dt$;
- $g^{(k)}(x) = f^{(k)}(a+xh)h^k,\ k \ge 0$;
- $\frac{1}{2}g(0) + g(1) + \ldots + g(n-1) + \frac{g(n)}{2} = \frac{f(a)}{2} + f(a+h) + f(a+2h) + \ldots + f(b-h) + \frac{f(b)}{2} = \frac{1}{h}T(h)$, wobei $T(h) = R(h)$ der Trapezsumme aus Beispiel 11.1.4 entspricht.

Damit ergibt aus (11.3.3) insgesamt

$$\frac{1}{h}\int_a^b f(t)dt = \frac{1}{h}T(h) + \sum_{k=1}^{s} \frac{B_{2k}}{(2k)!}h^{2k-1}\left(f^{(2k-1)}(a) - f^{(2k-1)}(b)\right) - \frac{B_{2s+2}}{(2s+2)!}nh^{2s+2}f^{(2s+2)}(\xi)$$

mit $\xi \in (a,b)$ oder äquivalent dazu

$$T(h) = \int_a^b f(t)dt + \sum_{k=1}^{s} \frac{B_{2k}}{(2k)!}h^{2k}\left(f^{(2k-1)}(b) - f^{(2k-1)}(a)\right) + \frac{B_{2s+2}}{(2s+2)!}(b-a)h^{2s+2}f^{(2s+2)}(\xi). \tag{11.3.4}$$

Beachte, daß dies eine Entwicklung der Trapezsumme nach Potenzen von $h = \frac{b-a}{n}$ ist. Daher läßt (11.3.4) sich auch als

$$T(h) = \tau_0 + \tau_1 h^2 + \tau_2 h^4 + \ldots + \tau_s h^{2s} + \alpha_{s+1}(h)h^{2s+2} \tag{11.3.5}$$

schreiben, wobei $\tau_0 := \int_a^b f(t)dt = I(f)$ das gesuchte Integral, $\tau_i := \frac{B_{2i}}{(2i)!}\left(f^{(2i-1)}(b) - f^{(2i-1)}(a)\right)$ unabhängig von h und $\alpha_{s+1}(h) := \frac{B_{2s+2}}{(2s+2)!}(b-a)f^{(2s+2)}(\xi(h))$, $\xi = \xi(h) \in (a,b)$ das Restglied mit

$$|\alpha_{s+1}(h)| \le \left|\frac{B_{2s+2}}{(2s+2)!}(b-a)\right| \left\|f^{(2s+2)}\right\|_\infty \tag{11.3.6}$$

ist. Entwicklungen der Form (11.3.5), 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 $T(0) = \tau_0 = \int_a^b f(t)dt$. Als nächstes wird anhand der asymptotischen Entwicklung für die Trapezsumme (11.3.5) gezeigt, wie man die Genauigkeit beim Integrieren verbessern kann.

11.4 Extrapolation

Das wichtige Prinzip der Extrapolation ist in der Numerik eine allgemeine Methode zur Verbesserung der Genauigkeit. Im Fall der numerischen Integration läßt sich dies für die Trapezregel wie folgt verwenden: Oben haben wir gesehen, daß die Trapezsumme $T(h) = h\left(\frac{f(a)}{2} + f(a+h) + \ldots + f(b-h) + \frac{f(b)}{2}\right)$ in (11.1.4) zur Berechnung von $\int_a^b f(x)dx$ eine Approximation der Ordnung h^2 ist. Andererseits besitzt $T(h)$ eine asymptotische Entwicklung (11.3.5), falls $f \in \mathcal{C}^{2s+2}([a,b])$:

$$T(h) - \tau_0 = \tau_1 h^2 + \tau_2 h^4 + \tau_3 h^6 + \ldots + \tau_s h^{2s} + O(h^{2s+2}). \tag{11.4.1}$$

Die Idee der Extrapolation besteht nun darin, zur Verbesserung der Genauigkeit den Term $T(h)$ für Schrittweiten

$$h_0 = h, \quad h_1 = \frac{h}{2}, \quad h_2 = \frac{h}{4}, \quad \ldots, \quad h_m = \frac{h}{2^m}$$

zu betrachten. Bilde anschließend aus $T(h_0),\ T(h_1),\ T(h_2),\ \ldots,\ T(h_m)$ das Interpolationspolynom (in h^2), das dann an der Stelle $h = 0$ ausgewertet wird. Dieses Verfahren heißt Extrapolation, da $0 \notin \{h_0, \ldots, h_m\}$.

Diese Idee wenden wir auf (11.4.1) an: Bildung von $T\left(\frac{h}{2}\right)$ ergibt

$$\begin{aligned} T\left(\frac{h}{2}\right) - \tau_0 &= \tau_1\frac{h^2}{4} + \frac{\tau_2}{2^4}h^4 + \frac{\tau_3}{2^6}h^6 + \ldots + \frac{\tau_s}{2^{2s}}h^{2s} + O(h^{2s+2}) \\ &=: \frac{\tau_1}{4}h^2 + \hat{\tau}_2 h^4 + \hat{\tau}_3 h^6 + \ldots \end{aligned}$$

Nach Multiplikation mit 4/3 und Subtraktion von 1/3–mal (11.4.1) fällt der h^2–Term heraus, so daß

$$\begin{aligned} & \left(\frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h)\right) - \frac{4}{3}\tau_0 + \frac{1}{3}\tau_0 = \tilde{\tau}_2 h^4 + \tilde{\tau}_3 h^6 + \dots \\ \iff \quad & \left(\frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h)\right) - I(f) = \tilde{\tau}_2 h^4 + \tilde{\tau}_3 h^6 + \dots \end{aligned} \tag{11.4.2}$$

gilt. Durch einfaches Kombinieren der Trapezsumme bzgl. h und bzgl. $\frac{h}{2}$ läßt sich die Genauigkeit also auf h^4 steigern.

Die Annäherungsformel

$$T_1(h) := \frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h) \tag{11.4.3}$$

läßt sich auch wie folgt erklären: man bestimmt das lineare Interpolationspolynom der Funktion

$$T(\sqrt{x}) = \tau_0 + \tau_1 x + \tau_2 x^2 + \dots + \tau_s x^s + O(x^{s+1})$$

mit $x > 0$ zu den Punkten

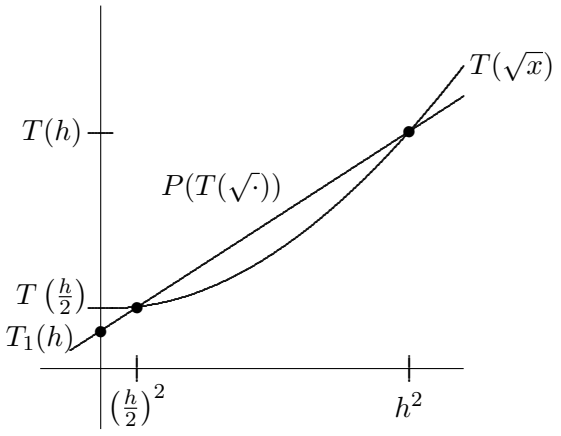
$$\left(h^2, T(h)\right) \quad \text{und} \quad \left(\left(\frac{h}{2}\right)^2, T\left(\frac{h}{2}\right)\right),$$

was

$$P\left(T(\sqrt{\cdot})\Big|h^2, \left(\frac{h}{2}\right)^2\right)(x) = T(h) + \frac{T\left(\frac{h}{2}\right) - T(h)}{\left(\frac{h}{2}\right)^2 - h^2}(x - h^2)$$

ergibt, und wertet P an der Stelle $x = 0$ aus (extrapoliert, weil man $T(0)$ annähern will), d.h.

$$\begin{aligned} P\left(T(\sqrt{\cdot})\Big|h^2, \left(\frac{h}{2}\right)^2\right)(0) &= T(h) + \frac{T\left(\frac{h}{2}\right) - T(h)}{\left(\frac{h}{2}\right)^2 - h^2}(-h^2) \\ &= T(h) + \left(T\left(\frac{h}{2}\right) - T(h)\right)\frac{4}{3} \\ &= \frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h) \\ &= T_1(h). \end{aligned}$$



Eine Wiederholung obiger Idee mit $T_1(h) = \frac{4}{3}T\left(\frac{h}{2}\right) - \frac{1}{3}T(h)$ aus (11.4.3) liefert folgendes: nach (11.4.2) gilt

$$\begin{aligned} T_1(h) - I(f) &= \tilde{\tau}_2 h^4 + \tilde{\tau}_3 h^6 + \ldots + O(h^{2s+2}) \\ T_1\left(\frac{h}{2}\right) - I(f) &= \tilde{\tau}_2 \frac{h^4}{16} + \tilde{\tau}_3 \frac{h^6}{64} + \ldots + O(h^{2s+2}). \end{aligned}$$

Multipliziere die erste Gleichung mit $\frac{1}{15}$ und ziehe sie vom $\frac{16}{15}$–fachen der zweiten Gleichung ab. Dann gilt

$$\begin{aligned} & \frac{16}{15}\left(T_1\left(\frac{h}{2}\right) - I(f)\right) - \frac{1}{15}\left(T_1(h) - I(f)\right) = \overline{\tau}_3 h^6 + \ldots + O(h^{2s+2}) \\ \iff \quad & \frac{16}{15}T_1\left(\frac{h}{2}\right) - \frac{1}{15}T_1(h) - I(f) = \overline{\tau}_3 h^6 + \ldots + O(h^{2s+2}), \end{aligned}$$

d.h.

$$T_2(h) := \frac{16T_1(\frac{h}{2}) - T_1(h)}{15}$$

hat einen Fehler der Ordnung 6. $T_2(h)$ läßt sich wie $T_1(h)$ über *Extrapolation*, erklären, d.h. es gilt

$$P\left(T\sqrt{\cdot}\,\middle|\, h^2, \left(\frac{h}{2}\right)^2, \left(\frac{h}{4}\right)^2\right)(0) = T_2(h).$$

$T_2(h)$ ist also die Auswertung an $x = 0$ des *quadratischen* Interpolationspolynoms von $T(\sqrt{\cdot})$ an den Stützstellen $h^2, \left(\frac{h}{2}\right)^2, \left(\frac{h}{4}\right)^2$.

Allgemein gilt folgendes Schema: Sei h eine feste Anfangsschrittweite (z.B. $h = b - a$). Definiere

$$\begin{aligned} T_{i,0} &:= T(2^{-i}h),\ i = 0, 1, 2, \ldots \\ T_{i,j} &:= \frac{4^j T_{i,j-1} - T_{i-1,j-1}}{4^j - 1},\ j = 1, 2, \ldots,\ i \geq j. \end{aligned}$$

Dies ergibt das sogenannte Romberg–Schema aus den 50'er Jahren:

$$\begin{array}{lccccccc} T(h) = T_{0,0} & & & & & & & \\ & \searrow & & & & & & \\ T\left(\frac{h}{2}\right) = T_{1,0} & \to & T_{1,1} & & & & & \\ & \searrow & & \searrow & & & & \\ T\left(\frac{h}{4}\right) = T_{2,0} & \to & T_{2,1} & \to & T_{2,2} & & & \\ & \searrow & & \searrow & & \searrow & & \\ T\left(\frac{h}{8}\right) = T_{3,0} & \to & T_{3,1} & \to & T_{3,2} & \to & T_{3,3} & \\ \vdots & & & & & & \ddots & \\ T\left(\frac{h}{2^s}\right) = T_{s,0} & \to & & & \cdots & & \to & T_{s,s} \end{array} \tag{11.4.4}$$

Das Romberg–Schema entspricht dem Schema von Neville–Aitken aus Kapitel 8 zur Auswertung von P an der Stelle $x = 0$. Es ist so angelegt, daß der Fehler in der j–ten Spalte von der Ordnung $2j$ ist, d.h. $T_{s,s}$ liefert eine Approximation von für $I(f)$ von der Ordnung $2s$.

Bei der *praktischen Durchführung* des Extrapolationstableaus (11.4.4) muß man die Werte $T(h), T\left(\frac{h}{2}\right), T\left(\frac{h}{4}\right), \ldots, T\left(\frac{h}{2^s}\right)$ berechnen. Dabei greift man auf schon berechnete Werte zurück: für $h = b - a$ etwa gilt $T(h) = h\left(\frac{f(a)}{2} + \frac{f(b)}{2}\right)$. Weiter ist $T\left(\frac{h}{2}\right) = h\left(\frac{f(a)}{2} + f(a + \frac{h}{2}) + \frac{f(b)}{2}\right)$, so daß nur $f(a + \frac{h}{2})$ neu berechnet werden muß usw.

Ein *praktisches Problem* ist, daß man oft nicht von vornherein weiß, wie man s zu wählen hat, damit das Integral mit der gewünschten Genauigkeit approximiert wird (sonst könnte man zuerst alle Funktionswerte $f(a+2^{-s}h)$ 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 $\int_a^b |f(x)|dx$ und ε klein ist.

Das ROMBERG–Schema in (11.4.4) beruht jeweils auf Halbierung der Schrittweite, also der ROMBERG–Folge

$$h_0 = b-a,\ h_1 = \frac{h}{2},\ h_2 = \frac{h}{4},\ \ldots$$

Es gibt auch andere, z.B. die BULIRSCH–Folge

$$h_0 = b-a,\ h_1 = \frac{h_0}{2},\ h_2 = \frac{h_0}{3},\ \ldots$$

Der Vorteil hier ist, daß die Rechenarbeit bei der Auswertung von f nicht so stark ansteigt, aber man dafür auch den Fehler nicht so stark reduziert.

Ausgangspunkt für das Extrapolationsschema war die Existenz einer asymptotischen Entwicklung (11.4.1). Daher ist auch eine Verallgemeinerung auf andere Situationen möglich, z.B. numerische Differentiation, gewisse Diskretisierungsschemata bei gewöhnlichen Differentialgleichungen sowie Fehlerschätzungen (vergl. hierzu [S], [DH]).

11.5 GAUSS–Quadratur

Bei den NEWTON–COTES–Formeln in (11.2.4) entsprechen sich im wesentlichen der Grad der Exaktheit (d.h. die Polynome werden bis zu diesem Grad exakt integriert) und die Anzahl der Stützstellen. Letztere waren zunächst willkürlich als äquidistant vorgegeben worden. Wir gehen nun der Frage nach, ob man durch eine geeignete Wahl der Stützstellen und Gewichte den Exaktheitsgrad erhöhen kann. Wir halten zunächst fest, daß man in der Formel

$$I_m(f) = h\sum_{j=0}^{m} c_j f(x_j)$$

bedingt durch die Stützstellen $x_0, \ldots, x_m$ und die Gewichte $c_0, \ldots, c_m$ $2m+2$ Freiheitsgrade hat. Unsere Aufgabe sei es nun, das Integral

$$\int_c^d \omega(x) f(x) dx \tag{11.5.1}$$

mit einer Gewichtsfunktion $\omega(x) > 0$ durch Ausdrücke

$$\sum_{j=0}^{m} \omega_j f(x_j) \tag{11.5.2}$$

zu approximieren, so daß die Integration exakt für Polynome möglichst hohen Grades ist, d.h. so daß

$$\int_c^d \omega(x) f(x) dx = \sum_{j=0}^{m} \omega_j f(x_j) \tag{11.5.3}$$

für alle $f \in \mathcal{P}_N$ ist für möglichst großes N. Wenn die x_i hier äquidistant gewählt werden, so entstehen die NEWTON–COTES–Formeln mit $N \in \{m, m+1\}$.

Bemerkung 11.5.4 Es gilt $N < 2m+2$, denn: Sei $N = 2m+2$ und wähle $P(x) = \prod_{i=0}^{m}(x-x_i)^2 \in \mathcal{P}_{2m+2}$. Dann gilt

$$0 = \sum_{j=0}^{m} \omega_j P(x_j) = \int_c^d \omega(x)P(x)dx > 0.$$

Widerspruch, da $P(x) > 0$ als quadratisches Polynom ist. ∎

Aber $N = 2m + 1$ ist möglich:

Satz 11.5.5 Es seien $\omega(x) > 0$ eine feste Gewichtsfunktion auf dem Intervall $[c, d]$ und $\langle f, g\rangle := \int_c^d \omega(x)f(x)g(x)dx$ das zugehörige gewichtete Skalarprodukt. Zu beliebigem $m \in \mathbb{N}$ existieren eindeutige, paarweise verschiedene Stützstellen $x_0, \ldots, x_m \in (c, d)$ und eindeutige Gewichte $\omega_0, \ldots, \omega_m$, so daß

$$\sum_{j=0}^{m} \omega_j f(x_j) = \int_c^d \omega(x)f(x)dx + E(f) \tag{11.5.6a}$$

und

$$E(f) = 0 \tag{11.5.6b}$$

für alle $f \in \mathcal{P}_{2m+1}$ gilt, d.h. die Quadratur (11.5.6a) ist exakt vom Grad $2m + 1$. Genauer gilt: Die x_j sind die Nullstellen des $(m + 1)$–ten Orthogonalpolynoms P_{m+1} bezüglich ω und

$$\omega_j = \int_c^d \omega(x) \prod_{\substack{k=0 \\ k\neq j}}^{m} \frac{x - x_k}{x_j - x_k} dx > 0, \tag{11.5.7}$$

wobei $\ell_{jm}(x) := \prod_{\substack{k=0 \\ k\neq j}}^{m} \frac{x-x_k}{x_j-x_k}$ die LAGRANGE–Fundamentalpolynome aus (8.3.5) sind. Für den Fehler gilt, falls $f \in \mathcal{C}^{2m+2}([c, d])$ ist,

$$\begin{aligned} E(f) &= \frac{f^{(2m+2)}(\xi)}{(2m+2)!}\langle P_{m+1}, P_{m+1}\rangle \\ &= \frac{f^{(2m+2)}(\xi)}{(2m+2)!}\int_c^d \omega(x)\prod_{i=0}^{m}(x-x_i)^2 dx \end{aligned} \tag{11.5.8}$$

mit $\xi \in (c, d)$, d.h.

$$|E(f)| \leq \|P_{m+1}\|^2 \frac{\|f^{(2m+2)}\|_{\infty,[c,d]}}{(2m+2)!}.$$

∎

Orthogonalpolynome sind Folgen $\{P_k\}_{k\geq 0}$ von Polynomen, exakt vom Grad k, die für $i \neq j$ die Gleichung

$$\int_a^b \omega(x)P_i(x)P_j(x)dx = 0$$

erfüllen. Man kann zeigen, daß es zu jedem Skalarprodukt (mit Gewichtsfunktion ω) eindeutig bestimmte Orthogonalpolynome mit führendem Koeffizienten 1 gibt. Jedes Orthogonalpolynom hat genau k einfache Nullstellen $a < x_i < b$ für $i = 0, \ldots, k-1$, so daß die Darstellung

$$P_k(x) = \prod_{i=0}^{k-1} (x - x_i)$$

existiert. Die folgende Tabelle listet gängige Orthogonalpolynome auf.

Bezeichnung	$[a, b]$	Gewicht ω	$\langle P_{m+1}, P_{m+1} \rangle$
Gauss–Legendre	$[-1, 1]$	1	$\frac{2}{2m+1}$
Gauss–Tschebyscheff	$[-1, 1]$	$(1-t^2)^{-1/2}$	$\frac{\pi}{2}$ für $m > 0$
Gauss–Laguerre	$[0, \infty)$	$t^\alpha e^{-t},\ \alpha > -1$	$\frac{\Gamma(\alpha+m+1)}{m!}$
Gauss–Hermite	$(-\infty, \infty)$	e^{-t^2}	$\sqrt{\pi} 2^n n!$

Die Gauss–Legendre–Polynome kommen nur selten vor, da die Trapezsummenextrapolation meist überlegen ist. Bei den Gauss–Tschebyscheff–Polynomen kann die Extrapolation nicht angewandt werden, da ω bei $|t|$ schwach singulär ist. Die Güte der Approximation kann für $\omega \neq 1$ nur durch Erhöhung der Ordnung, nicht durch Zerlegung in Teilintegrale erfolgen.

Beweis von Satz 11.5.5: Sei P_{m+1} das Orthogonalpolynom bzgl. ω vom Grad $m+1$. Für jedes Polynom $P \in \mathcal{P}_{2m+1}$ lassen sich nach dem Euklid'schen Algorithmus Polynome $Q, R \in \mathcal{P}_m$ konstruieren, so daß

$$P = P_{m+1} Q + R$$

gilt. Eine Auswertung von P_{m+1} an den Nullstellen $x_0, \ldots, x_m$ ergibt

$$P(x_j) = P_{m+1}(x_j) Q(x_j) + R(x_j) = R(x_j)$$

für alle $j = 0, \ldots, m$. Betrachte R als Interpolationspolynom. Die Lagrange–Darstellung von R zu x_j lautet

$$R(x) = \sum_{j=0}^{m} R(x_j) \ell_{jm}(x) = \sum_{j=0}^{m} P(x_j) \ell_{jm}(x),$$

was

$$\begin{aligned}
I(P) &= \int_c^d \omega(x) P(x) dx \\
&= \int_c^d \omega(x) P_{m+1}(x) Q(x) dx + \int_c^d \omega(x) R(x) dx \\
&= \int_c^d \omega(x) R(x) dx \\
&= \sum_{j=0}^{m} P(x_j) \int_c^d \omega(x) \ell_{jm}(x) dx \\
&= \sum_{j=0}^{m} P(x_j) \omega_j
\end{aligned}$$

impliziert und die Gleichungen (11.5.6a) und (11.5.6b) zeigt. Bleibt noch $\omega_j > 0$ zu zeigen. Definiere

$$S_j(x) := \prod_{\substack{k=0 \\ k \neq j}}^{m} \left(\frac{x - x_k}{x_j - x_k} \right)^2 = \ell_{jm}^2 \in \mathcal{P}_{2m}$$

für $j = 0, \ldots, m$. Dann gilt

$$0 < \int_c^d \omega(x) S_j(x) dx = \sum_{i=0}^{m} \omega_i S_j(x_i) = \omega_j$$

für $j = 0, \ldots, m$. Die Darstellung des Fehlers (11.5.8) folgt aus der Darstellung des Fehlers bei der Polynominterpolation, der TAYLOR–Entwicklung und den Eigenschaften der Orthogonalpolynome [DH]. ∎

11.6 Schwierige Integranden

Bei den bisher diskutierten Methoden zur Berechnung von $\int_a^b f(x)dx$ treten Probleme auf, wenn f *unstetig* ist. Wenn Unstetigkeitsstellen bekannt sind, unterteile das Integral dort. Sind diese nicht bekannt und werden nicht mitbehandelt, liegt keine Konvergenz vor bzw. treten falsche Ergebnisse auf. In solchen Fällen sollte man *adaptive Quadraturverfahren* verwenden ([DH]). Bei Polstellen liegt der sogenannte *Nadelimpuls* (cusp) vor, so daß jedes Quadraturprogramm versagt, wenn die Spitze genügend schmal ist.

Im Fall, daß f *schwach singulär* ist, d.h. die Ableitung $f^{(k)}$ nicht für ein oder mehrere k existiert, (z.B. bei

$$I(f) = \int_0^\pi \sqrt{t} \cos t dt$$

hat die Ableitung $f'(t) = \frac{\cos t}{2\sqrt{t}} + \sqrt{t}(-\sin t)$ bei $t = 0$ eine Polstelle) werden adaptive Quadraturverfahren extrem langsam und nicht–adaptive liefern falsche Ergebnisse. Häufig ist es in solchen Fällen möglich, die Singularität durch eine Substitution zu beseitigen. Setze oben z.B. $s = \sqrt{t}$. Dann wird das Integral zu

$$\int_0^\pi \sqrt{t} \cos t dt = \int_0^{\sqrt{\pi}} s \cos s^2(2s) ds.$$

Als *Faustregel* sollte man generell, wenn man nicht weiß, ob f glatt ist, die Quadraturmethode sorgfältig wählen.

11.7 Zweidimensionale Integration

Quadraturformeln auf dem Einheitsquadrat $[0, 1]^2$ und dem Einheitssimplex lassen sich mittels affiner Transformationen auf beliebige Rechtecke und Dreiecke anwenden, da affine Transformationen den Grad der Exaktheit erhalten.

Integration über das Einheitsquadrat

Hier ergeben sich in natürlicher Weise Produktformeln aus den univariaten Formeln. Zur Berechnung von

$$I(f) := \int_0^1 \int_0^1 f(x, y)\ dx\, dy$$

kann man in jede Richtung die NEWTON–COTES–Formeln (11.2.4) einsetzen (Tensorproduktansatz). Es gilt dann

$$I(f) \approx h^2 \sum_{i=0}^{m} \sum_{j=0}^{m} c_i c_j f(x_i h, y_i h) =: I_m.$$

Man kann dann zeigen, daß I_m exakt für alle Polynome $p \in \text{span}\{x^{k_1} y^{k_2} :\ 0 \le k_1, k_2 \le m\}$ ist.

Integration über das Einheitsdreieck

Hier besteht die Aufgabe darin, Quadraturformeln zu finden, so daß alle Monome der Form $x^{k_1}y^{k_2}$, $0 \leq k_1, k_2$ und $k_1 + k_2 \leq m$ exakt integriert werden. Beispiele vom Exaktheitsgrad 1 sind etwa

$$Q(f) = \frac{1}{2} f\left(\frac{1}{3}, \frac{1}{3}\right) \quad \text{oder} \quad Q(f) = \frac{1}{6}\left(f(0,0) + f(1,0) + f(0,1)\right).$$

Vom Exaktheitsgrad 2 sind

$$Q(f) = \frac{1}{6}\left[f\left(\frac{1}{2}, 0\right) + f\left(0, \frac{1}{2}\right) + f\left(\frac{1}{2}, \frac{1}{2}\right)\right]$$

oder

$$Q(f) = \frac{1}{6}\left[f\left(\frac{1}{6}, \frac{1}{6}\right) + f\left(\frac{2}{3}, \frac{1}{6}\right) + f\left(\frac{1}{6}, \frac{2}{3}\right)\right].$$

Für weitere Details und andere Verfahren sei auf den Übersichtsartikel [C] hingewiesen.

Schlußbemerkungen

Die Inhalte der Kapitel 1 bis 11 bilden eine Ansammlung verschiedener Techniken zur Lösung verschiedener Probleme: Interpolation, numerische Lösung von Gleichungen und Gleichungssystemen etc. Gemeinsamer Leitfaden war, diese Probleme *rechenbar* zu machen, d.h. sie in eine für Rechner geeignete Form zu bringen. In dieser Vorlesung wurden dazu die wichtigsten Prinzipien aus den einzelnen Bereichen vorgestellt. Zwei nicht behandelte Themen sind die *Numerische Differentiation* und die schnelle FOURIER–Transformation. Den Rest der Vorlesung wird die *Numerik gewöhnlicher Differentialgleichungen* einnehmen.

A Differenzenverfahren für elliptische Differentialgleichungen

Das *Differenzenverfahren* ist eine eher klassische Methode, um die Lösung einer partiellen Differentialgleichung zu ermitteln. Heute nimmt man dazu in aller Regel *Variationsansätze*. Man teilt partielle Differentialgleichungen in drei Gruppen ein. Dies sind *elliptische, hyperbolische* und *parabolische* Differentialgleichungen. Die Diskretisierung elliptischer Differentialgleichungen führt auf große lineare Gleichungssysteme mit besonderer Struktur. Diese dienen als Prototyp, um Lösungsmethoden für lineare Gleichungssysteme zu testen. Daher können wir die in Kapitel 3 erworbenen Kenntnisse auf dieses Problem anwenden. Sei $\Omega \subset \mathbb{R}^2$ ein offenes und zusammenhängendes Gebiet mit dem Rand $\partial\Omega$. Gegeben seien die Funktionen

$$\begin{aligned} f &: \Omega \to \mathbb{R} \\ g &: \partial\Omega \to \mathbb{R}. \end{aligned}$$

Wir machen uns nun auf die Suche nach einer Funktion $u : \overline{\Omega} \to \mathbb{R}$, die das DIRICHLET–Problem

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= g \quad \text{auf } \partial\Omega \end{aligned} \tag{A1}$$

löst, wobei

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

der LAPLACE–Operator ist. Die erste Gleichung von (A1) ist die sogenannte POISSON–Gleichung. Unsere Aufgabe sei es nun, das Problem (A1) approximativ zu lösen. Dafür nehmen wir $\Omega = (0,1)^2$ an. Um dies zu bewerkstelligen, wählen wir die folgende Diskretisierung des Gebiets. Wir legen ein uniformes, d.h. äquidistantes Gitter der Schritt– oder Maschenweite $h = \frac{1}{n}$ über Ω. Man vergleiche hierzu die Skizze am Anfang von Kapitel 5 bei der Lösung des STURM–LIOUVILLE–Problems. Damit ist die Menge der inneren Gitterpunkte

$$\Omega_h := \{(x,y) \in \Omega : x = ih, \quad y = jh, \quad i,j = 1,\ldots,N-1\}$$

und der Randpunkte

$$\partial\Omega_h := \{(x,y) \in \partial\Omega : x = ih, \quad y = jh, \quad i,j = 0,\ldots,N\}.$$

Jetzt berechnen wir Näherungen für die Funktionswerte u auf Ω_h, mit

$$u_h(z) \approx u(z)$$

mit $z \in \Omega_h$. Die Idee hierzu ist, daß man die Differentialgleichung (A1) diskretisiert und das hierdurch entstehende Gleichungssystem löst. Dies machen wir auf die folgende Art und Weise: Wir ersetzen die Terme $\frac{\partial^2}{\partial x^2}$ und $\frac{\partial^2}{\partial y^2}$ durch die jeweiligen Differenzenquotienten

$$\begin{aligned} \frac{\partial^2 u(x,y)}{\partial x^2} &\approx \frac{u(x-h,y) - 2u(x,y) + u(x+h,y)}{h^2} \\ \frac{\partial^2 u(x,y)}{\partial y^2} &\approx \frac{u(x,y-h) - 2u(x,y) + u(x,y+h)}{h^2}. \end{aligned} \tag{A2}$$

Im folgenden gelte die Notation $u_{ij} \approx u(ih,jh)$. Damit gilt

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2}(ih, jh) &\approx \frac{1}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) \\ \frac{\partial^2 u}{\partial y^2}(ih, jh) &\approx \frac{1}{h^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}), \end{aligned} \tag{A3}$$

und die diskretisierte Form von $-\Delta u = f$ ist

$$\frac{1}{h^2}(4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}) = f_{ij} \tag{A4}$$

mit $f_{ij} := f(ih, jh)$ und $i, j = 1, \ldots, N-1$. Zur Berechnung von $u_{i,j}$ werden Nachbarwerte benutzt. Dies kürzt man mittels eines 5–Punkte–Differenzensterns

$$-\Delta \approx \Delta_h := \frac{1}{h^2} \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

ab. Dieser ist nützlich für die geometrische Darstellung der Approximation auf Ω_h. Zur Berechnung der u_{ij} über lineare Gleichungssysteme ist eine derartige Festlegung nötig. Wir nehmen hier die lexikographische Numerierung von links unten horizontal nach rechts oben. Da die Randwerte durch die Problemstellung bereits bekannt sind und nur Werte in Ω_h gesucht sind, gibt es insgesamt

$$n := (N-1)^2$$

Unbekannte. (A4) liefert somit ein Gleichungssystem der Größe n:

$$A\underline{u} = \underline{\tilde{f}} \tag{A5}$$

mit $\underline{u} = (u_{1,1}, u_{1,2}, u_{1,3}, \ldots, u_{N-1,N-1})^T$, $\underline{tf} = (\tilde{f}_{1,1}, \tilde{f}_{1,2}, \ldots)^T$ und

$$A := \frac{1}{h^2} \begin{pmatrix} T & -I & & & \\ -I & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & T \end{pmatrix} \in \mathbb{R}^{n \times n},$$

wobei $I \in \mathbb{R}^{(N-1)\times(N-1)}$ die Einheitsmatrix und

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}$$

sind. Die entstandene Matrix A hat die folgenden Merkmale. Sie ist in der Regel recht groß und hat in jede Richtung $(N-1)$ Freiheitsgrade. Weiter ist sie symmetrisch positiv definit und dünn besetzt, denn es gibt weniger als $5n$ von Null verschiedene Einträge. Daher bietet sich zur Lösung des linearen Gleichungssystems das CHOLESKY–Verfahren an.

B Liste mit numerischen Vokabeln

nach: Baule, Rainer: Mathematisches Wörterbuch — Numerik — Englisch – Deutsch, `http://www.math.uni-goettingen.de/baule/wbuch.html`

absolute error	absoluter Fehler
accuracy	Genauigkeit
algorithm	Algorithmus
alternation theorem	Alternantensatz
application	Anwendung
approximation	Approximation
auxiliary variable	Hilfsvariable
back substitution	Rückwärtseinsetzen
backward (error) analysis	Rückwärts(fehler)analyse
Banach fixed–point theorem	Banach'scher Fixpunktsatz
band(ed) matrix	Bandmatrix
basis	Basis
basis function	Basisfunktion
basis variable	Basisvariable
Bernoulli polynomial	Bernoulli–Polynom
Bernstein polynomial	Bernstein–Polynom
Bezier curve	Bezier–Kurve
bisection method	Bisektionsverfahren
Chebyshev approximation	Tschebyscheff–Approximation
Chebyshev polynomial	Tschebyscheff–Polynom
Chebyshev system	Tschebyscheff–System
Cholesky decomposition	Cholesky–Zerlegung
column interchange	Spaltenvertauschung
column pivot search	Spaltenpivotsuche
column pivoting	Spaltenpivotisierung
column sum criterion	Spaltensummenkriterium
column–sum norm	Spaltensummennorm
compatible	verträglich; passend (Norm)
complete pivoting	Totalpivotisierung
complexity	Aufwand
condition	Kondition
condition number	Konditionszahl
conditionally positive definite	bedingt positiv definit
contractive	kontrahierend
convergence	Konvergenz
convergence acceleration	Konvergenzbeschleunigung
convergence criterion	Konvergenzkriterium
convex	konvex
convex combination	Konvexkombination
coordinate	Koordinate
critical point	kritischer Punkt
cubic spline	Kubischer Spline
curve	Kurve
data	Daten

data–dependent	datenabhängig
decomposable	zerlegbar
decomposition	Zerlegung
decomposition method	Zerlegungsverfahren
definite, positive / negative	definit, positiv / negativ
deflation	Deflation
degenerate	ausgeartet
degree	Grad
degree of freedom	Freiheitsgrad
diagonally dominant	diagonaldominant
difference quotient	Differenzenquotient
difference scheme	Differenzenschema
discrete	diskret
discretization	Diskretisierung
duality	Dualität
efficiency	Effizienz
eigenvalue problem	Eigenwertproblem
elimination	Elimination
elimination method	Eliminationsverfahren
equidistant	äquidistant
equilibration	Äquilibrierung
error	Fehler
error analysis	Fehleranalyse
error bound	Fehlerschranke
error estimate	Fehlerabschätzung
error law	Fehlergesetz
estimate	Abschätzung
extrapolation	Extrapolation
Fast Fourier Transform (FFT)	schnelle Fourier–Transformation
field	Körper
fixed–point	Fixpunkt
fixed–point arithmetic	Festkommaarithmetik
fixed–point theorem	Fixpunktsatz
floating–point arithmetic	Fließkommaarithmetik
floating–point number	Fließkommazahl
forward (error) analysis	Vorwärts(fehler)analyse
forward substitution	Vorwärtseinsetzen
Fourier transform	Fourier–Transformation
Frobenius norm	Frobenius–Norm
Gauss elimination	Gauss–Elimination
Gauss quadrature	Gauss–Quadratur
Gauss–Jordan method	Gauss–Jordan–Verfahren
Gauss–Seidel method	Gauss–Seidel–Verfahren
Gaussian	Gauss–Glocke
Givens rotation	Givens–Rotation
gradient	Gradient
Hermite interpolation	Hermite–Interpolation
Hermite polynomial	Hermite–Polynom
Hessenberg matrix	Hessenberg–Matrix
Hessenberg form	Hessenberg–Form

Hilbert matrix	Hilbert–Matrix
Horner scheme	Horner–Schema
Householder matrix	Householder–Matrix
Householder transformation	Householder–Transformation
ill–conditioned	schlecht konditioniert
ill–posed	schlecht gestellt
indecomposable	unzerlegbar
initial value	Startwert
input data	Eingabedaten
input data error	Eingabefehler
input set	Eingabemenge
instability	Instabilität
instable	instabil
interpolation	Interpolation
interval arithmetic	Intervallarithmetik
iteration	Iteration
iteration matrix	Iterationsmatrix
iteration method	Iterationsverfahren
iterative method	Iterationsverfahren
kernel	Kern
knot, node	Knoten
Lagrange interpolation formula	Lagrange–Interpolationsformel
Landau symbol	Landau–Symbol
least squares method	Methode der kleinsten Quadrate
linear system of equations	Lineares Gleichungssystem
lower triangular matrix	untere Dreiecksmatrix
machine accuracy	Maschinengenauigkeit
matrix norm subordinate to the v. n.	zugeordnete Matrixnorm (einer V–Norm)
method	Methode; Verfahren
method error	Verfahrensfehler
method of conjugate gradients	konjugierte–Gradienten–Verfahren
midpoint rule	Mittelpunktregel
natural spline	natürlicher Spline
negative definite	negativ definit
Newton's method	Newton–Verfahren
Newton–Cotes formula	Newton–Cotes–Formel
nondegenerate	nicht ausgeartet
norm	Norm
normal equation	Normalgleichung
normed space	normierter Raum
numerical	numerisch
operator	Operator
operator norm	Operatornorm
optimization	Optimierung
order of convergence	Konvergenzordnung
output set	Ausgabemenge
over–determined	überbestimmt
overrelaxation	Overrelaxation
partial pivoting	Teilpivotisierung
partition of unity	Teilung der Eins

PEANO kernel	PEANO–Kern
permutation	Permutation
perturbation	Störung
perturbation lemma	Störungslemma
perturbation matrix	Störungsmatrix
perturbation theory	Störungstheorie
pivot element	Pivotelement
pivot search	Pivotsuche
pivoting	Pivotisierung
point	Punkt
polyhedron	Polyeder
polynomial	Polynom
positive definite	positiv definit
positive semidefinite	positive semidefinit
power iteration	Potenzmethode
preconditioning	Präkonditionierung
program	Programm
programming	Programmierung
projection	Projektion
pseudoinverse	Pseudoinverse
QR decomposition	QR–Zerlegung
quadrature	Quadratur
radial	radial
RAYLEIGH quotient	RAYLEIGH–Quotient
reconstruction	Rekonstruktion
recurrence formula	Rekursionsformel
recursion	Rekursion
recursion formula	Rekursionsformel
reference	Referenz
regula falsi	Regula falsi
relative error	relativer Fehler
relaxation	Relaxation
remainder term	Restglied
REMEZ algorithm	REMES–Algorithmus
residual	Residuum
ROMBERG method	ROMBERG–Verfahren
rounding error	Rundungsfehler
roundoff error	Rundungsfehler
row equilibration	Zeilenäquilibrierung
row interchange	Zeilenvertauschung
row sum criterion	Zeilensummenkriterium
row–sum norm	Zeilensummennorm
secant method	Sekantenverfahren
semidefinite, positive / negative	semidefinit, positiv / negativ
seminorm	Seminorm
set	Menge
side condition	Nebenbedingung
signed integer	ganze Zahl
significant digit	signifikante Stelle
simplex	Simplex

simplex method	Simplexverfahren
SIMPSON rule; SIMPSON's rule	SIMPSON–Regel
singular value	Singulärwert
singular–value decomposition	Singulärwertzerlegung
slack	Schlupf
slack variable	Schlupfvariable
smooth	glatt
sparse	dünn besetzt
spectral norm	Spektralnorm
spectral radius	Spektralradius
speed of convergence	Konvergenzgeschwindigkeit
stability	Stabilität
stable	stabil
starting vertex	Startecke
strong column sum criterion	starkes Spaltensummenkriterium
strong row sum criterion	starkes Zeilensummenkriterium
subdivision	Subdivision
support	Träger
system of equations	Gleichungssystem
total pivoting	Totalpivotisierung
trapezoidal rule	Trapezregel
triangular decomposition	Dreieckszerlegung
triangular matrix, upper / lower	Dreiecksmatrix, obere / untere
under–determined	unterbestimmt
unsigned integer	natürliche Zahl
upper triangular matrix	obere Dreiecksmatrix
vertex	Ecke
weak column sum criterion	schwaches Spaltensummenkriterium
weak row sum criterion	schwaches Zeilensummenkriterium
well–conditioned	gut konditioniert
well–posed	gut gestellt

Literatur

[BS] F. Black, M. Scholes, *The Pricing of Options and Corporate Liabilities*, Journal of Political Economy **81**, 1973, pp. 637–654.

[Bo] C. de Boor, *A Practical Guide to Splines*, 1978, Springer, Berlin et al.

[Br] D. Braess, *Finite Elemente*, 2. Auflage 1997, Springer, Berlin et al.

[BM] R. Brealey, S. Myers, *Principles of Corporate Finance*, 7th Edition 2003, McGraw–Hill Publications, Boston et al.

[C] R. Cools, *Constructing cubature formulae: the science behind the art*, Acta Numerica, 1997, pp. 1–54

[CR] J. Cox, S. Ross et al. *Option Pricing: A Simplified Approch*, Journal of Financial Economics **7**, 1979, pp. 229–263.

[DR] W. Dahmen, A. Reusken, *Einführung in die numerische Mathematik für Ingenieure*, Vorlesungsskript, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 2002.

[DH] P. Deuflhard, A.Hohmann, *Numerische Mathematik 1. Eine algorithmisch orientierte Einführung*, 3. Auflage 2002, de Gruyter Lehrbuch, Berlin et al.

[GL] G.H. Golub, C.F. van Loan, *Matrix Computations*, 3rd Edition 1996, Johns Hopkins University Press, Baltimore et al.

[HH] G. Hämmerlin, K.-H. Hoffmann, *Numerische Mathematik*, 4. Auflage 1994, Springer, Berlin et al.

[H] M. Hanke–Bourgois, *Grundlagen der Numerischen Mathematik und des wissenschaftlichen Rechnens*, 1. Auflage 2002, Teubner, Stuttgart et al.

[KR] B. Kernigham, D. Ritchie, *Programmieren in C*, 2. Ausgabe 1990, Carl Hanser Verlag, München et al.

[L] C. Los, *Computational Finance*, 1. Auflage 2001, World Scientific, Singapore et al.

[OR] J. M. Ortega, W. C. Rheinholdt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, 1970

[Se] R. Seydel, *Tools for Computational Finance*, 1. Auflage 2002, Springer, Berlin et al.

[S] J. Stoer, *Numerische Mathematik 1*, 8. Auflage 1999, Springer, Berlin et al.

[SB] J. Stoer, R. Bulirsch, *Numerische Mathematik 2*, 4. Auflage 2000, Springer, Berlin et al.

[SB2] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, 2nd Edition 1993, Springer, Berlin et al.

[W] J. H. Wikinson, *The Algebraic Eigenvalue Problem*, 1965, Oxford University Press, Oxford

[T] E. Zeidler (Ed.) et al., *Teubner Taschenbuch der Mathematik*, 1996, Teubner, Stuttgart et al.

Index