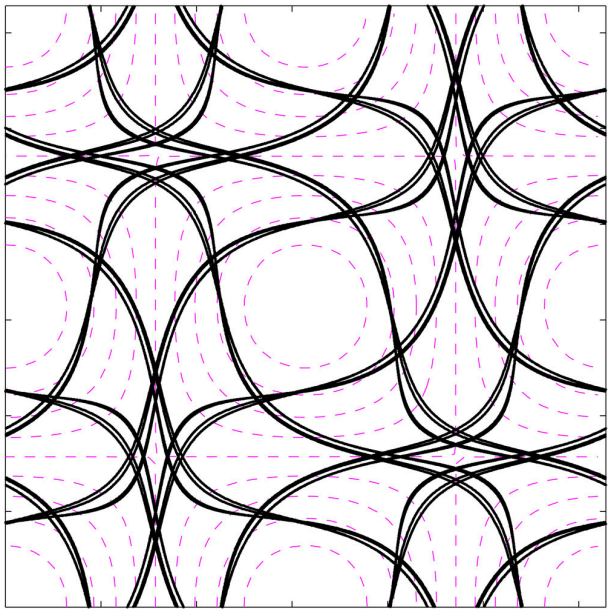
Skriptum zur Vorlesung
TU Wien, LVA-Nr. 322.036

NUMERISCHE METHODEN DER INGENIEURWISSENSCHAFTEN

Hendrik C. Kuhlmann



Hendrik C. Kuhlmann
Institut für Strömungsmechanik und Wärmeübertragung
Technische Universität Wien
Getreidemarkt 9
A-1060 Wien
Austria

Version: Sommersemester 2021

Das Frontispiz zeigt ein Beispiel von besonderen Trajektorien (schwarze Linien), welche punktförmige schwere Teilchen in einem Strömungsfeld aus schachbrettartig angeordneten Wirbeln (Taylor-Green-Strömung, magenta gestrichelt) für lange Zeiten annehmen können. Die Trajektorien besitzen hier eine höhere Periode als die Wirbelströmung. Der Ausschnitt zeigt eine räumliche Periode der Strömung. Beachte, daß die schweren Teilchen der Strömung nicht exakt folgen können.

Vorbemerkungen

Ziel des Kurses ist die Vermittlung grundlegenden Kenntnisse und Fähigkeiten zur numerischen Lösung einfacher Ingenieurprobleme. Die Veranstaltung ist auch eine nützliche Grundlage für die weiterführenden Kurse

- 302.017 VO Grundlagen der numerischen Methoden der Strömungs- und Wärmetechnik,
- 302.042 VO Numerische Methoden der Strömungsmechanik.

Die vorliegende erste Version, die im Laufe der Vorlesung des SS 2008 entstanden ist, wird sicher noch etliche Korrekturen und Verbesserungen erfahren. Dies bezieht sich auch auf eine bessere Motivation der einzelnen Problemklassen sowie auf die Hinzunahme weiterer Beispielrechnungen. Für Hinweise und Verbesserungsvorschläge wäre ich dankbar.

H. C. K.
im Juni 2008

In der zweiten Version wurden die algebraischen Grundlagen in einen Anhang verschoben. Es wird dem Leser geraten, sich diese Grundlagen in Erinnerung zu rufen. Desweiteren wurden kleine Passagen zu Beginn der ersten Kapitel eingefügt, welche die jeweilige Fragestellungen motivieren sollen. Außerdem wurde das Literaturverzeichnis aktualisiert, kleinere Fehler im Text korrigiert und der Index erstellt. Weitere laufenden Revisionen werden sich über das SS09 erstrecken. Geplant sind auch noch die weiteren Themen *Fouriertransformation* und, falls es die Zeit erlaubt, *Wavelets*.

H. C. K.
im März 2009

In der aktuellen Version wurden weitere Korrekturen berücksichtigt.

H. C. K.
im Juni 2010

Im SS 2012 wurden weitere Korrekturen des Skriptums vorgenommen und ein kurzes Kapitel über symplektische Integration eingefügt.

H. C. K.
im Juni 2012

Ab dem SS 2013 wird das Skriptum über das Grafische Zentrum der TU Wien vertrieben.

H. C. K.
im Dezember 2013

Literatur

Es gibt eine umfangreiche Literatur zu dem Thema. Die meiste Literatur ist naturgemäß relativ mathematisch ausgerichtet. Mir persönlich haben die beiden Bücher

- Golub and Ortega (1996): Golub, G. and Ortega, J. M. (1996), *Scientific Computing - Eine Einführung in das wissenschaftliche Rechnen und parallele Numerik*, Teubner, Stuttgart. (vergriffen)
- Trefethen and Bau, III (1997): Trefethen, L. N. and Bau, III, D. (1997), *Numerical linear algebra*, SIAM, Philadelphia. (€ 40,49)

am meisten zugesagt (Preise von ca. 2008). Die Vorlesung basiert über weite Strecken auf diesen beiden Quellen. Daneben scheinen mir auch die folgenden Bücher erwähnenswert, wobei die alphabetische Liste sicher unvollständig ist.

- Dahmen and Reusken (2006): Dahmen, W. and Reusken, A. (2006), *Numerik für Ingenieure und Naturwissenschaftler*, Springer, Berlin, Heidelberg. (€ 29,95)
- Freund et al. (2007): Freund, R. W., Hoppe, R. H. W., Stoer, J. and Burlisch, R. (2007), *Stoer/Bulirsch: Numerische Mathematik 1*, Springer, Berlin, Heidelberg. (€ 24,95)
- Golub and van Loan (1989): Golub, H. G. and van Loan, H. G. (1989), *Matrix Computations*, Johns Hopkins University Press. (*die Bibel*, € 34,95)
- Hanke-Bourgeois (2002): Hanke-Bourgeois, M. (2002), *Grundlagen der numerischen Mathematik und des wissenschaftlichen Rechnens*, Teubner, Wiesbaden. (€ 52,00)
- Huckle and Schneider (2006): Huckle, T. and Schneider, S. (2006), *Numerische Methoden*, Springer, Berlin, Heidelberg. (€ 21,00)
- Moler (2004): Moler, C. B. (2004), *Numerical Computing with MATLAB*, SIAM, Philadelphia.(€ 40,99)
- Press et al. (1992): Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992), *Numerical Receipes in C*, Cambridge University Press, Cambridge. (umfangreich, vergriffen)
- Quarteroni and Saleri (2006): Quarteroni, A. Saleri, F. (2006), *Wissenschaftliches Rechnen mit MATLAB*, Springer, Berlin, Heidelberg. (€ 29,95)
- Stoer and Burlisch (2005): Stoer Stoer, J., Burlisch, R. (2005), *Numerische Mathematik 2*, Springer, Berlin, Heidelberg. (€ 26,95)

Inhaltsverzeichnis

1. Vorbetrachtungen

1.1. Motivation und numerischer Ansatz

Computer werden bei der Lösung von wissenschaftlichen und ingenieurmäßigen Problemen in praktisch allen Bereichen standardmäßig eingesetzt. Als Beispiele seien nur die Berechnung von Satellitenbahnen genannt, die gesamte Luftfahrttechnik, die Auslegung von Brücken und Gebäuden oder die Simulation komplexer elektronischer Komponenten und deren thermisches Verhalten (Kühlung). Um solche Probleme lösen zu können, müssen zunächst *mathematische Modelle* entwickelt werden. Diese bestehen in vielen Fällen aus partiellen Differentialgleichungen. Für deren numerische Lösung existieren teils umfangreiche Software-Pakete, die für viele Situationen fertige Lösungsmethoden bereitstellen. Der Ingenieur muß *nur* noch als Operator tätig werden. Dabei kann es mehr oder weniger zeitaufwendig sein, sich in mächtige Softwarepakete einzuarbeiten. Die Einarbeitung in die Prozeduren an sich bereitet bei guter Software meist jedoch keine großen intellektuellen Schwierigkeiten. Schwieriger ist es dagegen, zu verstehen, wie die numerische Lösung gewonnen wird. Ein detailliertes Verständnis ist oft nicht mit vertretbarem Aufwand zu erreichen, da die Software sehr komplex sein kann und zwecks Optimierung nicht selten auf ausgeklügelte Algorithmen zurückgreift. Die meisten Verfahren basieren jedoch auf gewissen Grundlagen. Deren Verständnis ist wichtig, wenn man auf Probleme stößt, für die es keine fertigen Programme gibt und für die man die Lösung selbst programmieren muß. Grundlegende Kenntnisse über numerische Methoden sind auch dann wichtig, wenn die Qualität (Genauigkeit) eines numerischen Ergeb-

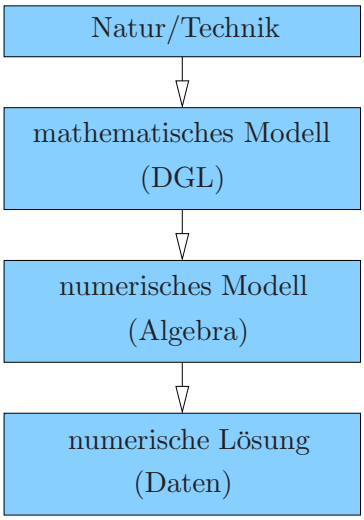


Abbildung 1.1.: Sequenz der Approximation natürlicher und technischer Vorgänge durch numerisch berechnete Daten. Bei jedem der drei Schritte entstehen Fehler.

nisses beurteilt werden soll. Denn die Abbildung der natürlichen bzw. technischen Vorgänge auf ein numerisches Resultat erfolgt in verschiedenen Schritten. In der Regel entstehen bei jeder Stufe der Approximation gewisse Fehler (Abb. 1.1).

Die Anwendung numerischer Methoden in den Natur- und Ingenieurwissenschaften, oft auch *wissenschaftliches Rechnen* genannt, basiert auf Techniken der *numerischen Analysis*. Sie entstand schon vor der Erfindung des Computers als Teilgebiet der Mathematik. Die praktische Umsetzung der Methoden der numerischen Analysis erfordert Hilfsmittel wie Computer, Betriebssysteme, Programmiersprachen, etc. Die beiden letztgenannten Gegenstände fallen in das Gebiet der Informatik und sind nicht (bzw. nur peripher) Gegenstand unserer Betrachtungen.

In den meisten Fällen werden die Probleme in Form von Differentialgleichungen formuliert, auf algebraische Probleme zurückgeführt und schließlich näherungsweise gelöst (Abb. 1.1). Das wichtigste Hilfsmittel der numerischen Analysis ist daher die Algebra. Darum ist es erforderlich, die Grundlagen der Algebra zu beherrschen. Zur Erinnerung sind im Anhang A die wichtigsten Regeln der Algebra zusammengestellt. Es wäre nützlich, diese mathematischen Grundlagen zu Beginn dieses Kurses zu wiederholen.

1.2. Aspekte der Modellierung

1.2.1. Modellbildung

Um ein naturwissenschaftliches oder ein technisches Problem zu lösen, benötigt man zunächst die Kenntnis aller *Einflußfaktoren*. Zusätzlich werden die *physikalischen Gesetze* benötigt (z.B. die Newtonschen Gesetze, die ihrerseits eine nichtrelativistische Näherung darstellen), welche die Einflußfaktoren mit einer Reaktion des Systems verbinden, z.B. mit der zeitlichen Entwicklung (Dynamik). Bei der mathematischen Formulierung des Problems können manche Einflußfaktoren eventuell vernachlässigt werden, wenn man keinen wesentlichen Einfluß dieser Größen auf das Ergebnis erwartet. Das Resultat einer derartigen Modellentwicklung besteht meist aus einer oder mehreren *partiellen Differentialgleichungen* sowie gewissen *Anfangs-* und *Randbedingungen*.

Beispiele: Wie reagiert eine dünne eingespannte Platte, wenn sie seitlich belastet wird (Abb. 1.2)? Zur Lösung des Problems benötigt man die Differentialgleichungen, welche die Deformationen mit den Kräften in Zusammenhang bringen. Außerdem spielen die Einspannbedingungen eine Rolle (Randbedingungen) sowie die anfänglich vorhandene Deformation (Anfangsbedingungen). Ein anderes Beispiel ist das Absinken einer schweren Kugel in einem viskosen Fluid (Abb. 1.2b).

1.2.2. Diskretisierung

Bei kontinuierlichen Problemen besteht das Modell in der Regel aus partiellen Differentialgleichungen. Dann sucht man im Prinzip die Lösung (zum Beispiel die

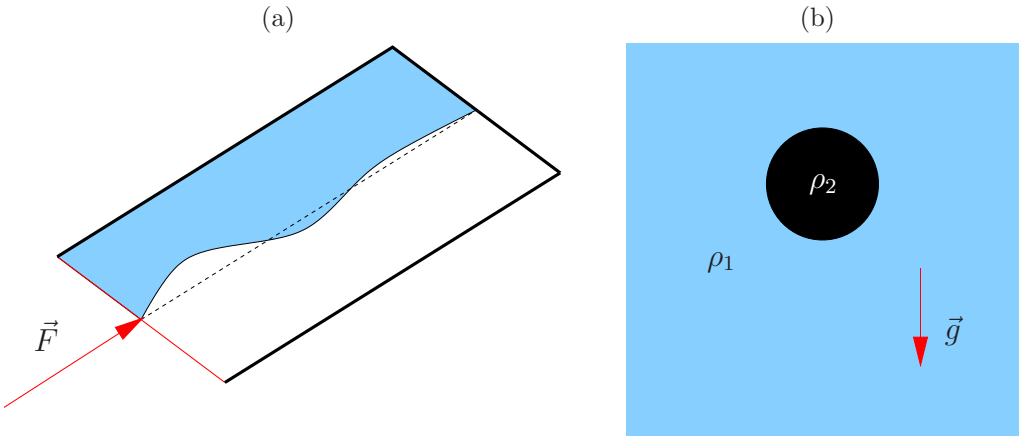


Abbildung 1.2.: (a) Belastung und Deformation einer dünnen Platte (nur eine Hälfte ist blau dargestellt), die an drei Seiten fest und horizontal eingespannt ist (dicke schwarze Umrandung) und entlang einer Seite homogen belastet wird (rote Linie). Die Deformation entlang der Mittellinie ist übertrieben dargestellt. (b) Eine Kugel, die in einem viskosen Fluid unter dem Einfluß der Schwerebeschleunigung absinkt ($\rho_2 > \rho_1$).

Auslenkung der Platte in Abb. 1.2a) an allen (unendliche vielen) Raumpunkten als Funktion der (unendlich vielen) Zeitpunkte.

In vielen Fällen läßt sich die exakte Lösung aber nicht in geschlossener Form angeben. Deshalb versucht man, die exakte Lösung mit numerischen Mitteln zu approximieren. Bei der *Diskretisierung* werden die unendlich vielen Freiheitsgrade des Systems (Auslenkung der ungestörten Platte an allen Punkten in der Ebene, bzw. Geschwindigkeitsvektor des Fluids an alle Raumpunkten) auf einige wenige zurückgeführt, womit das Problem endlich wird. Bei diesem Schritt entstehen *Diskretisierungsfehler* noch bevor mit der eigentlichen numerischen Rechnung begonnen wurde.

1.2.3. Numerische Lösung und mögliche Fehlerquellen

Das diskretisierte Problem mit nunmehr endlich vielen Unbekannten kann in einem nächsten Schritt numerisch gelöst werden. Da die Darstellung von Zahlen im Computer eine endliche Genauigkeit besitzt (es lassen sich nicht unendlich viele reelle Zahlen darstellen), entstehen bei diesem Schritt weitere Fehler. Dies sind die *Rundungsfehler*. Die Art der Fehler wird klar, wenn man sich verdeutlicht, daß noch nicht einmal die Zahl

$$\pi = 3.14159265358979323846264338327950288419716939937510582097494 4\ldots$$

exakt numerisch darstellbar ist.

Fehlerakkumulation

Auch wenn der Rundungsfehler bei einer einzelnen Operation (z.B. einer Multiplikation) klein ist, so kann es doch sein, daß sich die Fehler bei vielen Operationen (und das ist die Regel) nicht kompensieren sondern *akkumulieren*. Bei fehlerbehafteten Daten entstehen bei einer exakten Multiplikation Fehler erster und zweiter Ordnung

$$(x+\Delta x)(y+\Delta y) = xy + \underbrace{x\Delta y + y\Delta x}_{\text{1. Ordnung}} + \underbrace{\Delta x\Delta y}_{\text{2. Ordnung}} . \tag{1.1}$$

Wenn die beiden Fehlerterme erster Ordnung immer dasselbe Vorzeichen haben, akkumuliert der Fehler bei wiederholter Multiplikation und kann sogar größer werden als das Ergebnis.

Fatale Auslöschung

Ein ähnliches Problem ist die *fatale Auslöschung*. Wenn sich zwei Zahlen x und y nur um die letzte darstellbare Dezimale unterscheiden, dann steht für die Zahl $z = x - y$ nur eine Dezimalstelle zur Verfügung, was fatal ungenau ist. Bei der Subtraktion selbst entsteht kein Rundungsfehler. Aber alle Ergebnisse, die sich aus der Multiplikation mit z ableiten lassen sind dann nur auf einer Dezimalstelle genau. Sei zum Beispiel bei einer Genauigkeit von 8 Dezimalstellen

$$\begin{aligned} a \times z = a \times (x-y) &= \left(7.4992123\cdots \times 10^4\right) \times (1.8488409\cdots - 1.8488406\ldots) \\ &= \left(7.4992123\cdots \times 10^4\right) \times \left(3.\cdots \times 10^{-7}\right) = 2.24976369\cdots \times 10^{-2}. \end{aligned} \tag{1.2}$$

Hierbei ist nur die erste Dezimale des Ergebnisses richtig. Man wird also versuchen, durch geeignete *Algorithmen* (Rechenvorschriften), die fatale Auslöschung zu vermeiden.

Schlechte Kondition

Ein anderes Problem, das auftreten kann, ist eine extrem starke Abhängigkeit des Resultats von Fehlern in den Eingangsdaten. Man sagt dann, das Problem ist *schlecht konditioniert*. Durch geeignete Umformungen (Skalierungen), die *Vorkonditionierung* genannt werden, läßt sich dieses Problem in vielen Fällen vermeiden. Wir werden darauf noch in Kap. 2.1.4 zurückkommen.

Konvergenzfehler

Manchmal kann man das diskretisierte Problem nicht in einem einzigen Schritt numerisch lösen, sonder muß die Lösung durch wiederholtes Anwenden einer bestimmten Abfolge von Operationen *iterativ* gewinnen. Dabei wird davon ausgegangen, daß nach unendlich vielen Iterationen die exakte Lösung des diskreten Problems vorliegt. Da man die Iteration aber nach endlich vielen Schritten abbrechen muß, verbleibt ein gewisser *Abbruchfehler* oder *Konvergenzfehler*.

Effizienz

Bei umfangreichen Rechnungen spielt die *Effizienz* der Verfahren eine wichtige Rolle. Daher gilt es, mit möglichst wenigen Operationen das beste Ergebnis zu erzielen. Als Beispiel betrachten wir die Berechnung der Lösung eines linearen Gleichungssystems mit $N = 20$ Unbekannten

$$\mathsf{A} \cdot \vec{x} = \vec{b}. \tag{1.3}$$

Zur Lösung könnte man die *Cramersche Regel* verwenden, wonach die Komponenten x_n des Lösungsvektors durch

$$x_n = \frac{\det \mathsf{A}_n}{\det \mathsf{A}} \tag{1.4}$$

Gabriel Cramer*
1704–1752

gegeben sind, wobei die Matrix A_n dadurch entsteht, indem man die n-te Spalte von A durch den Vektor $\vec{b}$ ersetzt. Bei dieser Methode müssen $N + 1$ Determinanten berechnet werden. Die Determinante einer 20×20 Matrix A (was sehr klein ist) kann man mit Hilfe der *Leibniz-Formel* aus den Matrixelementen $a_{i,j}$ berechnen

$$\det \mathsf{A} = \sum_{\sigma \in S_N} \operatorname{sgn}(\sigma)\, a_{1,\sigma(1)} \ldots a_{N,\sigma(N)}, \tag{1.5}$$

wobei σ eine Permutation von $\{1, ..., N\}$ ist und S_N die Menge aller Permutationen von $\{1, ..., N\}$. Da es $N!$ verschiedene Permutationen von $\{1, ..., N\}$ gibt, sind zur Berechnung einer Determinante $\sim N!$ Operationen erforderlich. Mit $N = 20$ ist $N! > 10^{18}$. Der Intel Core i7 980 XE (ca. 2010) machte bestenfalls ca. 100 GFLOPS ($= 10^{11}$ Fließkomma-Operationen pro Sekunde). Damit ergibt sich eine gesamte Rechenzeit von $T > 100\,\mathrm{d}$.

Gottfried Wilhelm von Leibniz
1646–1716

Wenn man andererseits das *Gaußsche Eliminationsverfahren* (Kap. 2.1.1) verwendet, bei dem keine Determinante berechnet werden muß, kann man die Lösung $\vec{x}$ in ca. $N^3 = 8000$ Operationen erhalten, was mit dem Intel Core i7 980 XE in $\approx 10^{-7}\,\mathrm{s}$ erfolgen würde. Für sehr große Systeme ($N > 10^6$) ist dieser Aufwand aber immer noch zu hoch und man sucht Verfahren, die mit N^2 oder noch weniger Rechenoperationen skalieren. Dabei muß man allerdings die Struktur der Matrix ausnutzen. Je größer die Zahl der Unbekannten ist, desto wichtiger wird die Effizienz der Berechnung.

*Gabriel Cramer war ein Schweizer Mathematiker. Im Jahr 1750 veröffentlichte er das Buch *Introduction a l'analyse de lignes courbes algébriques*. In einem der Anhänge findet sich eine Formel zur Lösung linearer Gleichungssysteme, die später als Cramersche Regel bekannt wird. Sie gab den Anstoß zur Entwicklung der Determinantentheorie.

1.3. Programmierung

Beim Schreiben von Programmen sollte man von vorn herein einige Regeln beachten, die zwar zunächst etwas Mehraufwand bedeuten, auf längere Sicht aber viel Arbeit ersparen können. Die Regeln werden umso wichtiger, je umfangreicher die Programme werden. Hierzu zählen

- **Zuverlässigkeit** Das Programm sollte frei von Fehlern sein. Dazu muß man es ausgiebig testen. Dabei werden die numerischen Ergebnisse mit exakten oder anderen bekannten numerischen Daten (*Benchmarks*) für alle repräsentativen Parameter verglichen. Die wichtigsten Test sollten dokumentiert werden.
- **Robustheit** Das Programm sollte zuverlässig und robust sein. Das heißt, es sollte für alle relevanten Eingangsdaten hinreichend genaue Ergebnisse liefern. Falls gewisse Eingangsdaten zu singulärem Verhalten oder ungenauen Ergebnissen führen, sollte der Nutzer gewarnt werden.
- **Portierbarkeit** Das Programm sollte so geschrieben sein, daß die Eingangs- und Ausgangsdaten klar übergeben werden. Es sollte eine höhere Programmiersprache verwendet werden. Auf maschinenabhängige Tricks sollte verzichtet werden.
- **Wartungsfreundlichkeit** Das Programm sollte gut strukturiert und modular aufgebaut sein. Die Variablennamen sollten sinnvoll sein. Die einzelnen Teilschritte sollten im Programm selbst gut kommentiert sein, so daß ein anderer Nutzer sofort versteht, was in jedem Schritt gemacht wird. Bei umfangreichen Programmen mit vielen Ein- und Ausgabeoptionen ist eine separate Dokumentation wichtig.

Es gibt eine exzellente frei verfügbare Programm-Bibliothek, welche *Löser* für die wichtigsten elementaren Probleme zur Verfügung stellt. Sie heißt *LAPACK* (Linear Algebra PACKage), ist in FORTRAN geschrieben und wird von international renommierten angewandten Mathematikern gewartet und weiterentwickelt.[1] LAPACK greift auf die noch elementareren Routinen zurück, die von *BLAS* (Basic Linear Algebra Subprograms) zur Verfügung gestellt werden.

Eine sehr nützliche höhere Umgebung stellt *MATLAB* dar (MATrix LABoratory). Dies ist eine kommerzielle, plattformunabhängige Software, mit deren Hilfe mathematische Probleme komfortabel gelöst und graphisch ansprechend dargestellt

[1] Vorläufer von LAPACK waren LINPACK und EISPACK.

werden können. Das Besondere an MATLAB ist die matrix-orientierte Programmierung. Hierdurch kann die Kodierung in vielen Fällen sehr einfach erfolgen, da man nicht alle Indizes immer explizit anführen muß. Nicht-kommerzielle Alternativen zu MATLAB sind *Scilab* und *Octave*. In letzter Zeit ist *Python* immer beliebter geworden.

Ein weiteres nützliches Hilfsmittel ist das symbolische Rechnen. Softwarepakete MACSYMA, REDUCE, MAPLE und MATHEMATICA ermöglichen, gewisse *exakte* Operationen mit mathematischen Ausdrücken durchzuführen. Dazu zählt zum Beispiel das Vereinfachen von komplizierten Formelausdrücken, exaktes Integrieren oder exaktes Dividieren von rationalen Ausdrücken. Die Ergebnisse sind immer *exakt* im Gegensatz zum numerischen Rechnen, das prinzipiell *fehlerbehaftet* ist.

Wenn man ein Problem numerisch lösen will, gibt es zwei Aspekte zu beachten. Die Verifikation und die Validierung. Die *Verifikation* ist die Überprüfung des numerischen Codes hinsichtlich einer fehlerfreien Implementierung (s.o.). Zur Verifikation werden normalerweise bestimmte Testfälle (*benchmarks*) berechnet und die Ergebnisse mit denjenigen anderer Autoren verglichen. Handelt es sich um eine numerische Approximation (zum Beispiel einer Strömung), dann muß außerdem gezeigt werden, daß das numerische Ergebnis bei einer beliebigen Erhöhung der Genauigkeit (Gitterverfeinerung) gegen den korrekten Grenzwert konvergiert. Die *Validierung* bezieht sich auf ein physikalisches Modell. Zur Validierung eines Modells (Approximation der Realität) muß gezeigt werden, daß die Ergebnisse der Modellrechnung für alle möglichen Bedingungen hinreichend genau der Realität entsprechen. Dies macht in der Regel einen Vergleich mit Experimenten erforderlich (siehe dazu auch Roache, 2009).

1.4. Fließkomma-Arithmetik

Zahlen werden im Computer binär dargestellt. Dies ist durch die elektronische Realisation bedingt (leitender/nicht leitender Zustand). Die elementare Einheit ist ein *Bit*. Es kann die beiden Werte 0 und 1 annehmen. Mit 8 Bit kann man daher $2^8 = 256$ Werte (0 bis 255) darstellen. 8 Bit werden zu 1 *Byte* zusammengefaßt.[2]

Eine Untermenge der ganzen Zahlen $\mathbb{Z}$ kann exakt dargestellt werden. Die Beschränkung wird durch die Anzahl der zur Darstellung zur Verfügung stehenden Bits diktiert. Bei `integer*4`[3] stehen 4 Byte $= 4 \times 8$ Bit $= 32$ Bit zur Verfügung. Damit kann man $2^{32} = 4\,294\,967\,296$ Werte darstellen, also den Bereich von 0 bis 4 294 967 295. Wenn man ein Bit für das Vorzeichen verwendet, lassen sich mit 4 Byte die ganzen Zahlen aus dem Intervall $[-2\,147\,483\,647, 2\,147\,483\,648]$ darstellen. Dies sind 10 Dezimalstellen.

Bei der Darstellung der reellen und der komplexen Zahlen $\mathbb{R}$ bzw. $\mathbb{C}$ ist neben der Größenbeschränkung auch zu beachten, daß zwischen den darstellbaren Zahlen

[2] 2 Byte stellen ein *Wort* (16 Bit) dar und 4 Byte stellen ein *Doppelwort* (32 Byte) dar.
[3] Die 4 bezeichnet hier die zur Verfügung stehenden Bytes.

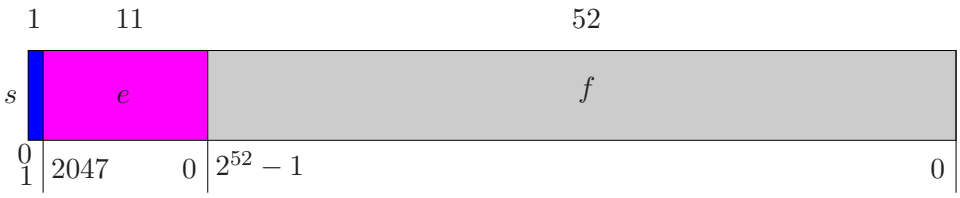


Abbildung 1.3.: Aufteilung der 8 Bytes einer `real*8` Fließkommazahl in ein Vorzeichen-Bit s, 11 Exponent-Bits e, und 52 Mantissen-Bits f. Die untere Zeile gibt den Wertebereich an, den s, e und f annehmen können.

Lücken entstehen. Die Größenbeschränkung ist meist nicht das Problem — außer eventuell bei der Berechnung von Determinanten.

Bei den *Fließkomma-Zahlen* $\mathbb{F} \subset \mathbb{R}$ wird die Position des Kommas separat abgespeichert und man hat eine gewisse Anzahl von Dezimalen für den Wert zur Verfügung. Daher skaliert der Abstand der Zahlen auch mit ihrer Größe.[4] Für *einfache* und *doppelte Genauigkeit* werden für reelle Zahlen 4 bzw. 8 Bytes verwendet. Dies sind $4 \times 8 = 32$ bzw. $8 \times 8 = 64$ Bits. Wenn man eine exponentielle Darstellung verwendet und für das Vorzeichen und den Betrag des Exponenten 8 bzw. 12 Bits verwendet, so verbleiben zur Darstellung der Mantisse 24 Bits (einfache Genauigkeit) bzw. 52 Bits (doppelte Genauigkeit).

Einfache Genauigkeit ist oft nicht ausreichend. Daher betrachten wir eine Fließkommazahl mit doppelter Genauigkeit `real*8`, wofür 64 Bit zur Verfügung stehen. Dies sind $2^{64} = 18\,446\,744\,073\,709\,551\,616$ Werte. Nach dem *IEEE-Standard 754*[5] wird eine reelle Zahl in der Form

$$x = (-1)^s 2^{(e-1023)} \underbrace{\left(1 + \frac{f}{2^{52}}\right)}_{m=1+f'} \tag{1.6}$$

dargestellt, wobei von den 64 Bit 1 Bit für das Vorzeichen ($s = 0, 1$), 11 Bits für den Exponenten e (Basis 2) und 52 Bits für die Mantisse m verwendet werden, die meistens normiert als $m = 1 + f'$ dargestellt wird. Hierbei regelt $0 \le f' = f/2^{52} < 1$ die Nachkommastellen (Abb. 1.3). Die reellen Zahlen aus dem Intervall $[1, 2)$ werden nach dem IEEE-Standard 754 dargestellt als

$$m = 1 + f' = \underbrace{1 + 0, 1 + 2^{-52}, 1 + 2 \times 2^{-52}, 1 + 3 \times 2^{-52}, ..., 1 + (2^{52} - 1)2^{-52}}_{2^{52} \text{ Werte}}, \tag{1.7}$$

diejenigen aus dem Intervall $[2, 4)$ wären dann

$$m \times 2^1 = \underbrace{2 + 0, 2 + 2^{-51}, 2 + 2 \times 2^{-51}, 2 + 3 \times 2^{-51}, ..., 2 + (2^{52} - 1)2^{-51}}_{2^{52} \text{ Werte}}, \tag{1.8}$$

[4] Bei sogenannten Festkommazahlen ist der Abstand zwischen zwei Zahlen immer gleich groß.
[5] Siehe http://754r.ucbtest.org/standards/754.pdf. Der Link hat sich zwischenzeitlich geändert.

und so weiter. Der *relative* Abstand benachbarter Zahlen ist also nie größer als $2^{-52} \approx 2.22 \times 10^{-16}$. In Matlab ist dies identisch mit der relativen Genauigkeit `eps`.[6] Die Zahlen $\mathbb{F}$ scheinen also relativ dicht zu liegen.[7]

Für den Exponenten e stehen im Prinzip $2^{11} = 2048$ Werte zur Verfügung. Die Werte $e = 0$ und $e = 2047$ werden aber zu Darstellung von 0, NaN und INFINITY folgendermaßen verwendet:[8]

1. Falls $e = 2047$ und $f \neq 0$, dann ist $x = \text{NaN}$, unabhängig von s,
2. Falls $e = 2047$ und $f = 0$, dann ist wie in (1.7) $x = (-1)^s \,\text{INFINITY}$,
3. Falls $0 < e < 2047$, dann ist $x = (-1)^s \, 2^{e-1023} \, (1 + f')$,
4. Falls $e = 0$ und $f \neq 0$, dann ist $x = (-1)^s \, 2^{e-1022} \, (0 + f')$ (denormalisierte Zahlen),
5. Falls $e = 0$ und $f = 0$, dann ist $x = (-1)^s \, 0$ (Null mit Vorzeichen).

Damit verbleibt ein ganzzahliger Bereich $e \in [1, 2046]$, womit für den Exponenten $e - 1023$ der Bereich $[-1022, 1023]$ zur Verfügung steht. Folglich ergibt sich der Betrag der größten darstellbaren Zahl zu $[1 + (1 - 2^{-52})] \times 2^{1023} \approx 1.798 \times 10^{308}$. Die vom Betrage her kleinste darstellbare Zahl ist $(1+0) \times 2^{-1022} \approx 2.225 \times 10^{-308}$.

Wenn man die exakten Rechenoperationen $+, -, \times$ und $/$ allgemein mit $*$ und deren diskrete Versionen mit $\circledast$ bezeichnet, dann sollte im Idealfall immer gelten

$$x \circledast y = (x * y)(1 + \epsilon), \tag{1.9}$$

mit einem relativen Fehler $\epsilon \leq \epsilon_{\text{maschine}}$. Für die meisten heutigen Computer ist dies der Fall. Die *Maschinengenauigkeit* sollte dann im Bereich $\epsilon_{\text{maschine}} \approx 2^{-24}$ (einfache Genauigkeit, 32 Bit) bzw. 2^{-52} (doppelte Genauigkeit, 64 Bit) liegen. Dies sind $\approx 6 \times 10^{-8}$ bzw. $\approx 10^{-16}$.[9] Dieselbe Relation sollte auch für Operationen mit komplexen Zahlen gelten, wobei jedoch $\epsilon_{\text{maschine}}$ für die Multiplikation und die Division um einen Faktor von $2^{3/2}$ bzw. $2^{5/2}$ zu vergrößern sind.

Durch die Rundung bei Rechenoperationen geht im allgemeinen auch die Assoziativität der Addition verloren, d.h.

$$(a \oplus b) \oplus c \neq (b \oplus c) \oplus a. \tag{1.10}$$

Das Ergebnis hängt davon ab, welche Zahlen zuerst miteinander verknüpft werden. Ähnliches gilt auch für das Distributivgesetz.

[6] Matlab rechnet intern mit `real*8`, zeigt die Zahlen aber normalerweise mit nur 4 Nachkommastellen dar. Wenn man eine genauere Anzeige möchte, kann man dies mit `format long` oder `format long e` erreichen.

[7] Bei einigen Algorithmen kann dies trotzdem problematisch sein, so daß man zu einer noch höherer Genauigkeit übergehen muß.

[8] Für die Details, siehe http://754r.ucbtest.org/standards/754.pdf.

[9] Der Intel Pentium von 1994 lieferte für doppelte Genauigkeit zunächst nur ein $\epsilon_{\text{maschine}} \approx 6.1 \times 10^{-6}$. Dies ist 11 Größenordnungen zu groß. Der Fehler konnte aber durch Korrektur einer Datentabelle behoben werden.

2. Lösung linearer Gleichungssysteme

Lineare Gleichungssysteme tauchen in sehr vielen Problemen auf. Ein typisches Beispiel ist die Lösung eines linearen Randwertproblems (Kap. 5.3). Die Matrizen werden insbesondere bei dreidimensionalen Randwertproblemen und bei einer hohen Gitterauflösung sehr groß. Nichtlineare Probleme werden meistens iterativ gelöst. Auch hierbei ist in jedem Iterationsschritt ein lineares Problem zu lösen (siehe z.B. Kap. 3.5.6).

Lineare Gleichungssysteme mit geringer Dimension kann man mit einem *direkten Löser* behandeln. Ist die Anzahl der Unbekannten aber sehr groß, so wird man eine numerische Näherung mit Hilfe eines schnellen iterativen Verfahrens anstreben.

2.1. Direkte Verfahren für lineare Systeme

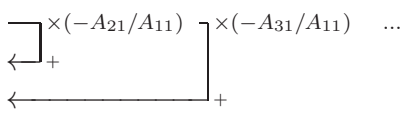
2.1.1. Gauß-Verfahren

Das *Gauß-Verfahren* ist ein universelles und direktes Verfahren zur Lösung linearer Gleichungssysteme

$$\mathsf{A} \cdot \vec{x} = \vec{b} \tag{2.11}$$

mit *voll besetzter Matrix* A. Das Gauß-Verfahren beruht auf der Idee, das Problem sukzessive auf kleinere Systeme zurückzuführen. Dazu betrachten wir die Matrix

$$\mathsf{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & . & . & . & A_{1N} \\ A_{21} & A_{22} & A_{23} & . & . & . & A_{2N} \\ . & . & . & & & & . \\ . & . & . & . & & & . \\ . & . & . & & . & & . \\ . & . & . & & & . & . \\ A_{N1} & A_{N2} & A_{N3} & . & . & . & A_{NN} \end{pmatrix} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}$$



. (2.12)

Die der Matrix-Gleichung (2.11) zugrundeliegenden Gleichungen für die Komponenten des Lösungsvektors können beliebig vertauscht, mit Faktoren multipliziert oder voneinander subtrahiert werden. Daher können wir die Matrix durch Zeilenoperationen in eine andere Matrix überführen, ohne daß sich die Lösung $\vec{x}$ ändert. Dieselben Operationen müssen natürlich auch auf den Vektor $\vec{b}$ angewandt werden.

Wie in (2.12) angedeutet, überführen wir A in eine Matrix, in der alle Elemente in der Spalte unterhalb von A_{11} gleich Null sind. Damit ist das System mit

N Unbekannten in ein System der Dimension $N-1$ überführt worden, denn die Gleichungen entsprechend den Zeilen $n = 2, \ldots, N$ enthalten nun nicht mehr die Unbekannte x_1. Dieses Verfahren wird nun weitergeführt, wobei im zweiten Schritt mit Hilfe der zweiten Zeile die Matrix-Elemente $A_{32}, ..., A_{N2}$ auf Null gebracht werden. Wenn man dies fortsetzt, hat man am Ende ein Problem, in dem nur die *obere Dreiecksmatrix* besetzt ist:

$$\mathsf{A} = \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ & A_{22} & \ldots & A_{2N} \\ & & \ddots & \vdots \\ & & & A_{NN} \end{pmatrix} =: \mathsf{U}. \tag{2.13}$$

Hierbei wurden für die obere Dreiecksmatrix U dieselben Symbole A_{ij} verwendet, obwohl nur die erste Zeile von A und die Komponente b_1 von $\vec{b}$ gleichgeblieben sind. Alle anderen Elemente wurden modifiziert. Da man die ursprünglichen Elemente aber nicht mehr benötigt, können die entsprechenden Speicherplätze mit den neuen Einträgen überschrieben werden. Dieser Teil des Algorithmus wird *forward elimination* genannt.

Nun können wir das Problem leicht lösen, denn mit (2.13) ergibt sich aus der letzten Gleichung von (2.11)

$$x_N = \frac{b_N}{A_{NN}}. \tag{2.14}$$

Dieses Resultat kann in Gleichung $N-1$ eingesetzt werden, um die Gleichung $N-1$ nach x_{N-1} aufzulösen. Wenn man dies sukzessive von $n = N$ rückwärts bis $n = 1$ weiterführt, erhalten wir aus der n-ten Gleichung (n-te Zeile von U)

$$A_{nn}x_n + \underbrace{\sum_{k=n+1}^{N} A_{nk}x_k}_{\text{bekannt}} = b_n, \tag{2.15}$$

mit der Lösung

$$x_n = \frac{b_n - \sum_{k=n+1}^{N} A_{nk}x_k}{A_{nn}}. \tag{2.16}$$

Alle Koeffizienten A_{nk} und b_n sind aus der *forward elimination* bekannt und die Komponenten des Lösungsvektors x_k mit $k > n$ wurden in den vorherigen Schritten berechnet. Dieser Teil des Algorithmus wird *back substitution* genannt.

Die *forward elimination* kostet $\sim N^3/3$ Operationen[1] und ist damit wesentlich aufwendiger als die *back substitution* mit $\sim N^2/2$ Operationen. Die Gauß-

[1] Die Anzahl der Operationen für große N ist $\propto \sum_{k=1}^{N} k^2 \approx \int_0^N k^2\,\mathrm{d}k = N^3/3$, weil in jedem Eliminationsschritt Untermatrizen der Größe $k \times k$ neu berechnet werden müssen. Für die *back substitution* benötigt man dagegen nur $\propto \sum_{k=1}^{N} k \approx \int_0^N k\,\mathrm{d}k = N^2/2$ Operationen.

Elimination ist also sehr rechenintensiv, aber für vollbesetzte Matrizen gibt es praktisch kein anderes Verfahren, das wesentlich besser wäre.[2]

Matlab: Algorithmus und Skalierung der Vorwärts-Elimination

Bei der Diskussion des Gauß-Algorithmus haben wir angenommen, daß die Diagonalelemente von Null verschieden sind. Falls dies einmal nicht der Fall sein sollte, kann man die betreffende Zeile n mit einer anderen Zeile $m > n$ vertauschen. Wenn die Matrix regulär ist, gibt es dort immer ein von Null verschiedenes Matrixelement. Die Zeilenvertauschung entspricht nur einer Vertauschung der Gleichungen.

Wenn die Gauß-Elimination auf große voll-besetzte Matrizen angewandt wird, kann es zu einer Akkumulation von Fehlern kommen (Golub and van Loan, 1989). Um dies zu vermeiden, sollten die Diagonalelemente (*Pivot-Elemente*) betragsmäßig so groß wie möglich sein.[3] Denn dann sind die Multiplikatoren im Gauß-Verfahren (siehe (2.12)) möglichst klein und Rundungsfehler werden minimiert. Um die Pivot-Elemente betragsmäßig zu maximieren, werden vor der Eliminierung einer Spalte n die Zeilen mit $m \geq n$ unterhalb des aktuellen Diagonalelements so vertauscht, daß auf der Diagonale das betragsmäßig größtmögliche Element steht. Diese Strategie nennt man *Teilpivotisierung*. Bei der *Totalpivotisierung* werden Zeilen *und* Spalten der jeweilige Untermatrix vertauscht, so daß das betragsmäßig größte Element der Untermatrix auf die Diagonalposition kommt. Weil die vollständige Pivotisierung mit einem großen Mehraufwand verbunden ist, wird meist darauf verzichtet.[4]

Die Gauß-Elimination läßt sich nicht gut vektorisieren oder parallelisieren. Sie wird daher relativ selten bei Problemen mit sehr vielen Unbekannten (z.B. CFD-Problemen) verwendet.

2.1.2. LU-Zerlegung

Eine wichtige Variante der Gauß-Elimination ist die *LU-Zerlegung*. Hierbei wird $\mathsf{A} = \mathsf{L} \cdot \mathsf{U}$ in ein Produkt aus einer unteren (L) und einer oberen Dreiecksmatrix (U) zerlegt. Diese Zerlegung ist vorteilhaft, wenn man mehrere Lösungen von (2.11) für verschiedene rechte Seiten $\vec{b}$ berechnen muß.

Den ersten Schritt der Gauß-Elimination, bei welcher alle Elemente unter dem Element A_{11} zu Null gemacht werden, kann man auch als Matrix-Multiplikation

[2] Die Verwendung der Cramerschen Regel (1.4) ist ausgeschlossen, da hierzu $O(N!)$ Operationen nötig wären. Die erforderliche aufwendige Berechnung von Determinanten sollte also tunlichst vermieden werden.

[3] Ansonsten können fatale Fehler auftreten, zum Beispiel wenn eine kleine Zahl von einer sehr großen Zahl abgezogen wird, deren absolute Ungenauigkeit größer ist als der kleine Subtrahend. Dann ändert sich die große Zahl nicht, was zu singulärem Verhalten führen kann.

[4] Bei diagonaldominanten oder symmetrischen positiv definiten Matrizen ist keine Teilpivotisierung erforderlich. Trotzdem kann die Genauigkeit bei positiv definiten Matrizen durch Teilpivotisierung etwas verbessert werden.

$\mathsf{L}_1 \cdot \mathsf{A}$ schreiben, wobei

$$\mathsf{L}_1 = \underbrace{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}}_{\mathsf{I}} - \underbrace{\begin{pmatrix} 0 & & & \\ l_{21} & 0 & & \\ \vdots & & \ddots & \\ l_{N1} & & & 0 \end{pmatrix}}_{\mathsf{C}_1} = \begin{pmatrix} 1 & & & \\ -l_{21} & 1 & & \\ \vdots & & \ddots & \\ -l_{N1} & & & 1 \end{pmatrix} \tag{2.17}$$

ist, mit $l_{k1} = A_{k1}/A_{11}$, $k = 2, \ldots, N$. Da in der Matrix C_1 nur die erste Spalte von Null verschieden ist, gehen in das Produkt $\mathsf{C}_1 \cdot \mathsf{A}$ nur die Elemente der ersten Zeile von A ein.[5] Im n-ten Zwischenschritt kann man die Elimination als Matrix-Multiplikation mit

$$\mathsf{L}_n = \mathsf{I} - \mathsf{C}_n = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{n+1,n} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{N,n} & & 1 \end{pmatrix} \tag{2.18}$$

auffassen, wobei

$$l_{k,n} = \frac{A_{kn}}{A_{nn}}, \qquad k = n+1, \ldots, N. \tag{2.19}$$

Alle Matrizen L_n sind untere Dreiecksmatrizen. Die Gauß-Elimination, welche auf eine obere Dreiecksmatrix führt, läßt sich also als eine Sequenz von Multiplikationen mit unteren Dreiecksmatrizen darstellen

$$\underbrace{\mathsf{L}_{N-1} \cdot \ldots \cdot \mathsf{L}_2 \cdot \mathsf{L}_1}_{\hat{\mathsf{L}}} \cdot \mathsf{A} = \mathsf{U}. \tag{2.20}$$

Damit haben wir U. Wenn man $\hat{\mathsf{L}}$ invertieren könnte, so daß $\mathsf{A} = \hat{\mathsf{L}}^{-1} \cdot \mathsf{U}$, und wenn diese Inverse eine untere Dreicksmatrix wäre, dann hätte man die LU-Zerlegung erreicht. Dazu beachten wir, daß alle Matrizen L_n regulär sind, da die Determinanten gleich 1 sind. Demnach ist ihr Produkt $\hat{\mathsf{L}}$ auch regulär. Es ist leicht zu sehen, daß ein Produkt zweier unterer Dreiecksmatrizen wieder eine untere Dreiecksmatrix ist. Außerdem kann man L_n leicht invertieren. Die Inverse erhält man einfach durch Vorzeichenwechsel vor den Elementen $l_{k,n}$, denn

$$\mathsf{L}_n^{-1} \cdot \mathsf{L}_n = (\mathsf{I} + \mathsf{C}_n) \cdot (\mathsf{I} - \mathsf{C}_n) = \mathsf{I} + \mathsf{C}_n - \mathsf{C}_n - \underbrace{\mathsf{C}_n \cdot \mathsf{C}_n}_{=0} = \mathsf{I}. \tag{2.21}$$

Damit kann man die Inverse von $\hat{\mathsf{L}}$ sofort angeben, denn mit

$$\underbrace{\mathsf{L}_1^{-1} \cdot \mathsf{L}_2^{-1} \cdot \ldots \cdot \mathsf{L}_{N-1}^{-1}}_{\hat{\mathsf{L}}^{-1}} \cdot \underbrace{\mathsf{L}_{N-1} \cdot \ldots \cdot \mathsf{L}_2 \cdot \mathsf{L}_1}_{\hat{\mathsf{L}}} = \mathsf{I} \tag{2.22}$$

[5] Mit $\mathsf{C}^{(1)} \cdot \mathsf{A} = \sum_j C_{ij}^{(1)} A_{jk} = C_{i1}^{(1)} A_{1k}$ trägt nur $j = 1$ zum Ergebnis bei.

erhalten wir die gesucht untere Dreiecksmatrix der LU-Zerlegung

$$\mathsf{L} := \hat{\mathsf{L}}^{-1} = \mathsf{L}_1^{-1} \cdot \ldots \cdot \mathsf{L}_{N-1}^{-1}. \tag{2.23}$$

Die Inverse $\hat{\mathsf{L}}^{-1}$ ist also auch wieder eine untere Dreiecksmatrix.[6] Damit kann man (2.20) nach A auflösen

$$\mathsf{A} = \hat{\mathsf{L}}^{-1} \cdot \mathsf{U} = \mathsf{L} \cdot \mathsf{U}. \tag{2.24}$$

Da alle Dreicksmatrizen L_n Einsen auf der Diagonale haben, hat auch $\hat{\mathsf{L}}$ Einsen auf der Diagonale.

Man kann nun zeigen (siehe unten), daß[7]

$$\mathsf{L} = \mathsf{L}_1^{-1} \cdot \mathsf{L}_2^{-1} \cdot \ldots \cdot \mathsf{L}_{N-1}^{-1} = \begin{pmatrix} 1 & & & & \\ \frac{A_{21}}{A_{11}} & 1 & & & \\ \frac{A_{31}}{A_{11}} & \frac{A_{32}}{A_{22}} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \frac{A_{N1}}{A_{11}} & \frac{A_{N2}}{A_{22}} & \cdots & \frac{A_{N,N-1}}{A_{N-1,N-1}} & 1 \end{pmatrix}. \tag{2.25}$$

In die Matrix L gehen also gerade jene Faktoren ein, die bei der Gauß-Elimination berechnet werden müssen. Damit kostet die LU-Zerlegung genausoviel wie die Gauß-Elimination. Die Gauß-Elimination liefert U und ganz nebenbei L ohne zusätzlichen Aufwand. Außerdem kann man für L denselben Speicherplatz wie für die untere Dreieckshälfte von A verwenden, denn die Zwischenergebnisse müssen nicht explizit gespeichert werden.

Um (2.25) zu zeigen, definieren wir den n-ten Spaltenvektor

$$\vec{l}_n := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ l_{n+1,n} \\ \vdots \\ l_{N,n} \end{pmatrix}. \tag{2.26}$$

Sei $\vec{e}_n$ der Einheitsvektor, der eine 1 in der n-ten Zeile besitzt und 0 sonst. Dann ist $\mathsf{L}_n = \mathsf{I} - \vec{l}_n \vec{e}_n = \mathsf{I} - \mathsf{C}_n$.[8] Damit erhalten wir für das Produkt

$$\mathsf{L}_n^{-1} \cdot \mathsf{L}_{n+1}^{-1} = \left(\mathsf{I} + \vec{l}_n \vec{e}_n\right) \cdot \left(\mathsf{I} + \vec{l}_{n+1} \vec{e}_{n+1}\right) = \mathsf{I} + \vec{l}_n \vec{e}_n + \vec{l}_{n+1} \vec{e}_{n+1}, \tag{2.27}$$

[6] Ganz allgemein kann man zeigen, daß die Inverse einer unteren (oberen) Dreiecksmatrix auch wieder eine untere (obere) Dreiecksmatrix ist.

[7] Beachte, daß nur die Koeffizienten A_{n1} aus der ursprünglichen Matrix stammen. Die Elemente A_{n2}, ... werden im Laufe der Gauß-Elimination wiederholt überschrieben.

[8] Mit $\vec{l}_n \vec{e}_n$ ist das dyadische Produkt gemeint. Dann ist die Reihenfolge der Faktoren wichtig und der zweite Faktor ist eigentlich transponiert zu nehmen. Also $\left[\vec{l}_n \vec{e}_n\right]_{ij} = \left[\vec{l}_n\right]_i [\vec{e}_n]_j$.

denn es ist $\vec{e}_n \cdot \vec{l}_{n+1} = 0$. Damit ist

$$\mathsf{L} = \mathsf{L}_1^{-1} \cdot \ldots \cdot \mathsf{L}_{N-1}^{-1} = \mathsf{I} + \vec{l}_1\vec{e}_1 + \ldots + \vec{l}_{N-1}\vec{e}_{N-1} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{N1} & l_{N2} & \ldots & l_{N,N-1} & 1 \end{pmatrix}. \quad (2.28)$$

Mit der LU-Zerlegung (2.24) kann man das lineare Problem $\mathsf{A} \cdot \vec{x} = \mathsf{L} \cdot \underbrace{\mathsf{U} \cdot \vec{x}} = \vec{b}$ in zwei Stufen lösen, wenn man $\mathsf{U} \cdot \vec{x} =: \vec{y}$ definiert, also

$$\mathsf{L} \cdot \vec{y} = \vec{b}, \quad (2.29a)$$
$$\mathsf{U} \cdot \vec{x} = \vec{y}. \quad (2.29b)$$

Die beiden Gleichungen können nun sukzessive mittels *back substitution* effizient gelöst werden, zuerst (2.29a) und dann (2.29b). Wenn man Lösungen für verschiedene Vektoren $\vec{b}$ auf der rechten Seite sucht, ist die LU-Zerlegung vorteilhaft. Denn die LU-Zerlegung muß nur einmal durchgeführt werden, da die Kenntnis von $\vec{b}$ dazu nicht erforderlich ist. Dann erhält man die verschiedenen Lösungen jeweils mit nur $O(N^2)$ Operationen.

Nicht jede Matrix A kann LU-zerlegt werden. Wenn aber die Determinanten aller Untermatrizen $\det|\mathsf{A}(1:n, 1:n)| \neq 0$ für alle $n \in [1, N-1]$ von Null verschieden sind, dann ist A LU-zerlegbar (hinreichende Bedingung). Wenn A außerdem regulär ist, d.h. $\det|\mathsf{A}| \neq 0$, dann ist die LU-Zerlegung eindeutig (Golub and van Loan, 1989).

In MATLAB kann man mit `[L,U] = lu(A)` die Matrizen `L` und `U` erhalten, welche die LU-Zerlegung von `A` darstellen. Dann kann man das Gleichungssystem $\mathsf{A} \cdot \vec{x} = \vec{b}$ lösen, indem man es mit Hilfe des MATLAB-Befehls `\` nach $\vec{x}$ auflöst: `x = U\(L\b)`.

2.1.3. Tridiagonale Systeme

Eine besonders effiziente Lösung existiert für *tridiagonale Systeme*. Diese ergeben sich zum Beispiel bei der Diskretisierung einer gewöhnlichen Differentialgleichung zweiter Ordnung (eindimensionales Problem) mittels zentraler finiten Differenzen. So hat (5.389) in Kap. 5.3.1 die Form

$$\begin{pmatrix} b_1 & c_1 & & & & & & \\ a_2 & b_2 & c_2 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & a_{n-1} & b_{n-1} & c_{n-1} & & & \\ & & & a_n & b_n & c_n & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & & & a_N & b_N \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \cdot \vec{x} = \vec{d}. \quad (2.30)$$

Dirichlet-Randbedingungen der DGL gehen dabei in die Elemente d_1 und d_N ein. In der Matrix sind nur die Hauptdiagonale und die beiden ersten Nebendiagonalen besetzt. Die untere Nebendiagonale a_n kann man mit dem Gauß-Schema eliminieren. Wenn der Algorithmus schon bis zur Zeile $n-1$ durchgeführt wurde, d.h. wenn $a_2 = \ldots = a_{n-1} = 0$ schon eliminiert wurden, dann wird in der nächsten Zeile a_n durch

$$\text{Zeile}(n) \quad \longrightarrow \quad \text{Zeile}(n) \; - \; \frac{a_n}{b_{n-1}} \times \text{Zeile}(n-1) \tag{2.31}$$

eliminiert. Die neuen Matrix-Elemente der Zeile n von A und das neue Element d_n ergeben sich dann zu

$$a_n \quad \longrightarrow \quad a_n - \frac{a_n}{b_{n-1}} b_{n-1} = 0, \tag{2.32}$$

$$b_n \quad \longrightarrow \quad b_n - \frac{a_n}{b_{n-1}} c_{n-1}, \tag{2.33}$$

$$c_n \quad \longrightarrow \quad c_n - \frac{a_n}{b_{n-1}} \times 0 = c_n, \tag{2.34}$$

$$d_n \quad \longrightarrow \quad d_n - \frac{a_n}{b_{n-1}} d_{n-1}. \tag{2.35}$$

Dieser sogenannte *forward sweep* führt auf die einfache obere Dreiecksmatrix[9]

$$\begin{pmatrix} b_1 & c_1 & & & & & & \\ & b_2 & c_2 & & & & & \\ & & \ldots & & & & & \\ & & & b_{n-1} & c_{n-1} & & & \\ & & & & b_n & c_n & & \\ & & & & & \ldots & & \\ & & & & & & b_{N-1} & c_{N-1} \\ & & & & & & & b_N \end{pmatrix} \cdot \vec{x} = \vec{d}, \tag{2.36}$$

Llewellyn Hilleth Thomas
1903–1992

wodurch das Gleichungssystem mit *back substitution* gelöst werden kann. Wenn $x_N, x_{N-1}, ..., x_{n+1}$ schon berechnet wurden, dann ergibt sich x_n aus

$$b_n x_n + c_n x_{n+1} = d_n \quad \Rightarrow \quad x_n = \frac{d_n - c_n x_{n+1}}{b_n}. \tag{2.37}$$

Dieses Verfahren wird auch *Thomas-Algorithmus* oder TDMA (***T**ri-**D**iagonal **M**atrix **A**lgorithm*) genannt. Der Algorithmus ist besonders ökonomisch, denn die erforderlichen Operationen skalieren linear mit der Anzahl N der Unbekannten. Effizienter geht es praktisch nicht. Wenn immer möglich wird daher dieser Algorithmus bzw. Varianten davon benutzt. Der Thomas-Algorithmus kann leicht auf Systeme mit mehreren Diagonalen

[9] Beachte, daß die modifizierten Werte hier dieselbe Bezeichnung haben.

erweitert werden. Außerdem kann man damit Systeme lösen, deren Matrizen eine *Block-Tridiagonalstruktur* besitzen. Dann sind alle obigen Operationen vektoriell zu verstehen.

Im dem Spezialfall, daß alle Elemente einer Diagonale denselben Wert besitzen, kann man das System noch effizienter lösen. Dies kann man mit *zyklischer Reduktion* erreichen, die nur $O(\ln N)$ Operationen benötigt.

2.1.4. Kondition eines linearen Problems

Das Gauß-Verfahren mit Teilpivotisierung ist ein robuster direkter Löser. Ohne Pivotisierung ist es aber für große Matrizen instabil. Auch können Probleme entstehen, wenn ein Gleichungssystem *schlecht konditioniert* ist. Eine schlechte *Kondition* bedeutet, daß die numerische Lösung äußerst sensitiv von den Matrixelementen oder von der rechten Seite der Gleichung abhängt. Im Extremfall ist eine numerische Lösung gar nicht möglich.

Ein Beispiel

Das Verhalten sei an einem Beispiel demonstriert. Dazu nehmen wir an, daß die verwendete Arithmetik nur eine Genauigkeit von 3 signifikanten Dezimalstellen besitzt. Wir betrachten das Problem

$$0.832x_1 + 0.448x_2 = 1.00, \tag{2.38a}$$
$$0.784x_1 + 0.421x_2 = 0. \tag{2.38b}$$

Mit

$$l_{21} = \frac{A_{21}}{A_{11}} = \frac{0.784}{0.832} = 0.942308\ldots \approx 0.942, \tag{2.39}$$

wobei die roten Ziffern der Rundung zum Opfer fallen, erhalten wir die neuen Matrixelemente

$$A_{22}^{(1)} = 0.421 - 0.942 \times 0.448 = 0.421 - 0.422016\ldots \approx -0.001, \tag{2.40}$$
$$b_2^{(1)} = 0 - 0.942 \times 1.00 = -0.942. \tag{2.41}$$

Dies führt auf das System

$$0.832x_1 + 0.448x_2 = 1.00, \tag{2.42a}$$
$$-0.001x_2 = -0.942. \tag{2.42b}$$

Backsubstitution ergibt die berechnete Lösung

$$x_1 = -506\ (-439) \qquad \text{und} \qquad x_2 = 942\ (817). \tag{2.43}$$

In Klammern ist die auf drei Stellen genaue *exakte* Lösung angegeben. Der Fehler der Lösung durch das Gauß-Verfahren beträgt $\approx 15\,\%$.

Um der Ursache für diese Abweichung auf den Grund zu gehen, kann man sich fragen: Wie sieht denn das Gleichungssystem aus (Koeffizienten $\tilde{A}_{ij}$), dessen exakte Lösung (2.43) ist? Dazu setzen wir das erhaltene Ergebnis (2.43) rückwärts in die Gleichungen ein, wobei wir

- dieselbe rechte Seite $\vec{b}$ verwenden,
- dasselbe $A_{11} = 0.832 = \tilde{A}_{11}$ verwenden, da diese Größe im Gauß-Verfahren nicht verändert wurde, und
- denselben Faktor $l_{21} = 0.942$ verwenden wie in (2.39). Dadurch wird $\tilde{A}_{21} = l_{21}\tilde{A}_{11} = 0.942 \times 0.832 = 0.783744$ festgelegt.

Mit (x_1, x_2) nach (2.43) können wir dann die noch fehlenden Matrixelemente $\tilde{A}_{12}$ und $\tilde{A}_{22}$ bestimmen. Die erste Gleichung (2.38a) liefert

$$0.832 \times (-506) + \tilde{A}_{12} \times 942 = 1, \tag{2.44}$$

mit dem Ergebnis $\tilde{A}_{12} = 0.447974$. Die zweite Gleichung (2.38b) ergibt

$$0.783744 \times (-506) + \tilde{A}_{22} \times 942 = 0, \tag{2.45}$$

was auf $\tilde{A}_{22} = 0.420992$ führt. Insgesamt man also das Gleichungssystem mit $\tilde{\mathsf{A}}$

$$0.832x_1 + 0.447974x_2 = 1.00, \tag{2.46a}$$
$$0.783744x_1 + 0.420992x_2 = 0. \tag{2.46b}$$

Seine exakte Lösung ist (2.43). Die Änderungen der Koeffizienten $\tilde{A}_{ij}$ gegenüber A_{ij} (2.38) sind blau hervorgehoben. Die maximale Abweichung der Koeffizienten von denjenigen aus (2.38) beträgt $\approx 0.03\,\%$. Diese kleine Variation führt auf eine große Variation (15 %) der exakten Lösung, was einer Verstärkung um einen Faktor von

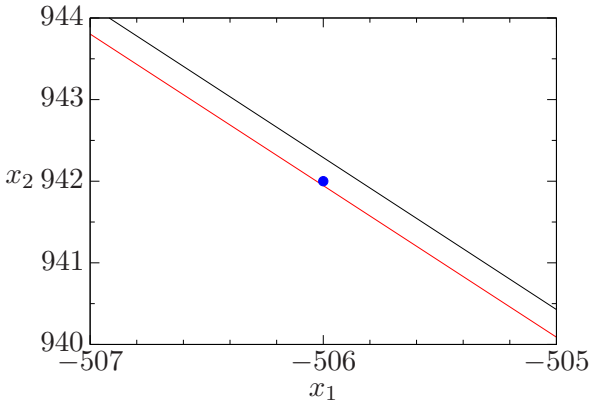


Abbildung 2.1.: Die Lösungen der beiden Gleichungen (2.38) in der Nähe des Punktes (2.43) (blau), den man als Schnittpunkt mittels Gauß-Verfahren und drei signifikanten Stellen erhält. Die beiden Geraden sind fast parallel.

ca. 500 entspricht! Diese Sensitivität liegt daran, daß die Matrix in (2.38) beinahe singulär ist. Für ein 2×2-System läßt sich die Singularität geometrisch interpretieren. Die beiden Geraden, die durch die beiden Gleichungen in (2.38) beschrieben werden, sind fast parallel. Eine kleine Änderung der Steigung oder des Achsenabschnitts einer der Geraden hat dann eine extreme Verschiebung des Schnittpunkts zur Folge. Beide Geraden aus (2.38) sind zusammen mit dem berechneten (falschen) Schnittpunkt in Abb. 2.1 gezeigt.

Wenn man das gegenüber (2.38) leicht veränderte System

$$0.832x_1 + 0.448x_2 = 1.00, \tag{2.47a}$$
$$0.784x_1 + (0.421+\epsilon)\, x_2 = 0, \tag{2.47b}$$

betrachtet, dann ändert sich die Steigung einer der beiden Geraden mit ϵ. Es gibt dann einen Wert $\epsilon^* = 0.00115384615\ldots$, für den beide Geraden parallel sind. Das System ist dann singulär. In der Umgebung von ϵ^* gibt es jedoch unendlich viele Werte ϵ, für welche (2.47) nahezu singulär ist.

Im allgemeinen (d.h. bei zufälligen Matrizen) ist es relativ unwahrscheinlich, daß eine Matrix singulär ist. Normalerweise ist es daher kaum möglich, allein von der Problemstellung darauf zu schließen, ob eine Matrix singulär oder nahezu singulär ist oder nicht. Man kann aber die resultierende Matrix hinsichtlich ihrer Konditionierung untersuchen. Ein Maß für die Kondition ist die *Konditionszahl*.

Kondition einer Matrix

David Hilbert
1862–1943

Ein klassisches Beispiel für eine schlecht konditionierte Matrix ist die Hilbert-Matrix

$$H_N = \begin{pmatrix} 1 & 1/2 & \ldots & 1/N \\ 1/2 & 1/3 & & 1/(N+1) \\ \vdots & & \ddots & \vdots \\ 1/N & & & 1/(2N-1) \end{pmatrix}. \tag{2.48}$$

Bei Verwendung einfacher Genauigkeit (ca. 8 Dezimalstellen) unterscheidet sich die numerisch berechnete Inverse von H_8 schon in der ersten Dezimalstelle von der exakten Inversen H_8^{-1}.[10]

Die Determinante einer Matrix A kann eine Aussage über die Kondition des zugehörigen linearen Problems machen. Dazu beachten wir, daß die Determinante das Volumen des Parallelepipeds angibt, das durch ihre Spaltenvektoren im $\mathbb{R}^N$ aufgespannt wird. Wenn wir alle Spaltenvektoren mit Hilfe ihrer Länge α_n auf 1

[10] Die Inverse von H_N lautet

$$(H_N)^{-1}_{i,j} = \frac{(-1)^{i+j}}{(i+j-1)} \frac{(N+i-1)!(N+j-1)!}{[(i-1)!(j-1)!]^2(N-i)!(N-j)!}.$$

normieren, erhalten das normierte Volumen des Parallelepipeds

$$V = \frac{\det \mathsf{A}}{\alpha_1 \dots \alpha_N}, \qquad \text{mit} \quad \alpha_n = \sqrt{A_{1n}^2 + \dots + A_{Nn}^2}. \tag{2.49}$$

Wenn nun alle Spaltenvektoren orthogonal zueinander sind, dann ist die Matrix A optimal konditioniert und das normiert Volumen (2.49) ist $V = 1$. Falls aber zwei Spaltenvektoren nahezu parallel sind (fast linear abhängig sind), dann ist die Matrix A schlecht konditioniert und das Volumen des Parallelepipeds tendiert zu Null. Je kleiner V also ist, desto schlechter ist die Kondition der Matrix A.

Unabhängig von dieser geometrischen Interpretation mißt man die Kondition einer Matrix am besten dadurch, daß man untersucht, wie empfindlich die Lösung $\vec{x}$ des linearen Problems

$$\mathsf{A} \cdot \vec{x} = \vec{b} \tag{2.50}$$

von einer Variation der Matrix A abhängt. Dazu betrachten wir eine kleine Variation $\delta\mathsf{A}$ von A. Dann wird sich auch die Lösung $\vec{x}$ etwas ändern. Für die geänderte Lösung $\vec{x} + \delta\vec{x}$ muß gelten

$$(\mathsf{A} + \delta\mathsf{A}) \cdot (\vec{x} + \delta\vec{x}) = \vec{b}, \tag{2.51}$$

oder

$$\mathsf{A} \cdot \vec{x} + \mathsf{A} \cdot \delta\vec{x} + \delta\mathsf{A} \cdot (\vec{x} + \delta\vec{x}) = \vec{b}. \tag{2.52}$$

Mit (2.50) erhalten wir

$$\mathsf{A} \cdot \delta\vec{x} = -\delta\mathsf{A} \cdot (\vec{x} + \delta\vec{x})\,. \tag{2.53}$$

Daher muß für die gesuchte Änderung $\delta\vec{x}$ der Lösung gelten

$$\delta\vec{x} = -\mathsf{A}^{-1} \cdot \delta\mathsf{A} \cdot (\vec{x} + \delta\vec{x})\,. \tag{2.54}$$

Um die Größe der Änderung zu beziffern, schätzen wir $\|\delta\vec{x}\|$ mit Hilfe der Norm ab. Dies führt auf[11]

$$\|\delta\vec{x}\| \le \left\|\mathsf{A}^{-1}\right\| \|\delta\mathsf{A}\| \, \|\vec{x} + \delta\vec{x}\|\,. \tag{2.55}$$

Für die relative Änderung der Lösung im Vergleich zur relativen Änderung der Norm der Matrix erhalten wir so

$$\frac{\|\delta\vec{x}\|}{\|\vec{x} + \delta\vec{x}\|} \le \underbrace{\|\mathsf{A}\| \left\|\mathsf{A}^{-1}\right\|}_{\text{cond}(\mathsf{A})} \frac{\|\delta\mathsf{A}\|}{\|\mathsf{A}\|}. \tag{2.56}$$

Das Verhältnis der relativen Änderungen definiert die *Konditionszahl* der Matrix A

$$\text{cond}(\mathsf{A}) = \|\mathsf{A}\| \left\|\mathsf{A}^{-1}\right\|. \tag{2.57}$$

Es ist leicht zu zeigen, daß $\text{cond}(\mathsf{A}) \ge 1$ ist.[12] Wenn $\text{cond}(\mathsf{A})$ dicht bei 1 liegt, so ist die Lösung nicht sehr sensitiv gegenüber einer Variation der Matrixelemente.

[11] Wenn die Matrix-Norm (A.667) auf der verwendeten Vektornorm basiert, gelten die Abschätzungen $\|\mathsf{A} \cdot \mathsf{B}\| \le \|\mathsf{A}\|\|\mathsf{B}\|$ (*Submultiplikativität*) und $\|\mathsf{A} \cdot \vec{x}\| \le \|\mathsf{A}\|\|\vec{x}\|$ (*Verträglichkeit*).

[12] Aufgabe: Zeige unter Verwendung der Definition der Matrix-Norm (A.667), daß $\text{cond}(\mathsf{A}) \ge 1$.

Die Matrix ist *gut konditioniert*. Je größer die Konditionszahl ist, desto sensitiver verhält sich die Lösung und die Matrix ist *schlecht konditioniert*. Die praktische Bedeutung der Konditionszahl hängt von der Genauigkeit der Daten und der Zahlendarstellung im Computer ab. Eine Konditionszahl von 10^6 kann beispielsweise den Verlust von 6 Dezimalstellen bedeuten. Dies ist kein Problem bei doppelter Genauigkeit (15–16 Stellen genau), kann sich aber bei einfacher Genauigkeit (7–8 Stellen genau) katastrophal auswirken. Man kann dieselbe Konditionszahl (2.57) auch aus der Sensitivität der Lösung $\vec{x}$ bezüglich einer Variation der rechten Seite $\vec{b}$ herleiten.[13]

Es ist normalerweise schwierig, die Konditionszahl genau zu berechnen, ohne die Matrix zu invertieren. Die Programmpakete LAPACK und LINPACK bieten jedoch Werkzeuge, mit deren Hilfe sich die Konditionszahl abschätzen läßt. Man findet, daß die Konditionszahl ungefähr mit dem Quadrat der Matrix-Dimension $\sim N^2$ anwächst.[14]

Die Konditionszahl nach (2.57) hängt von der gewählten Norm ab. Nimmt man die l_2-Norm (Euklidische Norm) so ist

$$\mathrm{cond}(\mathsf{A}) = \left| \frac{\sigma_{\max}}{\sigma_{\min}} \right|, \tag{2.58}$$

wobei $\sigma_{\max}$ und $\sigma_{\min}$ die betragsmäßig größten und kleinsten *Singulärwerte* von A sind (siehe (A.661)). Dies gilt für den allgemeinen Fall, bei dem Eigenvektoren von A mehrfach auftreten können. Ist die Matrix A *normal*, dann sind alle Eigenvektoren orthogonal zueinander und es gilt in der l_2-Norm

$$\mathrm{cond}(\mathsf{A}) = \left| \frac{\lambda_{\max}}{\lambda_{\min}} \right|, \tag{2.59}$$

wobei $\lambda_{\max}$ und $\lambda_{\min}$ die betragsmäßig größten und kleinsten *Eigenwerte* von A sind. Falls schließlich A *unitär* ist, dann hat A orthonormale Spaltenvektoren (vgl. (A.612)) und das Volumen des zugehörigen Parallelepipeds nach (2.49) ist $V = 1$. In diesem Fall ist $\mathrm{cond}(\mathsf{A}) = 1$.

Die nach (2.58) definierte Konditionszahl kann in MATLAB durch den Befehl `cond(A)` berechnet werden, denn der betragsmäßig größte und kleinste Singulärwert läßt sich iterativ ermitteln (für die Bestimmung von Eigenwerten siehe Kap. 6.2). Mit `cond(A,p)` berechnet man die Konditionszahl in der p-Norm und `condest(A)` berechnet eine untere Grenze für die l_1-Norm einer Matrix, was insbesondere bei großen dünn-besetzten Matrizen hilfreich ist.

[13] Aufgabe: Betrachten Sie die Sensitivität der Lösung $\vec{x} \to \vec{x} + \delta\vec{x}$ des linearen Problems

$$\mathsf{A} \cdot \vec{x} = \vec{b}$$

bezüglich einer Variation der rechten Seite $\vec{b} \to \vec{b} + \delta\vec{b}$. Drücken Sie die Größe der relativen Änderung der Lösung $\|\delta\vec{x}\| \,/\, \|\vec{x}\|$ durch diejenige der rechten Seite aus und stellen Sie einen Zusammenhang her mit der Konditionszahl der Matrix A.

[14] Für Tridiagonal-Matrizen mit konstanten Diagonalen kann man die Konditionszahl explizit berechnen und auch ihre Abhängigkeit $\sim N^2$ bestätigen (siehe Golub and Ortega, 1996).

2.1.5. Vorkonditionierung

Um dem Problem einer schlechten Kondition zu begegnen, kann man das lineare System (2.50) mit einer regulären Matrix P^{-1} von links multiplizieren, genau wie bei einer Ähnlichkeitstransformation (siehe Kap. A.5.3). Man erhält dann dieselbe Lösung $\vec{x}$ aus dem Problem

$$\mathsf{P}^{-1} \cdot \mathsf{A} \cdot \vec{x} = \mathsf{P}^{-1} \cdot \vec{b}. \tag{2.60}$$

Hierbei sollte die Matrix P^{-1} so beschaffen sein, daß $\mathsf{P}^{-1} \cdot \mathsf{A}$ eine deutlich geringere Konditionszahl besitzt als A.

Zwei Extremfälle sind sofort klar. Sei $\mathsf{P} = \mathsf{I}$, dann ändert sich die Kondition nicht. Falls andererseits $\mathsf{P} = \mathsf{A}$, dann ist $\mathrm{cond}(\mathsf{P}^{-1} \cdot \mathsf{A}) = \mathrm{cond}(\mathsf{A}^{-1} \cdot \mathsf{A}) = \mathrm{cond}(\mathsf{I}) = 1$. Um dies zu erreichen, müßte man A aber invertieren, was einer Lösung des Ausgangsproblem gleichkäme. Aus diesem Grund versucht man, für die Matrix P eine leicht invertierbare Näherung von A zu finden. Leider ist dies nicht immer einfach und natürlich auch vom jeweiligen Problem abhängig.

2.2. Iterative Lösung linearer Gleichungssysteme

Jedes lineare System kann im Prinzip mit dem Gauß-Algorithmus bzw. mit der LU-Zerlegung gelöst werden. Die Dreiecksmatrizen der LU-Zerlegung sind normalerweise *voll besetzt*. Dies ist auch dann der Fall, wenn die Matrix A *dünn besetzt* ist. Der Gauß-Algorithmus ist jedoch mit einem erheblichen numerischen Aufwand verbunden (Anzahl der Operationen $\sim N^3$).

In den allermeisten Fällen stellt das zu lösende lineare Problem eine diskretisierte Version eines kontinuierlichen Problems dar. Der bei der Diskretisierung entstehende Diskretisierungsfehler ist normalerweise sehr viel größer als der Rundungsfehler der Computerarithmetik. Es macht daher keinen Sinn, das lineare System genauer zu lösen als durch den Diskretisierungsfehler vorgegeben. Dies legt eine iterative Lösung des linearen Problems nahe, auch für dünn besetzte Matrizen.[15] Iterative Methoden sind immer dann effizient, wenn jede Iteration billig ist und wenn nicht zu viele Iterationen benötigt werden, um zu einer hinreichenden Genauigkeit der Approximation der exakten Lösung zu kommen. Das Buch von Saad (2003) enthält einen sehr guten Überblick über iterative Verfahren.

2.2.1. Allgemeines Konzept

Zur Demonstration des allgemeinen Konzeptes betrachten wir das lineare System

$$\mathsf{A} \cdot \vec{x} = \vec{b} \tag{2.61}$$

und zerlegen die Matrix A in

$$\mathsf{A} = \mathsf{M} - \mathsf{N}. \tag{2.62}$$

[15] Nichtlineare Systeme erfordern generell iterative Methoden.

Damit lautet das Problem

$$\mathsf{M} \cdot \vec{x} = \mathsf{N} \cdot \vec{x} + \vec{b}. \tag{2.63}$$

Um zu einem Iterationsschema zu kommen, setzen wir einfach

$$\mathsf{M} \cdot \vec{x}^{(n+1)} = \mathsf{N} \cdot \vec{x}^{(n)} + \vec{b} \tag{2.64}$$

Die entscheidende Frage ist, wie man A in M und N aufspalten muß, um eine schnelle Konvergenz von $\vec{x}^{(n)}$ gegen die Lösung von (2.61) zu erzielen. Für eine effiziente Lösung sollte das Iterationsschema (2.64) mit möglichst wenigen Operationen möglichst schnell konvergieren. Dies erfordert

1. eine schnelle Berechnung der rechten Seite von (2.64): Wenn wir davon ausgehen, daß A dünn besetzt ist, dann sind auch M und N dünn besetzt (im Gegensatz zur LU-Zerlegung) und $\mathsf{N} \cdot \vec{x}^{(n)}$ ist schnell zu berechnen.

2. eine einfache Lösung des verbleibenden linearen Systems: Um das System schnell lösen zu können, sollte M schnell zu invertieren sein. Daher sollte M möglichst diagonal, tridiagonal oder von Dreiecksform sein.

3. eine Konvergenz nach wenigen Iterationen: Um mit wenigen Iterationen auszukommen, sollte schließlich M eine möglichst gute Approximation von A darstellen, wodurch $\mathsf{N} \cdot \vec{x}$ in einem gewissen Sinne *klein* wird. Im Grenzfall $\mathsf{N} \cdot \vec{x} = 0$ wäre man ja nach nur einer Iteration fertig.

Bevor wir uns einer theoretischen Konvergenzbetrachtung zuwenden, wollen wir noch einige weitere Größen definieren und deren Beziehungen untereinander klären. Nach n Iterationen haben wir im allgemeinen nur eine Näherung der exakten Lösung $\vec{x}$ erhalten. Dann ist (2.61) nicht exakt erfüllt, und es bleibt ein *Residuum* $\vec{\rho}^{(n)}$ übrig

$$\mathsf{A} \cdot \vec{x}^{(n)} = \vec{b} - \vec{\rho}^{(n)}. \tag{2.65}$$

Wenn wir dies von der exakten Gleichung abziehen, erhalten wir

$$\mathsf{A} \cdot \underbrace{\left(\vec{x} - \vec{x}^{(n)}\right)}_{\vec{\epsilon}^{(n)}} = \mathsf{A} \cdot \vec{\epsilon}^{(n)} = \vec{\rho}^{(n)}. \tag{2.66}$$

Hierbei haben wir den *Fehler* $\vec{\epsilon}^{(n)} := \vec{x} - \vec{x}^{(n)}$ definiert. Zwischen dem Fehler und dem Residuum besteht ein linearer Zusammenhang. Wenn das Residuum verschwindet, dann verschwindet auch der Fehler. Eine weitere nützliche Größe ergibt sich, wenn wir $\mathsf{M} \cdot \vec{x}^{(n)}$ von (2.64) abziehen. Dann erhalten wir eine Gleichung für die *Korrektur* $\vec{\delta}^{(n)} := \vec{x}^{(n+1)} - \vec{x}^{(n)}$

$$\mathsf{M} \cdot \underbrace{\left(\vec{x}^{(n+1)} - \vec{x}^{(n)}\right)}_{\vec{\delta}^{(n)}} = \underbrace{\underbrace{(\mathsf{N} - \mathsf{M})}_{-\mathsf{A}} \cdot \vec{x}^{(n)} + \vec{b}}_{\vec{\rho}^{(n)}}. \tag{2.67}$$

Für die Korrektur gilt also

$$\mathsf{M} \cdot \vec{\delta}^{(n)} = \vec{\rho}^{(n)}. \tag{2.68}$$

2.2.2. Konvergenz iterativer Löser

Hier wollen wir mit einfachen Betrachtungen untersuchen, worauf es ankommt, wenn man eine schnelle Konvergenz eines iterativen Verfahrens erhalten möchte. Dazu suchen wir zunächst eine Gleichung für die Entwicklung des *Iterationsfehlers* $\vec{\epsilon}^{(n)} = \vec{x} - \vec{x}^{(n)}$. Wenn man von der Ausgangsgleichung (2.63) die Iterationsgleichung (2.64) subtrahiert, erhält man

$$\mathsf{M} \cdot \vec{\epsilon}^{(n+1)} = \mathsf{N} \cdot \vec{\epsilon}^{(n)} \qquad \Rightarrow \qquad \vec{\epsilon}^{(n+1)} = \mathsf{M}^{-1} \cdot \mathsf{N} \cdot \vec{\epsilon}^{(n)}. \tag{2.69}$$

Für ein konvergentes Verfahren muß gelten: $\lim_{n\to\infty} \vec{\epsilon}^{(n)} = 0$. Für den Grenzwert der Iteration (2.69) und die Konvergenzgeschwindigkeit spielen die *Eigenwerte* λ_k und die *Eigenvektoren* $\vec{\psi}_k$ der Matrix $\mathsf{M}^{-1} \cdot \mathsf{N}$ eine entscheidende Rolle. Sei daher

$$\mathsf{M}^{-1} \cdot \mathsf{N} \cdot \vec{\psi}_k = \lambda_k \cdot \vec{\psi}_k, \tag{2.70}$$

mit $k = 1, \ldots, K$, wobei K die Anzahl der Gleichungen (Dimension der Matrix A) ist. Wenn wir annehmen, daß die Eigenvektoren ein *vollständiges Funktionensystem* bilden, dann können wir jeden beliebigen Anfangsfehler als Superposition von Eigenvektoren darstellen

$$\vec{\epsilon}^{(0)} = \sum_{k=1}^{K} a_k \vec{\psi}_k. \tag{2.71}$$

Wenn man dies in (2.69) einsetzt, sieht man, daß jeder Iterationsschritt lediglich einen Faktor λ_k vor $\vec{\psi}_k$ ausspuckt. Somit erhalten wir

$$\vec{\epsilon}^{(n)} = \sum_{k=1}^{K} a_k \left(\lambda_k\right)^n \vec{\psi}_k. \tag{2.72}$$

Wenn $\vec{\epsilon}^{(n)}$ für $n \to \infty$ gegen Null gehen soll, muß für *alle* Eigenwerte gelten: $|\lambda_k| < 1$, denn die Eigenvektoren sind linear unabhängig voneinander.

Nach einigen Iterationen wird derjenige Term in (2.72) dominieren, der zum betragsmäßig größten Eigenwert gehört. Dies sei o.B.d.A. λ_1. Dann ist

$$\vec{\epsilon}^{(n)} \approx a_1 \left(\lambda_1\right)^n \vec{\psi}_1. \tag{2.73}$$

Wenn wir die Iteration bei einer *Toleranzschwelle* $\|\vec{\epsilon}^{(n)}\| = O(\delta)$ für den Iterationsfehler abbrechen, erhalten wir als Abbruchbedingung (es sei $\|\vec{\psi}_1\| = O(1)$)

$$\delta \sim a_1 \left|\lambda_1\right|^n, \tag{2.74}$$

woraus wir durch Logarithmieren die Anzahl n der erforderlichen Iterationen erhalten

$$n \sim \frac{\ln\left|\delta/a_1\right|}{\ln\left|\lambda_1\right|}. \tag{2.75}$$

Wenn also der *spektrale Radius* R (größter Betrag eines Eigenwerts: $R = \max_k |\lambda_k|$; hier $R = |\lambda_1|$) gegen 1 geht, wird n sehr groß und die Konvergenz ist langsam. Umgekehrt ist die Konvergenz schnell, wenn der Betrag des betragsmäßig größten Eigenwerts von $\mathsf{M}^{-1} \cdot \mathsf{N}$ möglichst klein wird. Dies entspricht dem Fall, in dem $\mathsf{N} \cdot \vec{x}$ möglichst klein und $\mathsf{M} \approx \mathsf{A}$ ist. Denn für $\mathsf{M} = \mathsf{A}$ und damit $\mathsf{N} = 0$ wäre man ja schon nach einer einzigen Iteration fertig.

Um entscheiden zu können, wann man eine Iteration abbrechen sollte, muß man den *Iterationsfehler* $\vec{\epsilon}^{(n)}$ bestimmen. Da man die Eigenwerte λ_k von $\mathsf{M}^{-1} \cdot \mathsf{N}$ aber meist nicht kennt oder sie nur mit großem Aufwand berechnen kann, muß der Iterationsfehler anders abgeschätzt werden. Hierbei kann man ausnutzen, daß die Beträge des Fehlers $\vec{\epsilon}^{(n)}$, des Residuums $\vec{\rho}^{(n)}$ und des Inkrements $\vec{\delta}^{(n)}$ nach einer Anfangsphase in ähnlicher Weise mit dem Iterationsindex n skalieren.

2.2.3. Einige elementare iterative Methoden

Um iterative Verfahren zu demonstrieren, betrachten wir als Beispiel die inhomogene stationäre Diffusionsgleichung in zwei Dimensionen (*Poisson-Gleichung*)

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = b. \tag{2.76}$$

Sie beschreibt u.a. das stationäre Temperaturfeld in einer dünnen Platte, die einer homogenen Wärmezufuhr (z.B. durch Strahlung) ausgesetzt ist. Zusätzlich müssen noch eine Anfangsverteilung der Temperatur und Randbedingungen (z.B. die Funktionswerte auf der Berandung des Gebiets) vorgegeben werden.

Die Poisson-Gleichung (2.76) wird nun diskretisiert. Dazu überziehen wir die (x, y)-Ebene mit einem äquidistanten Gitter der Gitterweiten Δx in x-Richtung und Δy in y-Richtung. Die Gitterlinien werden mit Indizes i (in x-Richtung) und j (in y-Richtung) numeriert. An den diskreten Schnittpunkten (i, j) der Gitterlinien i und j wird die endliche Anzahl von Unbekannten $\phi_{i,j}$ definiert (Abb. 2.2a). Wenn man die in der Poisson-Gleichung (2.76) auftretenden zweiten Ableitungen durch zentrale finite Differenzen approximiert, erhält man die diskretisierte Version der Poisson-Gleichung

$$\frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{\Delta x^2} + \frac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{\Delta y^2} = b_{i,j}. \tag{2.77}$$

Sie muß für jeden inneren Gitterpunkt (i, j) erfüllt sein. Wenn wir in jeder Raumrichtung K Punkte verwenden, sind dies $N = K^2$ Gleichungen für K^2 Unbekannte $\phi_{i,j}$.

Wir führen nun die *geographische Notation* ein, wobei die Punkte in der Umgebung des jeweiligen zentralen Punktes $P \mathrel{\widehat{=}} (i, j)$ entsprechend ihrer Himmelsrichtung benannt werden (Abb. 2.2). Dann kann man Gleichung (2.77) schreiben als

$$A_P \phi_P + \sum_{l=W,S,N,E} A_l \phi_l = b_P, \tag{2.78}$$

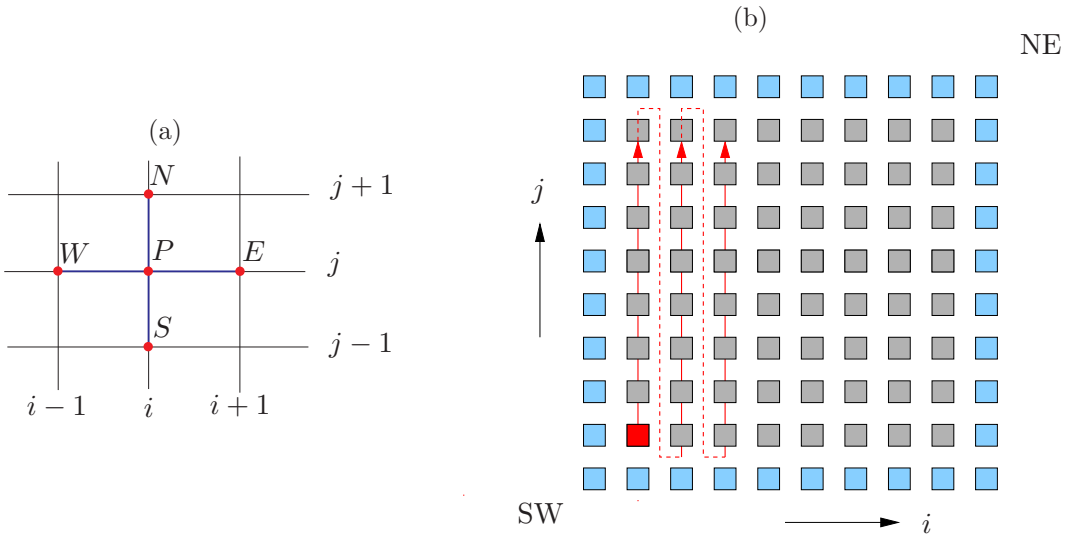


Abbildung 2.2.: (a) 5-Punkt-Stern und geographische Bezeichnung der Gitterpunkte. (b) Anordnung der Unbekannten $\phi_{i,j}$ innerhalb des Vektors $\vec{\phi}$. Die Gitterpunkte werden von unten links beginnend schnell von Süd (S) nach Nord (N) durchlaufen und dann (langsamer Index) von West (W) nach Ost (E). Die Werte am Rand (blau) sind vorgegeben.

wobei der Index P alle $N = K \times K$ Gitterpunkte durchläuft und die Summe jeweils nur über die 4 physikalischen Nachbarpunkte des aktuellen Punkts P zu nehmen ist. Wir ordnen nun die N Variablen wie in Abb. 2.2b in der Form

$$\vec{\phi} = (\phi_{1,1} : \phi_{1,K},\ \phi_{2,1} : \phi_{2,K},\ \ldots,\ \phi_{K,1} : \phi_{K,K})^{\mathrm{T}}, \tag{2.79}$$

wobei der zweite (schnelle) Index vertikal von unten (S) nach oben (N) läuft, und der erste (langsame) Index von links (W) nach rechts (E). Dann ist der globale Index n des Vektors $\vec{\phi} = \phi_n$ gegeben durch $n = (i-1)K + j$.

Wenn man nun die N Gleichungen für alle inneren Punkte (grau in Abb. 2.3b) aufstellt, erhält man ein lineares Gleichungssystem, wobei die Matrix A *penta-diagonal* ist, so wie in Abb. 2.3 dargestellt. In unserem speziellen Beispiel eines äquidistanten Gitters sind die Diagonalen konstant mit den Werten $A_P = 4$ und $A_W = A_S = A_N = A_E = -1$ (bis auf einen gemeinsamen Faktor $\Delta x^{-2} = \Delta y^{-2}$, vgl. (2.77)).

Jacobi-Iteration

Eine der einfachsten iterativen Methoden ist die *Jacobi-Iteration*. Hierbei ist M diagonal und die Diagonalelemente werden identisch mit denjenigen von A gewählt.

Wenn nun $\mathsf{M} = \mathrm{diag}(\mathsf{A})$ identisch mit der Diagonalen von A gewählt wird (siehe Abb. 2.4a), lautet die *Jacobi-Iteration* entsprechend (2.64)

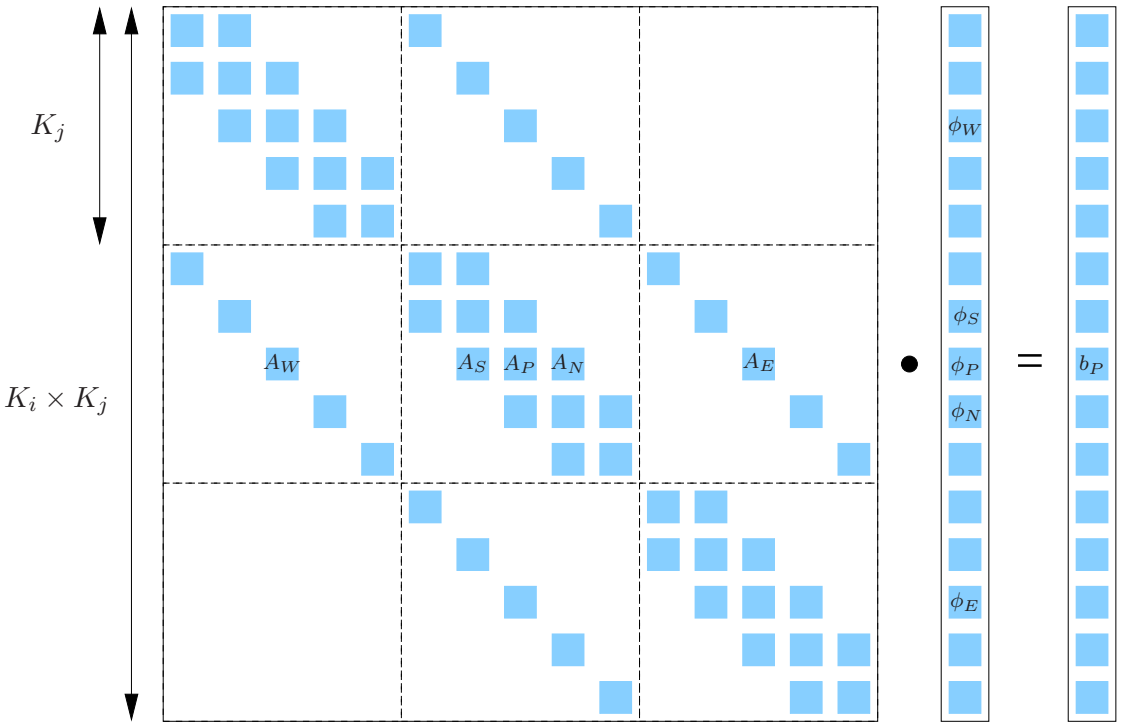
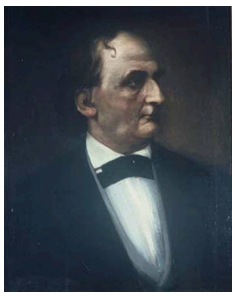


Abbildung 2.3.: Anordnung der Koeffizienten in der Matrix A für finite Differenzen wie in Abb. 2.2a bei Verwendung der geographischen Notation und Anordnung der Variablen wie in Abb. 2.2b. Die leeren Matrixelemente auf den beiden ersten Nebendiagonalen symbolisieren Werte von den Randbedingungen. Sie werden dem Vektor $\vec{b}$ zugeschlagen, so daß diese Einträge in der Matrix Null sind. Genauso werden die bekannten Funktionswerte am Rand behandelt, die in die ersten K Gleichungen und die letzten K Gleichungen eingehen.



Philipp Ludwig Ritter von Seidel 1821–1896

$$\phi_P^{(n+1)} = \frac{1}{A_P}\left(b_P - \sum_{l=W,S,N,E} A_l \phi_l^{(n)}\right). \tag{2.80}$$

Der Aufwand zur Lösung in jedem Iterationsschritt ist $O(N)$. Man kann zeigen, daß die Anzahl der für die Konvergenz erforderlichen Iterationen proportional ist zur Anzahl der Unbekannten $N = K^2$. Daher benötigt man zur Lösung des Systems $\sim N^2$ Operationen. Dies sind mehr Operationen als man für einen direkten Löser nach Art von TDMA (Aufwand $\sim N$) benötigen würde. Daher wird die Jacobi-Iteration selten verwendet.

Gauß-Seidel-Iteration

Bei der *Gauß-Seidel-Iteration* wird für M die *untere Dreiecksmatrix* von A verwendet (Abb. 2.4b). Diese stellt eine bessere Approximation von A dar als die Diagonale allein. Wir erwarten also eine Reduktion der Anzahl der erforderlichen Iterationen im

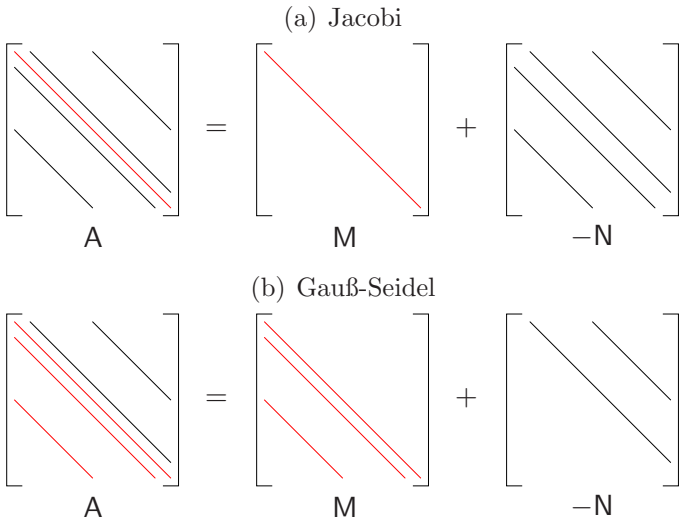


Abbildung 2.4.: Symbolische Darstellung der Zerlegung der Matrix A in die Iterationsmatrix M und einen Teil −N, welcher der rechten Seite der Gleichung zugeschlagen wird (vgl. (2.64)). (a) Jacobi-Iteration, (b) Gauß-Seidel-Iteration.

Vergleich zum Jacobi-Verfahren. Damit lautet die *Gauß-Seidel-Iteration* (vergleiche Abb. 2.3)

$$\sum_{l=W,S,P} A_l \phi_l^{(n+1)} = b_P - \sum_{l=N,E} A_l \phi_l^{(n)}. \tag{2.81}$$

Wenn man die Gleichungen beginnend mit der linken unteren Ecke des Gitters (SW-Ecke) löst, sind für jeden Punkt P die Werte an den Punkten S und W jeweils schon zuvor berechnet worden und daher bekannt. Dann kann man (2.81) nach $\phi_P^{(n+1)}$ auflösen

$$\phi_P^{(n+1)} = \frac{1}{A_P}\left(b_P - A_W\phi_W^{(n+1)} - A_S\phi_S^{(n+1)} - A_N\phi_N^{(n)} - A_E\phi_E^{(n)}\right). \tag{2.82}$$

Wie man in Abb. 2.5 sieht, konvergiert die Gauß-Seidel-Iteration doppelt so schnell wie die Jacobi-Iteration, benötigt aber immer noch $\sim N$ Iterationen und einen Gesamtaufwand von $O(N^2)$ Operationen. D.h., auch die Gauß-Seidel-Iteration ist noch zu langsam.

SOR-Methode

Durch eine Modifikation der Gauß-Seidel-Methode kann man die Konvergenz wesentlich beschleunigen. Die Idee dabei ist, die in jedem Schritt durchgeführte Korrektur künstlich zu vergrößern und damit die aktuelle Näherungslösung $\phi^{(n)}$ stärker zu korrigieren. Die Iteration mittels *sukzessiver Überrelaxation* (***S****uccessive* ***O****ver-*

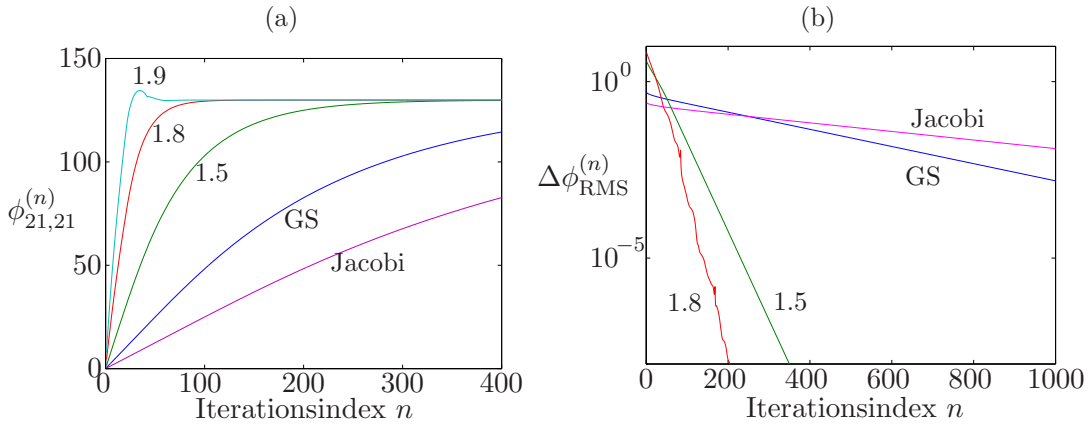


Abbildung 2.5.: (a) Konvergenz des Jacobi-, Gauß-Seidel- (GS) und des SOR-Verfahrens in Abhängigkeit vom Relaxationsfaktor ω (als Zahlen angegeben) für die Poisson-Gleichung $\nabla^2\phi = -1$, die auf $K_i \times K_j = 41 \times 41$ (inneren) Punkten mit zentralen finiten Differenzen auf einem Quadrat mittels finiter Differenzen nach (2.77) diskretisiert wurde. Die Anfangsbedingung und Randbedingungen sind $\phi^{(0)} \equiv 0$ und $\phi(\text{Rand}) = 0$. Für $\omega = 1.9$ ist man schon oberhalb des optimalen Relaxationsfaktors, da die Konvergenz nicht mehr monoton ist. Das (langsame) Gauß-Seidel-Verfahren (GS) entspricht $\omega = 1$. (b) Logarithmische Auftragung des mittleren RMS-Inkrements $\Delta\phi^{(n)}_{\mathrm{RMS}} := \sqrt{(K_iK_j)^{-1}\sum_{l=1}^{K_iK_j}\left(\phi_l^{(n+1)} - \phi_l^{(n)}\right)^2}$.

***R**elaxation*, SOR) lautet (wieder beginnend im Südwesten)

$$\begin{aligned}\phi_P^{(n+1)} &= \phi_P^{(n)} + \omega\left({}^{\mathrm{GS}}\phi_P^{(n+1)} - \phi_P^{(n)}\right)\\ &\overset{(2.82)}{=} \frac{\omega}{A_P}\left(b_P - A_W\phi_W^{(n+1)} - A_S\phi_S^{(n+1)} - A_N\phi_N^{(n)} - A_E\phi_E^{(n)}\right) + (1-\omega)\,\phi_P^{(n)}.\end{aligned} \tag{2.83}$$

Hierbei ist ${}^{\mathrm{GS}}\phi_P^{(n+1)}$ der Wert, der sich aus dem Gauß-Seidel-Verfahren (2.82) ergibt und ω der *Überrelaxationsfaktor*. Er gibt an um welchen Faktor die Gauß-Seidel-Korrektur $\left({}^{\mathrm{GS}}\phi_P^{(n+1)} - \phi_P^{(n)}\right)$ verstärkt wird ($\omega > 1$). Damit wird die Konvergenz beschleunigt. Für $\omega = 1$ erhält man die ursprüngliche Gauß-Seidel-Iteration (2.82). Für einfache Probleme wie die Poisson-Gleichung kann man den optimalen Wert $\omega_{\mathrm{opt}} > 1$ theoretisch berechnen.[16] Für kompliziertere Probleme kann man ihn nur empirisch finden. Dabei hilft es zu wissen, daß die Konvergenz monoton ist für

[16] Fletcher (1991) gibt an

$$\omega_{\mathrm{opt}} = \frac{2}{1+\sqrt{1-\mu^2}},$$

wobei μ der größte Eigenwert von $\mathsf{I} - \mathrm{diag}(\mathsf{A})^{-1}\cdot\mathsf{A}$ ist. Die Lösung dieses Problems kann aber genau so aufwendig sein, wie das Ausgangsproblem selbst.

$\omega < \omega_{\text{opt}}$ und oszillatorisch für $\omega > \omega_{\text{opt}}$. Für $\omega = \omega_{\text{opt}}$ ist die erforderliche Zahl der Iterationen proportional zur Wurzel aus der Anzahl der Unbekannten ($\sim \sqrt{N}$), was eine substantielle Verbesserung gegenüber der Gauß-Seidel-Iteration darstellt, bei der die Anzahl der Iterationen ja linear mit der Anzahl der Unbekannten ($\sim N$) ansteigt. Das typische Konvergenzverhalten der drei elementaren Methoden ist in Abb. 2.5 für die Lösung von $\nabla^2\phi = -1$ auf dem Einheitsquadrat dargestellt.

Die raum-zeitliche Entwicklung der iterierten Lösung für das Gauß-Seidel-Verfahren ist in Abb. 2.6 gezeigt. Man erkennt, daß kurzwellige Schwankungen sehr schnell gedämpft werden. Das macht den Gauß-Seidel-Operator zu einem guten *Glättungsoperator* für sogenannte Mehrgitterverfahren.

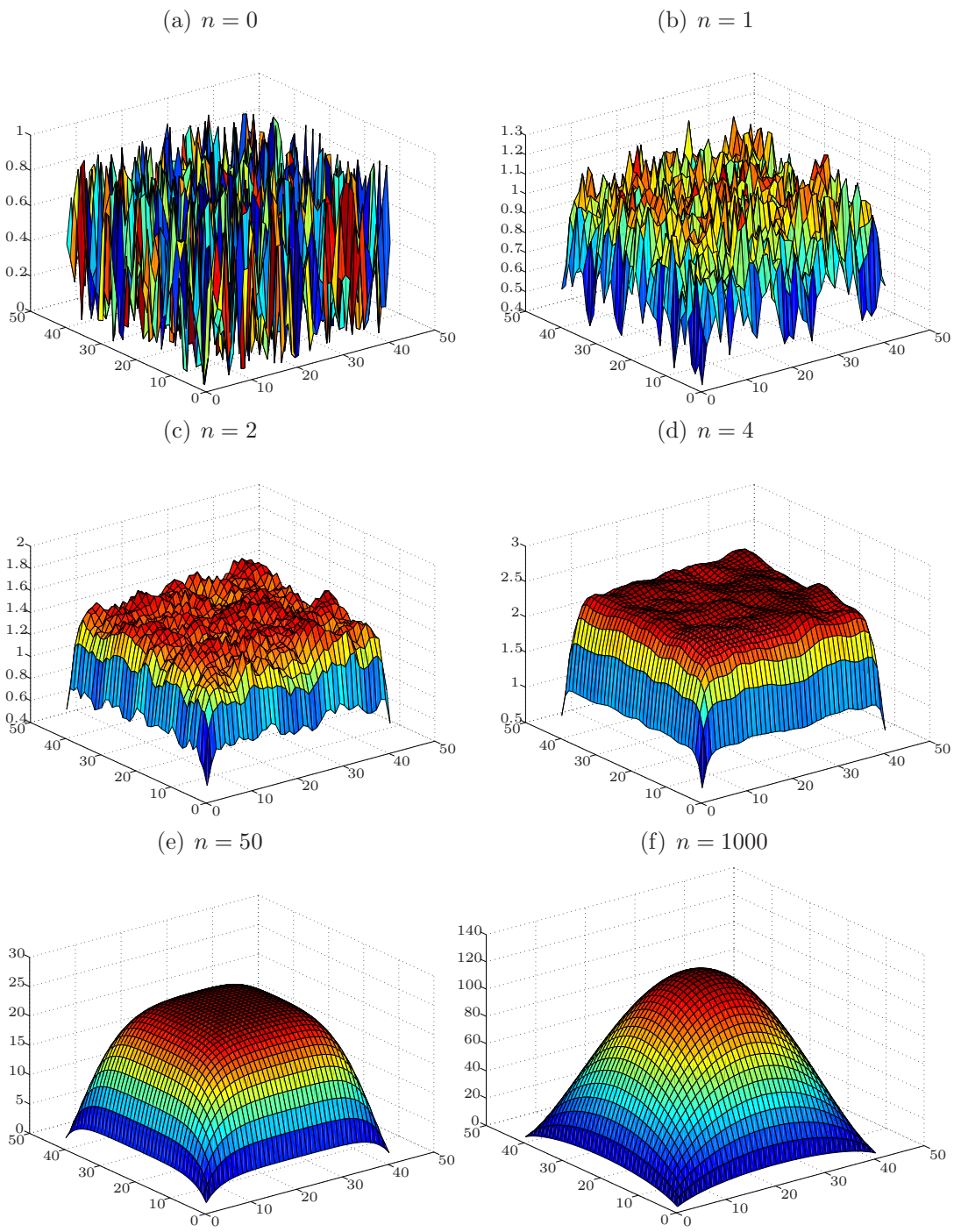


Abbildung 2.6.: Gauß-Seidel-Iteration von $\nabla^2\phi = -1$ auf einem Quadrat mit homogenen Dirichlet-Randbedingungen und einem Gitter von 41×41 internen Punkten. Gezeigt sind die zufälligen Anfangsbedingung ($n = 0$) auf dem Definitionsbereich $[0, 1] \times [0, 1]$ mit Funktionswerten $\phi_{ij} \in [0, 1]$ (a), sowie die Approximationen der Lösung ϕ nach $n = 1$ (b), $n = 2$ (c), $n = 4$ (d), $n = 50$ (e) und $n = 1000$ (f) Iterationen. Nur die Lösung über den inneren Punkten ist dargestellt. Man erkennt die guten Glättungseigenschaften der Gauß-Seidel-Iteration nach nur wenigen Iterationsschritten. Beachte die unterschiedlichen Skalen.

3. Interpolation und Approximation

Interpolation Zur Berechnung des elastischen Verhaltens von Körpern oder der Strömung um einen Körper herum (z.B. Rotorblatt, Tragflügel, etc.) muß man meist ein räumliches Rechengitter erstellen, welches der gegebenen Kontur der Körperoberfläche angepaßt ist (siehe Abb. 3.1). Um bei der Gittergenerierung hinreichend flexibel zu sein, ist es erforderlich, die Kontur an jedem beliebigen Punkt möglichst genau mathematisch beschreiben zu können. Dazu verwendet man einen endlichen Satz von Punkten auf der Kontur. Alle unendlich vielen Punkte zwischen den diskreten Stützstellen werden dann durch Funktionen beschrieben, die man durch *Interpolation* erhält.

Ein ähnliches Problem liegt vor, wenn man die Trajektorie eines Partikels in einer Strömung $\vec{u}(\vec{x},t)$ verfolgen möchte. Oft ist das Teilchen so klein, daß es die Strömung nicht beeinflußt. Dann kann man die Strömung vorab numerisch an diskreten Stützpunkten berechnen. Da man zur Berechnung der Trajektorie $\vec{X}(t)$ des Teilchens entsprechend

$$\ddot{\vec{X}}(t) = \vec{F}[\vec{u}(\vec{X},t), \dot{\vec{X}}] \tag{3.84}$$

das Geschwindigkeitsfeld aber an *allen* Raumpunkten benötigt, muß auch hier zwischen den Gitterpunkten interpoliert werden.

Approximation Ein verwandtes Problem ergibt sich, wenn man eine sehr große Anzahl von gegebenen Funktionswerten (z. B. fehlerbehaftete Meßwerte) als Funktion einer unabhängigen Variablen durch eine einfache (glatte) Funktion annähern möchte. Dies nennt man *Approximation*. Hierzu kommen zum Beispiel *Polynome* in Betracht. Sie besitzen den Vorteil, daß sie sich mit Hilfe der Grundrechenarten auswerten lassen und einfach differenziert und integriert werden können. Damit lassen sich sehr leicht Näherungen für Ableitungen und Integrale der betrachteten Funktion finden. Ein Nachteil der Polynome ist jedoch, daß sie für $x \to \pm\infty$ divergieren, die zu approximierende Funktion meist jedoch nicht.

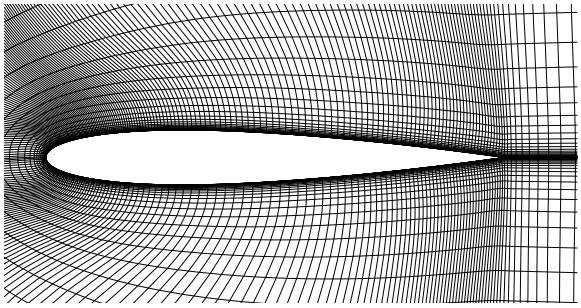


Abbildung 3.1.: Angepaßtes Gitter nach *Numerische Methoden der Thermo- und Fluiddynamik* von T. Rung, L. Xue, J. Yan, M. Schatz und F. Thiele (2002).

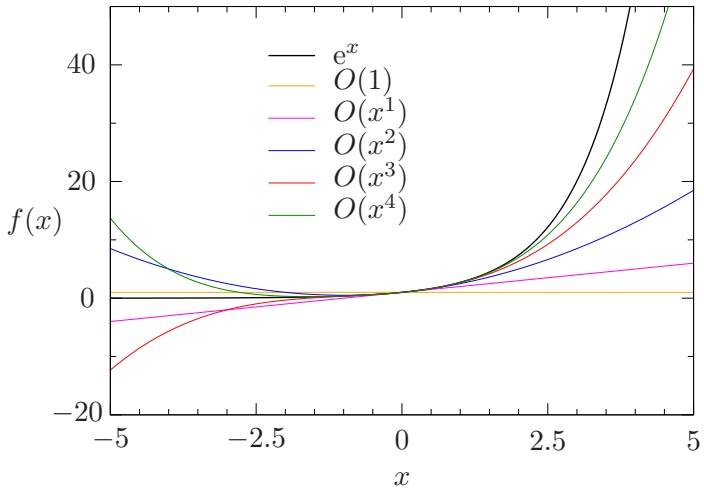


Abbildung 3.2.: Entwicklung der Exponentialfunktion e^x (schwarz) um den Punkt $x_0 = 0$. Die Abbruchordnung der Reihe ist farbig gekennzeichnet (bis zur vierten Ordnung).

3.1. Approximation durch eine Taylorreihe

Zunächst betrachten wir die Taylorreihe einer Funktion $f(x)$, entwickelt um den Punkt x_0

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \ldots + \frac{1}{n!}f^{(n)}(x_0)(x - x_0)^n + \frac{1}{(n+1)!}f^{(n+1)}(z)(x - x_0)^{n+1}. \tag{3.85}$$

Wenn man die Taylor-Reihe bei n abbricht, verbleibt ein Fehler (*Lagrangesches Restglied*), der durch den blauen Term gegeben ist. Für den Fehler muß man die $(n+1)$-te Ableitung an einer Zwischenstelle $z \in [x_0, x]$ (falls $x > x_0$ ist) auswerten.

Bei der Approximation einer Funktion durch eine Taylor-Reihe, werden nur Informationen (Funktionswert und Ableitungen) an einer einzigen Stelle x_0 ausgewertet. Als Beispiel sind in Abb. 3.2 die ersten Approximationen (oskuliernde Polynome) der Exponentialfunktion e^x gezeigt, die um den Punkt $x_0 = 0$ entwickelt wurde.

Brook Taylor
1685–1731

Bei der Approximation bekannter Funktionen kann man den Fehler mit Hilfe des Restglieds abschätzen. Da man aber den exakten Zwischenwert z nicht kennt, muß man den Fehler (das Restglied) über dem Intervall maximieren, über welches die Approximation verwendet wird. Wenn man die $(n+1)$-te Ableitung nicht berechnen kann, wird es sehr schwierig, den Fehler genau zu quantifizieren. Oft ist dies sogar unmöglich.

3.2. Polynom-Interpolation

Alexandre-Théophile Vandermonde 1735–1796

Bei der Approximation durch eine Taylor-Reihe stammt die Information nur von einem *einzigen* Punkt, dem Entwicklungspunkt. Nur dort ist die Approximation exakt. Bei der Interpolation stehen Informationen an *mehreren* Punkten zur Verfügung. Typischerweise möchte man eine kontinuierliche Funktion p finden, welche $n+1$ vorgegebene Werte $f_i = f(x_i)$ an $n+1$ verschiedenen Stellen x_i exakt annimmt. Bei der Polynom-Interpolation sucht man dann ein Polynom $p(x)$ mit dieser Eigenschaft.

Das Polynom $p(x)$, das die $n+1$ Bedingungen

$$p(x_i) = f_i, \qquad i = 0, \ldots n, \tag{3.86}$$

erfüllt, wird *Interpolationspolynom* genannt. Für $n = 1$ wissen wir, daß durch zwei verschiedene vorgegebene Punkte (bei x_0 und x_1) genau eine Ausgleichsgerade

$$p(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0) \tag{3.87}$$

gelegt werden kann. Dies ist ein Polynom ersten Grades.

Um die Bedingungen (3.86) im allgemeine Fall von $n+1$ Punkten zu erfüllen, benötigt man ein Polynom vom Grade n und macht daher den Ansatz

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n. \tag{3.88}$$

Die $n+1$ unbekannten Koeffizienten a_i kann man mit Hilfe der Bedingungen (3.86) finden. Die Bedingungen

$$p(x_i) = a_0 + a_1 x_i + \ldots + a_n x_i^n \stackrel{!}{=} f_i, \qquad i = 0, \ldots n, \tag{3.89}$$

kann man als ein lineares Gleichungssystem für den Vektor der unbekannten Koeffizienten a_i schreiben und erhält so

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \tag{3.90}$$

Die blaue Koeffizienten-Matrix in (3.90) wird *Vandermondesche Matrix* genannt. Wenn die Vandermondesche Matrix regulär ist (siehe Anhang A.4.4), d.h. wenn die Spalten (bzw. Zeilen) linear unabhängig voneinander sind, dann existiert eine eindeutige Lösung und das Interpolationspolynom kann eindeutig bestimmt werden. In der Praxis gibt es jedoch effizientere Methoden, das Interpolationspolynom zu erhalten.[1]

[1] In MATLAB kann man mit dem Befehl `A=vander(x)` eine $n \times n$ Vandermondsche Matrix erzeugen, wobei `x` ein Vektor der Länge n ist.

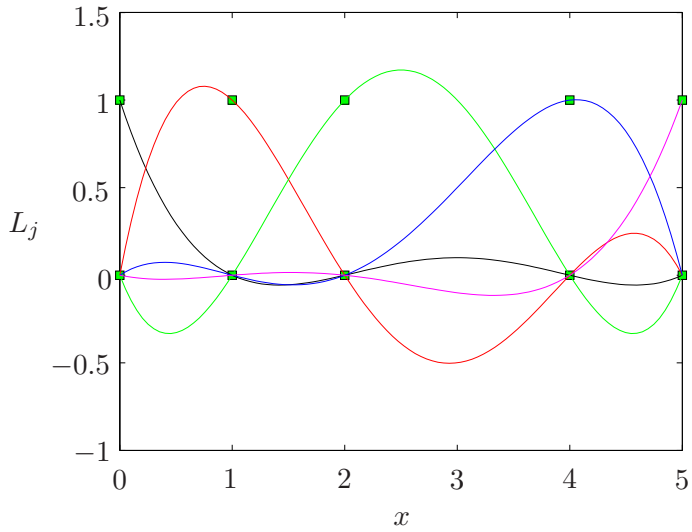


Abbildung 3.3.: Die 5 Lagrange-Polynome $L_j(x)$ vierter Ordnung zu den 5 Punkten $x_j \in \{0, 1, 2, 4, 5\}$. Jedes Polynom ist Eins am betreffenden Knoten und verschwindet an allen anderen Knoten.

3.2.1. Lagrange-Interpolation

Bei der Methode der finiten Elemente für stuktur- oder strömungsmechanische Berechungen wird die unbekannte Funktion F (gesuchtes Verrückungs- oder Geschwindigkeitsfeld) häufig als Überlagerung (gewichtete Summe) von gewissen fest vorgegebenen Ansatzfunktionen $\phi_j(x)$ dargestellt. Im eindimensionalen Fall macht man den Ansatz

$$F(x) = \sum_{j=0}^{n} f_j \phi_j(x). \tag{3.91}$$

Hierbei ist jede der Ansatzfunktionen $\phi_j(x)$ einem vorgegebene Gitterpunkt (Knoten) x_j zugeordnet. Es wäre nun schön, wenn die Lösung $F(x)$ an den Knotenpunkten x_j genau durch die Koeffizienten f_j in (3.91) repräsentiert würde: $F(x_j) = f_j$. Um dies zu erreichen, ist es zweckmäßig, Ansatzfunktionen zu verwenden, für welche $\phi_j(x_j) = 1$ und $\phi_j(x_k) = 0$ für $k \neq j$. Für die Ansatzfunktion $\phi_j(x)$ sollte also gelten $\phi_j(x_k) = \delta_{j,k}$. Funktionen ϕ_j mit dieser Eigenschaft lassen sich mit Hilfe von Polynomen konstruieren. Für paarweise verschiedene[2] Punkte $x_0, x_1, \ldots, x_n$ definiert man für jeden Punkt x_j ein zugehöriges *Lagrange-Polynom* $L_j(x)$

$$L_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}, \qquad j = 0, 1, \ldots, n. \tag{3.92}$$

[2] D.h. alle x-Werte müssen verschieden sein. Beispielsweise sind die Punkte $\{1, 2, 3, 3, 4\}$ nicht paarweise verschieden.

Wenn man das Produkt ausschreibt

$$\prod_{\substack{k=0\\k\neq j}}^{n} \frac{x-x_k}{x_j-x_k} = \frac{(x-x_0)\ldots(x-x_{j-1})\,\textvisiblespace\,(x-x_{j+1})\ldots(x-x_n)}{(x_j-x_0)\ldots(x_j-x_{j-1})\,\textvisiblespace\,(x_j-x_{j+1})\ldots(x_j-x_n)}, \tag{3.93}$$

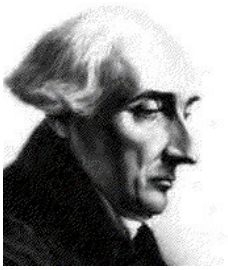
sieht man leicht, daß L_j an allen Stützpunkten $x = x_k$ mit $k \neq j$ verschwindet und nur am Punkt $x = x_j$ gleich eins ist. Die Lagrange-Polynome besitzen also die gesuchte Eigenschaft

$$L_j(x_k) = \begin{cases} 1 & x_k = x_j, \\ 0 & \text{sonst.} \end{cases} \tag{3.94}$$

Entweder ist in (3.92) ein Faktor gleich Null, oder alle Faktoren kürzen sich zu 1. Die Lagrange-Polynome sind im allgemeinen vom Grade n. Als Beispiel sind in Abb. 3.3 die Langrange-Polynome $L_j(x)$ zu den 5 Punkten $x_j \in \{0, 1, 2, 4, 5\}$ gezeigt.

Mit den Eigenschaften der einzelnen Lagrange-Polynome definiert man nun das *Lagrangesche Interpolationspolynom* als

$$p(x) := \sum_{j=0}^{n} f_j\, L_j(x). \tag{3.95}$$



Joseph-Louis Lagrange 1736–1813

Das Polynom $p(x)$ interpoliert die an den Knotenpunkten x_j vorgegebenen Werte f_j. Für jeden der vorgegebenen Punkte x_j verschwinden alle Lagrange-Polynome L_j, bis auf ein einziges, welches den Wert 1 annimmt. Dieses wird mit dem Wert f_j multipliziert. Damit sind die $n+1$ Bedingungen (3.86) erfüllt.

Wenn man also in (3.91) die Langrange-Polynome (3.92) als Ansatzfunktionen verwendet, kann man die Lösung $F(x) = p(x)$ als Polynom darstellen. Bei der Methode den finiten Elementen verwendet man meist eine recht niedrige Ordnung (erste oder zweite Ordnung) für die Ansatzfunktionen. Dabei werden nur die (wenigen) Knoten verwendet, die sich in einem gewissen Raumgebiet (Element) befinden. Das gesamte Rechengebiet besteht dann aus einer sehr großen Anzahl von Elementen.

Ein Beispiel ist in Abb. 3.4 gezeigt. Hier wurde die Funktion $f = \tanh(x)$ durch 14 äquidistante Stützstellen im Intervall $x \in [-6, 6]$ mittels (3.95) interpoliert. Man erkennt, daß die Fehler im Randbereich relativ groß werden. Dies hängt damit zusammen, daß jeder Funktionswert f_i an einem einzelnen Punkt x_i einen *globalen Einfluß* auf den gesamten Kurvenverlauf hat.

3.2.2. Newton-Interpolation

Die Lagrange-Interpolation (3.95) ist nicht komfortabel, wenn die Anzahl der gegebenen Wertepaare (x_j, f_j) nicht konstant ist. Denn die Lagrangeschen Polynome (3.92) hängen von allen $n+1$ Punkten ab. Deshalb müssen sie allesamt neu gebildet werden, wenn sich die Punktzahl ändert.

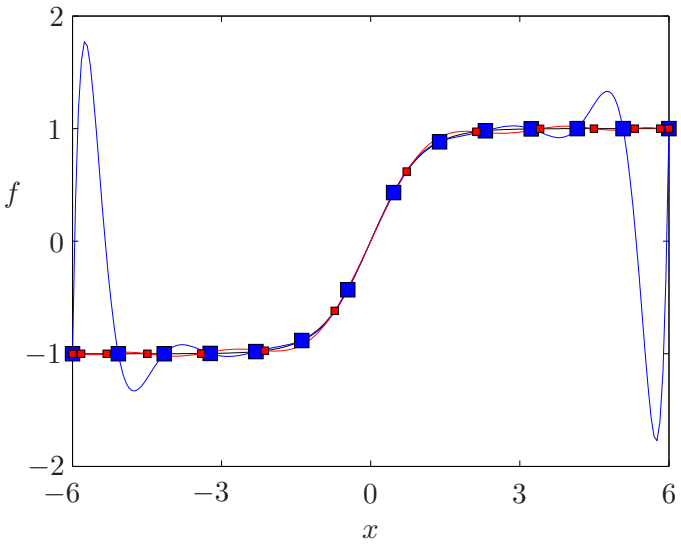
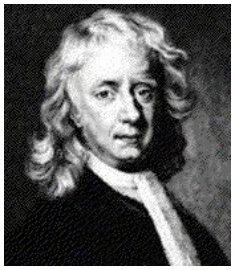


Abbildung 3.4.: Interpolation von $f = \tanh(x)$ (schwarz) mittels Lagrangescher Interpolation (blau) nach (3.95) an 14 äquidistanten Stützstellen x_i (blau). Man erkennt die schlechte Darstellung im Randbereich. Durch Verwendung von Chebyshev-Knoten (3.107) (rot) lassen sich die Oszillationen stark reduzieren.



Sir Isaac Newton
1642–1727

Eine Alternative stellt die *Newton-Interpolation* dar. Sie liefert ein Interpolationspolynom, das mit dem Lagrangeschen Interpolationspolynom (3.95) identisch ist. Bei der Newtonschen Formulierung läßt sich die Anzahl der Punkte sehr einfach erweitern. Voraussetzung für die Newton-Interpolation ist jedoch eine *äquidistante* Anordnung der Punkte im Abstand $x_{j+1} - x_j = h = \text{const}$.

Die Konstruktion des Newtonschen Interpolationspolynoms erfolgt folgendermaßen. Wir definieren die *finiten Differenzen* in Vorwärtsrichtung

$$\Delta f_j := f_{j+1} - f_j. \tag{3.96}$$

Finite Differenzen höherer Ordnung werden definiert als

$$\Delta^k f_j := \Delta\left(\Delta^{k-1} f_j\right) = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j. \tag{3.97}$$

Dann ergeben sich die Differenzen höherer Ordnung des ersten Funktionswertes f_0 als

$$\Delta f_0 = f_1 - f_0, \tag{3.98a}$$
$$\Delta^2 f_0 = \Delta\left(\Delta f_0\right) = \Delta f_1 - \Delta f_0 = f_2 - 2f_1 + f_0, \tag{3.98b}$$
$$\Delta^3 f_0 = \Delta^2 f_1 - \Delta^2 f_0 = \Delta f_2 - 2\Delta f_1 + \Delta f_0 = f_3 - 3f_2 + 3f_1 - f_0, \tag{3.98c}$$
$$\vdots$$
$$\Delta^n f_0 = f_n - \binom{n}{1} f_{n-1} + \binom{n}{2} f_{n-2} - \ldots + (-1)^n f_0. \tag{3.98d}$$

Hierbei sind

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\ldots(n-k+1)}{k!} \tag{3.99}$$

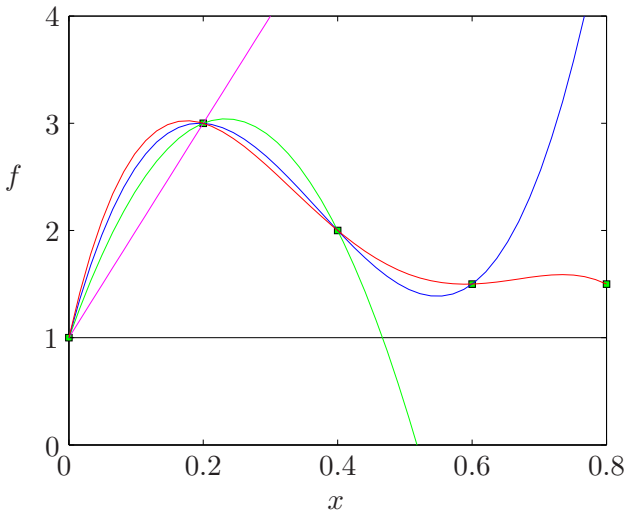


Abbildung 3.5.: Sukzessive Newton-Interpolation von fünf äquidistanten Punkten im Abstand $h = 0.2$.

die gewöhnlichen *Binomialkoeffizienten*.

Mit den soeben definierten Vorwärtsdifferenzen lautet das *Newtonsche Interpolationspolynom*

$$p_n = f_0 + \frac{x - x_0}{h}\Delta f_0 + \frac{(x - x_0)(x - x_1)}{2h^2}\Delta^2 f_0 + \ldots + \frac{(x - x_0)\ldots(x - x_{n-1})}{n!\, h^n}\Delta^n f_0. \tag{3.100}$$

Um zu beweisen, daß $p_n(x)$ nach (3.100) tatsächlich die Interpolationsbedingungen (3.86) erfüllt, prüft man zunächst leicht, daß $p_n(x_0) = f_0$, da alle anderen Summanden in (3.100) an der Stelle x_0 verschwinden. Dann betrachtet man $p_n(x_1)$

$$p_n(x_1) = f_0 + \underbrace{\frac{x_1 - x_0}{h}}_{=1}\Delta f_0 = f_1, \tag{3.101}$$

und so weiter. Den Beweis kann man durch Induktion führen (Freund et al., 2007). Das Newtonsche Interpolationspolynom (3.100) entspricht einer Taylorentwicklung um x_0, wobei die diskreten Ableitungen am Punkt x_0 mittels finiter Vorwärtsdifferenzen berechnet werden $\Delta f_0/h$, $\Delta^2 f_0/h^2$, etc.

Bei Erhöhung der Anzahl der Stützpunkte ergibt sich das Newton-Polynom einfach durch Hinzunahme weiterer Summanden bzw. höherer Differenzen. Wir können (3.100) daher leicht auf p_{n+1} erweitern

$$p_{n+1}(x) = p_n(x) + \frac{(x - x_0)\ldots(x - x_n)}{(n+1)!\, h^{n+1}}\Delta^{n+1} f_0. \tag{3.102}$$

Diese Flexibilität ist manchmal von Vorteil. Man sieht, daß der $(n+1)$-te Summand an den Stützstellen bei $x_0, \ldots, x_n$ verschwindet und deshalb die Funktionswerte an diesen Stellen nicht verändert. In Abb. 3.5 ist ein Beispiel gezeigt.

Da man durch $n + 1$ Punkte nur ein Polynom vom Grade n legen kann, ist das Interpolationpolynom eindeutig festgelegt. Daher sind die Newtonschen Interpolationspolynome identisch mit den Lagrangeschen Interpolationspolynomen!

3.2.3. Eindeutigkeit der Polynom-Interpolation und Verallgemeinerung

Die bisher beschriebenen drei Verfahren zur Erzeugung eines Polynoms vom Grade n (Vandermonde, Lagrange, Newton), welches $n+1$ vorgegebene Punkte genau reproduziert, liefern ein und dasselbe Interpolationspolynom, nur in einer jeweils anderen Darstellung.

> Zu $n+1$ beliebigen paarweise verschiedenen Punkten $x_0, \ldots, x_n$ und zugeordneten Werten $f_0, \ldots, f_n$ existiert ein *eindeutiges* Polynom vom Höchstgrade n, welches genau durch diese Punkte geht, d.h., welches (3.86) erfüllt.

Diesen Satz kann man durch Widerspruch beweisen (siehe z.B. Golub and Ortega, 1996): Angenommen es gäbe zwei Interpolationspolynome $p(x)$ und $q(x)$. Dann ist auch $r = p - q$ ein Polynom vom Höchstgrade n, welches an den $n+1$ Stützstellen x_i gerade seine Nullstellen hat. Aus dem Fundamentalsatz der Algebra[3] ergibt sich dann, dass $r \equiv 0$ das Nullpolynom ist, womit $p = q$. Damit ist auch der Beweis erbracht, daß die Vandermondesche Matrix (3.90) regulär ist (eindeutige Lösung), wenn die Punkte $x_0, \ldots, x_n$ paarweise verschieden sind.

Alle bisher betrachteten Interpolationen waren von der Form

$$p(x) = a_0\phi_0(x) + \ldots + a_n\phi_n(x), \tag{3.103}$$

wobei die $\{\phi_j(x)\}$ lediglich verschiedene *Basisfunktionen* sind. In den obigen Beispielen bestanden die Basisfunktionen aus verschiedenen Sätzen von Polynomen vom Höchstgrade n. Dies waren $\phi_j = x^j$ im Standardfall (Vandermonde), $\phi_j = L_j(x)$ bei der Lagrange-Interpolation und $\phi_j = (x-x_0)\ldots(x-x_j)$ bei der Newton-Interpolation.

Auch andere Polynome können verwendet werden, zum Beispiel die bekannten *Chebyshev-Polynome* (siehe Kap. 4.5.3). Hierfür benötigt man die Funktionswerte an ganz bestimmten Stützstellen eines festen Intervalls. Bei *Hermite-Polynomen* kann man zusätzlich neben den Funktionswerten auch verlangen, daß die ersten Ableitungen an den Stützstellen vorgegebene Werte annehmen. Manchmal ist es auch günstiger keine Polynome, sondern andere Funktionensysteme für die Interpolation zu verwenden, z.B. Exponentialfunktionen $\phi_j = \mathrm{e}^{\alpha_j x}$ (α_j fest vorgegeben) oder trigonometrische Funktionen $\sin(\alpha_i \pi x)$.

Unter Verwendung allgemeiner Basisfunktionen $\{\phi_j(x)\}$ lautet die Interpolationsbedingung offenbar

$$a_0\phi_0(x_i) + \ldots + a_n\phi_n(x_i) = f_i, \qquad i = 0, \ldots n, \tag{3.104}$$

[3] Der Gaußsche Fundamentalsatz der Algebra, spezialisiert auf Polynome mit reellen Koeffizienten, lautet: Jedes reelle Polynom $r(x)$ läßt sich in reelle Polynomfaktoren vom Grade eins oder zwei zerlegen, z.B. $r = (x-a)(x^2+b)$. Eine Polynom vom Grade n mit reellen Nullstellen kann dann nur maximal n verschiedene Nullstellen besitzen, nicht aber $n+1$. Das einzige Polynom vom Grade n mit $n+1$ Nullstellen ist das Nullpolynom.

was in Analogie zu (3.90) auf das Gleichungssystem

$$\mathsf{A}\cdot\vec{a} = \begin{pmatrix} \phi_0(x_0) & \phi_1(x_0) & \phi_2(x_0) & \dots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_n(x_1) \\ \vdots & & & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_n(x_n) \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix} \tag{3.105}$$

führt. Auch hier muß man für die Existenz und Eindeutigkeit der Lösung (der Interpolation) fordern, daß die Koeffizienten-Matrix regulär ist, d.h. $\det \mathsf{A} \neq 0$.

Bei der polynomialen Interpolation hat jeder Wert f_i einen globalen Einfluß auf das Verhalten des Interpolationspolynoms. Dies kann zu großen Fehlern führen, insbesondere, wenn die zu approximierende Funktion lokal stark variiert.

Ähnlich wie für das Restglied bei der Taylorreihe kann man den Fehler bei einer Polynom-Approximation präzisieren (ohne Beweis).

Fehler bei der Polynominterpolation: Im Falle einer zu interpolierenden Funktion $f(x)$ kann der Fehler durch Polynominterpolation angegeben werden. Es sei $f(x)$ auf dem Intervall, das die Punkte $x_0, \dots, x_n$ enthält, $(n+1)$-mal differenzierbar. Wenn $p_n(x)$ das eindeutige Interpolationspolynom vom Höchstgrade n ist, das (3.86) erfüllt, dann lautet der Fehler

$$f(x) - p_n(x) = \frac{(x-x_0)\dots(x-x_n)}{(n+1)!} f^{(n+1)}(z), \tag{3.106}$$

wobei $z \in [x_0, x_n]$ eine von x abhängige Zwischenstelle ist.

Wenn die $(n+1)$-te Ableitung $f^{(n+1)}$ an irgendeiner Stelle groß wird, kann demnach an einer anderen Stelle des Intervalls ein sehr großer Fehler auftreten. Diese unter Umständen ungünstige Eigenschaft hängt mit dem oben angesprochenen globalen Einfluß eines jeden Punktes zusammen. Daher erscheint es wünschenswert, zur Interpolation einer Funktion an einer Stelle x auch nur Funktionswerte an benachbarten Stützstellen zu verwenden. Dieses Konzept werden wir in Kap. 3.3 verfolgen.

Carl David Tolmé Runge 1856–1927

3.2.4. Lagrange-Interpolation auf Chebyshev-Stützpunkten

In Kap. 3.2.1 hatten wir gesehen, daß es am Rand des Gebiets der äquidistanten Lagrange-Interpolation zu übermäßig starken Oszillationen des Interpolationspolynoms kommt (Abb. 3.4). Dies wird auch als *Runge-Phänomen* bezeichnet. Die Oszillationen am Rand können vermieden werden, wenn man die Interpolationspunkte zum Rand hin verdichtet.

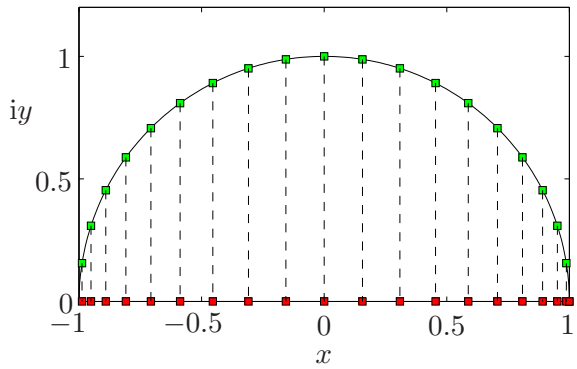


Abbildung 3.6.: Chebyshev-Punkte (rot) als Projektion gleichverteilter Punkte (grün) auf dem halben Einheitskreis für $i = 0, 1, \ldots, 20$.

Eine Möglichkeit der Verdichtung besteht in der Verwendung sogenannter *Chebyshev-Knoten*. Die Verteilung der Punkte ergibt sich, wenn man die $n + 1$ Interpolationspunkte äquidistant auf dem halben Einheitskreis in der komplexen Ebene verteilt, die Punkte auf die reelle x-Achse projiziert und dann das Intervall $[a, b]$, über welchem interpoliert wird, dem Durchmesser $[-1, 1]$ des Einheitskreises anpaßt (Abb. 3.6). Man erhält damit die Chebyshev-Knoten (mit den Randpunkten)

$$x_i = \frac{a+b}{2} - \frac{b-a}{2} \cos\left(\frac{i}{n}\pi\right), \qquad i = 0, 1, \ldots, n. \tag{3.107}$$

Wie die rote Kurve in Abb. 3.4 zeigt, werden mit dieser Punktwahl die Oszillationen am Rand stark unterdrückt.[4]

3.3. Splines

Trotz Verwendung von Chebyshev-Punkten kann das Runge-Phänomen nicht immer unterdrückt werden. Um diese Nachteile der Interpolation mit Polynomen hohen Grades zu vermeiden, kann man zu einer stückweisen Interpolation niedrigen Grades übergehen. Diese Art der Interpolation findet vielfache Anwendung im CAD-Bereich.

Eine einfache Interpolation, die nur Funktionswerte an benachbarten Stützstellen verwendet, ist die stückweise lineare Approximation. Ein Beispiel ist in Abb. 3.7 gezeigt. Dieser Ansatz kann verallgemeinert werden. Dabei wird das gesamte Intervall in Teilintervalle unterteilt, die jeweils dieselbe geringe Anzahl von Punkten enthalten. Dann wird nur innerhalb eines jeden Teilintervalls separat interpoliert.

[4] Eine alternative Chebyshev-Verteilung mit denselben Konvergenzeigenschaften ist (ohne Randpunkte)

$$x_i = \frac{a+b}{2} - \frac{b-a}{2} \cos\left[\frac{(2i+1)}{2n+2}\pi\right], \qquad i = 0, 1, \ldots, n.$$

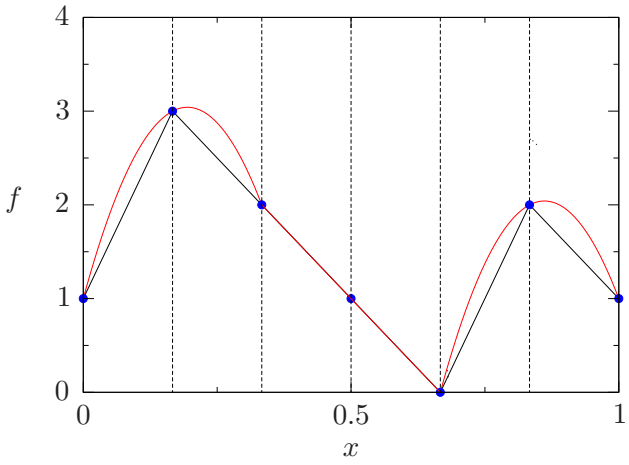


Abbildung 3.7.: Stückweise lineare (schwarz) und stückweise quadratische (rot) Approximation der sechs Punkte $(x, f) = (0, 1), (1/6, 3), (1/3, 2), (0.5, 1), (2/3, 0), (5/6, 2)$ und $(1, 1)$ (blaue Punkte).

Normalerweise besitzt die stückweise Interpolation eine sehr niedrige Ordnung, abhängig von der Punkteanzahl eines jeden Teilintervalls.

Punkte außerhalb eines Intervalls haben deshalb keinen Einfluß auf die lokale Approximation. Ein großer Nachteil der stückweisen Interpolation besteht darin, daß die Interpolation i.a. nicht glatt ist. D.h., sie ist an den Grenzen der Teilintervalle i.a. nicht differenzierbar.

3.3.1. Quadratische Splines

Damit die stückweise Polynomapproximation differenzierbar wird, muß man verlangen, daß die rechts- und linksseitigen Ableitungen an den Stützpunkten und Intervallgrenzen identisch sind.

Wir betrachten die Interpolation, bei der jedes Intervall nur zwei Stützpunkte enthält. Um die genannten Zusatzbedingungen zu erfüllen, setzen wir die Interpolationsfunktion jedoch nicht linear sondern quadratisch an. Als Beispiel betrachten wir die quadratische Interpolation zwischen den vier Punkten (x_1, x_2, x_3, x_4). Auf jedem der drei Intervalle $[x_i, x_{i+1}]$ definieren wir dann die quadratischen Funktionen

$$q_i(x) = a_{i2}x^2 + a_{i1}x + a_{i0}, \qquad i = 1, 2, 3. \tag{3.108}$$

Damit die Bedingung $q_i(x_i) = f_i$ (linker Randpunkt des i-ten Intervalls) und $q_i(x_{i+1}) = f_{i+1}$ (rechter Randpunkt des i-ten Intervalls) erfüllt ist, müssen wir fordern

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ q_1(x_1) = f_1, & q_1(x_2) = f_2, & q_2(x_3) = f_3, & \end{array} \tag{3.109a}$$

$$\begin{array}{ccc} q_2(x_2) = f_2, & q_3(x_3) = f_3, & q_3(x_4) = f_4. \end{array} \tag{3.109b}$$

Dies sind $2 \times 3 = 6$ Bedingungen. Damit die Ableitungen an den beiden inneren Intervallgrenzen x_2 und x_3 stetig sind, müssen auch noch die zwei weiteren Bedingungen

$$q_1'(x_2) = q_2'(x_2), \qquad q_2'(x_3) = q_3'(x_3), \tag{3.110}$$

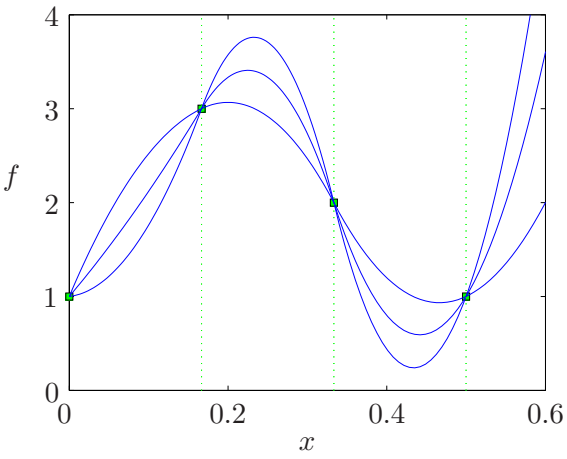


Abbildung 3.8.: Stückweise quadratische Interpolation der ersten 4 Punkte aus Abb. 3.7 für die Anfangssteigungen $q_1'(x_1) = 1$, 10 und 20.

erfüllt sein. Insgesamt müssen also 8 Bedingungen erfüllt werden. Dies ist möglich, da wir nach (3.108) insgesamt $3 \times 3 = 9$ Koeffizienten zur Verfügung haben. Um die Interpolation eindeutig zu bestimmen, haben wir sogar noch eine weitere Bedingung frei. Diese können wir verwenden, um die Anfangssteigung festzulegen: $q_1'(x_1) = d_1$. Die resultierenden 9 Bedingungen ergeben ein lineares System von 9 Gleichungen für die 9 Unbekannten a_{ij}. In Abb. 3.8 ist ein Beispiel gezeigt, in dem die ersten 4 Punkte aus Abb. 3.7 stückweise quadratisch interpoliert wurden.

Wenn man diese Betrachtungen auf n Knoten erweitert, hat man $n-1$ Interpolationsfunktionen, d.h. $3(n-1)$ Unbekannte a_{ij} zu bestimmen. Die Stetigkeit der stückweisen Interpolationen liefert $2(n-1)$ Bedingungen und die stetige Differenzierbarkeit an den inneren Punkten weitere $n-2$. Man hat also insgesamt $2(n-1)+(n-2) = 3n-4 = 3(n-1)-1$ Bedingungen. Im allgemeinen Fall bleibt also immer eine Bedingung frei, die man zu Festlegung der Anfangssteigung $q_1'(x_1)$ verwenden kann.

3.3.2. Kubische Splines

Bei vielen physikalischen Problemen reicht die einfache Differenzierbarkeit nicht aus. Man möchte lieber eine zweifach-differenzierbare Approximationen haben. Diese Forderung führt auf die *kubische Spline-Interpolation*. Die kubischen Splines stellen eine sehr wichtige und vielfach verwendete Klasse von Interpolationspolynomen dar.

Sei die Funktion $f(x)$ an den n Stützstellen x_i durch $f_i = f(x_i)$, $i = 1, ..., n$, vorgegeben. Die Interpolationsfunktion im Intervall $x \in [x_i, x_{i+1}]$ setzen wir dann als Polynom dritten Grades an

$$P_i(x) = f_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 1, ..., n-1. \tag{3.111}$$

Wir müssen also insgesamt $3(n-1)$ Koeffizienten (b_i, c_i, d_i), $i = 1, ..., n-1$ bestimmen.

Per constructionem gilt am linken Rand eines jeden Intervalls $P_i(x_i) = f_i$. Die drei freien Koeffizienten b_i, c_i und d_i werden so bestimmt, daß das Interpolationspolynom an allen inneren Intervallgrenzen zweifach stetig differenzierbar ist. Für

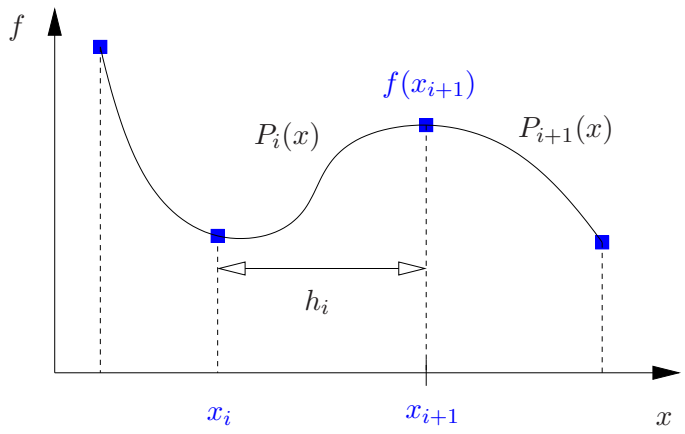


Abbildung 3.9.: Notation bei der kubischen Spline-Interpolation (Beispiel: `xfig`).

die inneren Punkte bedeutet dies (Abb. 3.9)

$$\begin{aligned}
&\text{Funktion stetig:} && P_i(x_{i+1}) = f_{i+1}, && i = 1, ..., n-2 && (3.112a)\\
&\text{1. Ableitung stetig:} && P_i'(x_{i+1}) = P_{i+1}'(x_{i+1}), && i = 1, ..., n-2 && (3.112b)\\
&\text{2. Ableitung stetig:} && P_i''(x_{i+1}) = P_{i+1}''(x_{i+1}), && i = 1, ..., n-2. && (3.112c)
\end{aligned}$$

Dies sind erst $3(n-2)$ Bedingungen. Hinzu kommt noch die Bedingung für den rechten Randpunkt $P_{n-1}(x_n) = f_n$. Es bleiben also noch zwei Bedingungen frei, die man an die gesuchten Koeffizienten stellen kann.

Sei $h_i = x_{i+1} - x_i$ der Abstand der Stützstellen. Mit $P_i''(x) = 2c_i + 6d_i(x - x_i)$ folgt dann aus (3.112c)

$$2c_i + 6d_i \underbrace{(x_{i+1} - x_i)}_{=h_i} = 2c_{i+1} + 6d_{i+1} \underbrace{(x_{i+1} - x_{i+1})}_{=0}$$

bzw.

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad i = 1, ..., n-2. \tag{3.113}$$

Wenn man dies in (3.112a) einsetzt, erhalten wir

$$f_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = f_i + b_i h_i + c_i h_i^2 + \frac{c_{i+1} - c_i}{3} h_i^2 = f_{i+1}.$$

Aufgelöst nach b_i ergibt sich

$$b_i = \frac{f_{i+1} - f_i}{h_i} - h_i \frac{c_{i+1} + 2c_i}{3}, \quad i = 1, ..., n-2. \tag{3.114}$$

Beachte, daß in (3.113) und (3.114) die für das letzte Intervall benötigten Koeffizienten d_{n-1} und b_{n-1} nicht enthalten sind. Um diese Werte mit einzuschließen, erhöhen wir den Laufindex i für beide Gleichungen um eins bis auf $n-1$ mit dem Verständnis, daß die in den Gleichungen für $i = n-1$ dann auftauchende Größe c_n eine noch zu bestimmende Hilfsgröße ist.

Aus (3.112b) erhalten wir für $i = 1, \ldots, n-2$

$$P_i'(x_{i+1}) = b_i + 2c_i \underbrace{(x_{i+1} - x_i)}_{h_i} + 3d_i \underbrace{(x_{i+1} - x_i)^2}_{h_i^2} \stackrel{!}{=} P_{i+1}'(x_{i+1}) = b_{i+1}. \tag{3.115}$$

Für $i = n-2$ taucht auf der rechten Seite die Unbekannte b_{n-1} auf, die wir durch die Hilfsgröße c_n ausgedrückt haben. Zusammen mit (3.113) für d_i und (3.114) für b_i erhalten wir formal die Gleichungen zur Bestimmung von c_i

$$h_i c_i + 2\,(h_i + h_{i+1})\,c_{i+1} + h_{i+1} c_{i+2} = 3\left(\frac{f_{i+2} - f_{i+1}}{h_{i+1}} - \frac{f_{i+1} - f_i}{h_i}\right), \quad i = 1, ..., n-2, \tag{3.116}$$

die auch die Hilfsgröße c_n einschließt. Dies ist im Prinzip ein tridiagonales System von $n-2$ Gleichungen für die Unbekannten c_1 bis c_n in der Form

$$\begin{bmatrix} \times & \times & \times & & & & & \\ & \times & \times & \times & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \times & \times & \times & \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ \\ \vdots \\ \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} g_1 \\ \\ \vdots \\ \\ g_{n-2} \end{bmatrix} \tag{3.117}$$

Da wir nur eine $(n-2) \times n$-Matrix für n Unbekannte haben, ist das System zweifach unterbestimmt. Insgesamt haben wir aber mit $n-2$ Gleichungen (3.117) für $c_{1,\ldots,n}$ und jeweils $n-1$ Gleichungen (3.113) (für $d_{1,\ldots,n-1}$) und (3.114) (für $b_{1,\ldots,n-1}$) insgesamt $3(n-2)+2 = 3(n-1)-1$ Gleichungen. Eine weitere Gleichung erhalten wir noch aus der schon oben erwähnten Bedingung $P_{n-1}(x_n) = f_n$ am rechten Rand

$$P_{n-1}(x_n) = f_{n-1} + b_{n-1} h_{n-1} + c_{n-1} h_{n-1}^2 + d_{n-1} h_{n-1}^3 = f_n, \tag{3.118}$$

womit wir auf die erforderlichen $3(n-1)$ Gleichungen kommen.

Da (3.117) aber zweifach unterbestimmt ist, haben wir noch 2 Bedingungen frei, um das System zu schließen. Zu diesem Zweck kann man beispielsweise die Steigungen von f an den beiden Rändern vorgeben, d.h. $P_1'(x_1) = f_1'$ und $P_{n-1}'(x_n) = f_n'$, die Steigung f_1' und die Krümmung f_1'' an nur einem Rand, oder höhere Ableitungen im Innern.

MATLAB stellt eine Spline-Interpolation in Form von `y=spline(x,f,r)` zur Verfügung, wobei die Vektoren `x` und `f` die Stützstellen und die zu interpolierenden Daten enthalten und `r` der Vektor der Stellen ist, an denen die Interpolationsfunktion `y` berechnet werden soll.[5]

[5] Der MATLAB-Befehl `y=spline(x,f,r)` verwendet die Zusatzbedingungen, daß die dritten Ableitungen an den Stellen x_2 und x_{n-1} stetig sind: $f_1'''(x_2) = f_2'''(x_2)$ und $f_{n-2}'''(x_{n-1}) = f_{n-1}'''(x_{n-1})$.

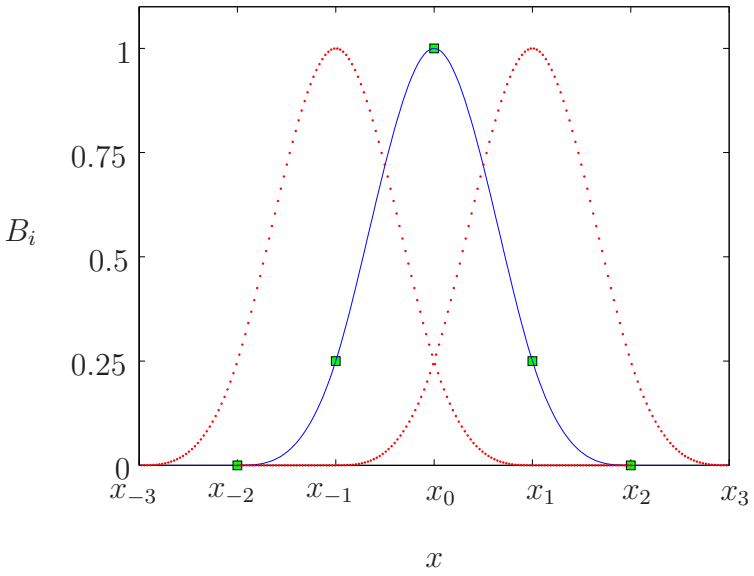


Abbildung 3.10.: B-Splines B_0 (blau) und B_{-1} sowie B_1 (beide rot gepunktet). Die Werte 0, 1/4 und 1 an den fünf Stützstellen von $B_0(x)$ sind grün markiert.

Der Begriff *Spline* stammt aus dem Schiffbau. Eine lange dünne Latte (Straklatte), die an einzelnen Punkten durch Nägel fixiert wird, biegt sich genau wie ein kubischer Spline mit natürlichen Randbedingungen ($f_1'' = f_n'' = 0$). Die Form ergibt sich daraus, daß die Latte bestrebt ist, die inneren Spannungen zu minimieren.

3.3.3. Kubische B-Splines

Einen anderen einfachen Zugang zu kubischen Spines erhält man über die sogenannten *basic splines* oder *B-Splines*, wenn man eine homogene Schrittweite $h_i = h = \text{const.}$ wählt. Dazu definiert man eine Menge von kubischen Splines, die allesamt dieselbe Kontur besitzen. Sie sind nur auf einer Länge von $4h$ von Null verschieden und gegeneinander verschoben (Abb. 3.10). Die kubischen B-Splines sind stückweise definiert als

$$B_i(x) = \begin{cases} \dfrac{1}{4h^3}\left(x - x_{i-2}\right)^3, & x_{i-2} \le x \le x_{i-1}, \\ \dfrac{1}{4} + \dfrac{3}{4h}\left(x - x_{i-1}\right) + \dfrac{3}{4h^2}\left(x - x_{i-1}\right)^2 - \dfrac{3}{4h^3}\left(x - x_{i-1}\right)^3, & x_{i-1} \le x \le x_i, \\ \dfrac{1}{4} + \dfrac{3}{4h}\left(x_{i+1} - x\right) + \dfrac{3}{4h^2}\left(x_{i+1} - x\right)^2 - \dfrac{3}{4h^3}\left(x_{i+1} - x\right)^3, & x_i \le x \le x_{i+1}, \\ \dfrac{1}{4h^3}\left(x_{i+2} - x\right)^3, & x_{i+1} \le x \le x_{i+2}, \\ 0, & \text{sonst.} \end{cases} \tag{3.119}$$

Zur Bestimmung von B_i stehen für die kubischen Funktionen in den 4 Intervallen $4 \times 4 = 16$ Koeffizienten zur Verfügung. Die Stetigkeit an den drei inneren Punkten, die beiden Randwerte 0 und die Normierung des Maximums auf 1 liefern $3+2+1 = 6$ Bedingungen. Dazu kommen noch $2 \times 5 = 10$ Bedingungen für die Stetigkeit der ersten und zweiten Ableitungen an den 5 Punkten (grün in Abb. 3.10).

Einen beliebigen kubischen Spline $c(x)$ kann man nun als Superposition kubischer B-Splines ansetzt. Dies ergibt die Darstellung

$$c(x) = \sum_{j=1}^{n} a_j B_j(x). \tag{3.120}$$

Die n unbekannten Koeffizienten a_j lassen sich mit Hilfe der n Interpolationsbedingungen

$$c(x_i) = f_i, \quad i = 1, \ldots, n \tag{3.121}$$

bestimmen. Wenn man beachtet, daß $B_j(x_i) = 0$ ist für $|i-j| \geq 2$ (nur die nächsten Nachbarn von x_i tragen etwas bei), erhält man durch Einsetzen von (3.120) in die Bedingungen (3.121) das Gleichungssystem

$$\begin{aligned}
a_1 B_1(x_1) + a_2 B_2(x_1) &= f_1, \\
a_1 B_1(x_2) + a_2 B_2(x_2) + a_3 B_3(x_2) &= f_2, \\
a_2 B_2(x_3) + a_3 B_3(x_3) + a_4 B_4(x_3) &= f_3, \\
&\vdots \\
a_{n-2} B_{n-2}(x_{n-1}) + a_{n-1} B_{n-1}(x_{n-1}) + a_n B_n(x_{n-1}) &= f_{n-1}, \\
a_{n-1} B_{n-1}(x_n) + a_n B_n(x_n) &= f_n.
\end{aligned} \tag{3.122}$$

Dies kann man auch in Matrix-Form schreiben als

$$\begin{pmatrix}
B_1(x_1) & B_2(x_1) & & & & & \\
B_1(x_2) & B_2(x_2) & B_3(x_2) & & & & \\
 & B_2(x_3) & B_3(x_3) & B_4(x_3) & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & B_{n-2}(x_{n-1}) & B_{n-1}(x_{n-1}) & B_n(x_{n-1}) \\
 & & & & B_{n-1}(x_n) & B_n(x_n)
\end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \\ \vdots \\ \\ a_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \\ \vdots \\ \\ f_n \end{pmatrix}. \tag{3.123}$$

Mit $B_i(x_i) = 1$ und $B_{i\pm1}(x_i) = 1/4$ erhalten wir das einfache tridiagonale System (vergleiche auch (3.116))

$$\frac{1}{4} \begin{pmatrix}
4 & 1 & & & \\
1 & 4 & 1 & & \\
 & \ddots & \ddots & \ddots & \\
 & & 1 & 4 & 1 \\
 & & & 1 & 4
\end{pmatrix} \cdot \begin{pmatrix} a_1 \\ \\ \vdots \\ \\ a_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \\ \vdots \\ \\ f_n \end{pmatrix}. \tag{3.124}$$

An dieser Stelle sei erwähnt, daß eine Matrix $\mathsf{A} = a_{ij}$ *diagonaldominant* heißt, genau dann wenn für alle $i = 1, \ldots, n$ gilt

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|. \tag{3.125}$$

Wenn anstelle von $\geq$ sogar $>$ gilt, so heißt die Matrix A *streng diagonaldominant*. Man kann zeigen, daß Matrizen, die streng diagonaldominant sind, auch regulär und damit invertierbar sind (siehe auch Anhang A.4.4). Offenbar ist die Matrix in (3.124) streng diagonaldominant. Damit ist das lineare Problem eindeutig lösbar. Es ist sogar sehr schnell lösbar (Kap. 2.1.3).

Bei der obigen Form (3.124) hat man zunächst keinen Einfluß auf die Randsteigungen, anders als im Fall der allgemeinen kubischen Splines in Kap. 3.3.2. Man kann aber anstelle der ersten und letzten Gleichung von (3.123) an den Rändern fordern

$$a_0 B_0(x_1) + a_1 B_1(x_1) + a_2 B_2(x_1) = f_1, \tag{3.126a}$$
$$a_{n-1} B_{n-1}(x_n) + a_n B_n(x_n) + a_{n+1} B_{n+1}(x_n) = f_n, \tag{3.126b}$$

wobei man a_0 und a_{n+1} beliebig wählen kann, um zum Beispiel die Randsteigungen festzulegen. Die Summanden mit den beliebig aber fest gewählten Werten a_0 und a_{n+1} kann man dann auf die rechte Seite der Gleichung bringen, was auf den modifizierten Vektor der Inhomogenitäten $[f_1 - a_0 B_0(x_1), f_2, \ldots, f_{n-1}, f_n - a_{n+1} B_{n+1}(x_n)]^{\mathrm{T}}$ führt.

Für eine direkte Lösung zu gegebenen Randsteigungen f'_1 und f'_n kann man die erste und die letzte Gleichung in (3.124) durch (3.126) ersetzen und die zwei zusätzlichen Gleichungen

$$a_0 B'_0(x_1) + a_1 \overbrace{B'_1(x_1)}^{=0} + a_2 B'_2(x_1) = f'_1, \tag{3.127a}$$
$$a_{n-1} B'_{n-1}(x_n) + a_n \underbrace{B'_n(x_n)}_{=0} + a_{n+1} B'_{n+1}(x_n) = f'_n, \tag{3.127b}$$

als 0-te und $(n+1)$-te Gleichung für a_0 und a_{n+1} verwenden. Die Matrix des erweiterten Systems hat dann noch Einträge $A_{0,2}$ und $A_{n+1,n-1}$, die man vorab durch Zeilenoperationen elimimieren kann, um wieder zu einem tridiagonalem System zu kommen.

3.4. Methode der kleinsten Quadrate

Im vorangegangenen Abschnitt haben wir gesehen, wie wir durch $n+1$ Funktionswerte an paarweise verschiedenen Stellen x_i genau ein Polynom vom Grade n legen können. Ein anderes Problem tritt auf, wenn man beispielsweise sehr viele ($> n+1$)

Meßwerte $f_i = f(x_i)$ durch ein Polynom niedrigen Grades n *approximieren* möchte. Dann wird in der Regel $p(x_i) \neq f_i$ sein. Dies ist kein Problem, da die Daten f_i im Falle von Messungen ohnehin fehlerbehaftet sind. Vielmehr hofft man, daß sich der Fehler bei Berücksichtigung einer sehr großen Anzahl von Punkten herausmittelt.

Man wird also versuchen, ein Polynom (oder eine andere Funktion) zu finden, welches die Meßdaten in einem gewissen Sinne optimal wiederspiegeln. Dieses Problem wird *Ausgleichsproblem* genannt. Um die gesuchte *Ausgleichsfunktion* zu finden, fordert man typischerweise, daß die Summe der Abstandsquadrate der Meßdaten von der gesuchten Ausgleichsfunktion minimal wird. Dies ist die *Methode der kleinsten Quadrate*.

Seien nun $m > n+1$ Punkte x_i gegeben, wobei mindestens $n+1$ von ihnen paarweise verschieden sind, und $f_1 = f(x_1), \ldots, f_m = f(x_m)$ die zugehörigen Funktionswerte (Meßwerte). Wir definieren dann die Ausgleichsfunktion als ein Polynom n-ten Grades

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n. \tag{3.128}$$

Den Fehler des Ausgleichspolynoms können wir als gewichtete Summe der quadratischen Abweichungen definieren

$$g(a_0, \ldots, a_n) = \sum_{i=1}^{m} w_i \left[p(x_i) - f_i\right]^2 > 0, \tag{3.129}$$

wobei die Faktoren $w_i > 0$ positive Gewichte sind, mit denen man einigen Punkten eine höhere und anderen Punkten eine niedrigere Priorität beimessen kann. Der Fehler (3.129) wird in der Regel von Null verschieden sein. Die Aufgabe besteht nun darin, die Unbekannten $a_0, \ldots, a_n$ so zu bestimmen, daß die *Fehlerfunktion* (3.129) *minimal* wird.

Man könnte im Prinzip auch andere Maße zur Definition des Fehlers heranziehen. Die Verwendung der quadratischen Abweichungen hat jedoch den Vorteil, daß die Fehlerfunktion g quadratisch in jeder Variablen a_j ist und deshalb nur genau ein Extremum (ein Minimum) besitzt.

3.4.1. Konstanter Fit

Im einfachsten Fall ist das gesuchte Polynom vom Grade 0, also eine Konstante c. Dann lautet die Fehlerfunktion

$$g(c) = \sum_{i=1}^{m} w_i \left(c - f_i\right)^2. \tag{3.130}$$

Wenn wir die Bedingungen für ein Minimum von $g(c)$ auswerten, erhalten wir

$$g'(c) = 2\sum_{i=1}^{m} w_i \left(c - f_i\right) = 2c\sum_{i=1}^{m} w_i - 2\sum_{i=1}^{m} w_i f_i \stackrel{!}{=} 0, \tag{3.131a}$$

$$g''(c) = 2\sum_{i=1}^{m} w_i > 0. \tag{3.131b}$$

Unter der Annahme positiver Gewichte ($w_i > 0$) ist (3.131b) immer erfüllt und wir erhalten aus (3.131a) den gewichteten Mittelwert

$$c = \frac{\sum_{i=1}^{m} w_i f_i}{\sum_{i=1}^{m} w_i}. \tag{3.132}$$

Wenn alle Punkte gleich gewichtet werden ($w_i = 1$) ergibt sich der einfache Mittelwert

$$c = \frac{\sum_{i=1}^{m} f_i}{m}. \tag{3.133}$$

3.4.2. Linearer Fit

Beim linearen Fit suchen wir das Polynom $a_0 + a_1 x$, welches die Fehlerfunktion

$$g(a_0, a_1) = \sum_{i=1}^{m} w_i \left(a_0 + a_1 x_i - f_i\right)^2. \tag{3.134}$$

minimiert. Die Minimierung von g muß nun bezüglich der beiden unbekannten Koeffizienten a_0 und a_1 durchgeführt werden. Damit die Fläche g über der (a_0, a_1)-Ebene ein Minimum aufweist, müssen beide partiellen Ableitungen von g (nach a_0 und nach a_1) an der Stelle des Extremums verschwinden. Dies führt auf

$$\frac{1}{2}\frac{\partial g}{\partial a_0} = \sum_{i=1}^{m} w_i \left(a_0 + a_1 x_i - f_i\right) = a_0 \sum_{i=1}^{m} w_i + a_1 \sum_{i=1}^{m} w_i x_i - \sum_{i=1}^{m} w_i f_i = 0, \tag{3.135a}$$

$$\frac{1}{2}\frac{\partial g}{\partial a_1} = \sum_{i=1}^{m} w_i x_i \left(a_0 + a_1 x_i - f_i\right) = a_0 \sum_{i=1}^{m} w_i x_i + a_1 \sum_{i=1}^{m} w_i x_i^2 - \sum_{i=1}^{m} w_i x_i f_i = 0. \tag{3.135b}$$

Dies stellt ein lineares Gleichungssystem der Ordnung 2 für die beiden Unbekannten a_0 und a_1 dar

$$\begin{pmatrix} \sum_{i=1}^{m} w_i & \sum_{i=1}^{m} w_i x_i \\ \sum_{i=1}^{m} w_i x_i & \sum_{i=1}^{m} w_i x_i^2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} w_i f_i \\ \sum_{i=1}^{m} w_i f_i x_i \end{pmatrix}. \tag{3.136}$$

3.4.3. Normalgleichungen

Im Falle eines Fits mittels eines Polynoms vom Grade n lautet die Fehlerfunktion

$$g(a_0, \ldots, a_n) = \sum_{i=1}^{m} w_i \left(a_0 + a_1 x_i + \ldots + a_n x_i^n - f_i\right)^2. \tag{3.137}$$

Die notwendigen Bedingungen für ein Minimum lauten dann

$$\frac{1}{2}\frac{\partial g(a_0, \ldots, a_n)}{\partial a_j} = \sum_{i=1}^{m} w_i \left(x_i\right)^j \left[a_0 + a_1 x_i + \ldots + a_n x_i^n - f_i\right] = 0, \quad j = 0, \ldots, n. \tag{3.138}$$

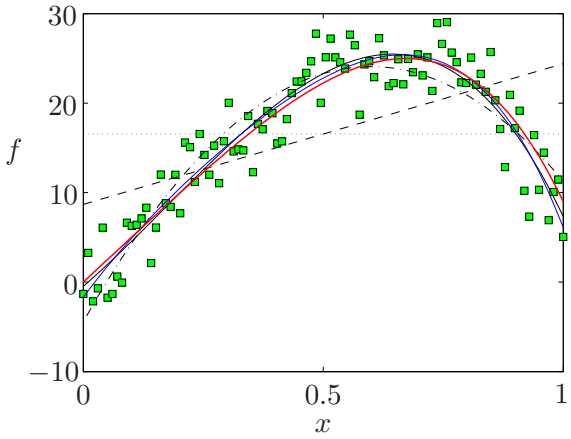


Abbildung 3.11.: Approximation der verrauschten Funktion $y = 50x - x^2 - 40x^4$ (grüne Punkte). Gezeigt sind die Polynom-Approximationen (Gewichtsfunktion $w_i = 1$) vom Grade 0 (gepunktet), 1 (gestrichelt), 2 (strichpunktiert), 3 (durchgezogen), 4 (blau) sowie die unverrauschte Funktion (rot).

Dies sind $n+1$ lineare Gleichungen für die $n+1$ Unbekannten a_i. Sie heißen *Normalgleichungen*. Man kann diese Gleichungen wieder in Matrix-Form schreiben

$$\begin{pmatrix} s_0 & s_1 & s_2 & \dots & s_n \\ s_1 & s_2 & s_3 & \dots & s_{n+1} \\ s_2 & & & \iddots & \vdots \\ \vdots & & \iddots & & \vdots \\ s_n & s_{n+1} & \dots & \dots & s_{2n} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \\ \vdots \\ \\ a_n \end{pmatrix} = \begin{pmatrix} c_0 \\ \\ \vdots \\ \\ c_n \end{pmatrix}, \tag{3.139}$$

wobei

$$s_j = \sum_{i=1}^{m} w_i \, (x_i)^j \qquad \text{und} \qquad c_j = \sum_{i=1}^{m} w_i \, (x_i)^j \, f_i. \tag{3.140}$$

Ein Beispiel für einen Polynom-Fit ist in Abb. 3.11 gezeigt. Für $m = n+1$ erhält man das eindeutige Interpolationspolynom.[6] Der MATLAB-Befehl `a=polyfit(x,f,n)` berechnet die Koeffizienten a_j eines Polynoms vom Grade n mittels kleinster Quadrate, wobei die m Datenpaare in den Vektoren $x(i) = x_i$ und $f(i) = f(x_i) = f_i$ vorgegeben werden müssen.

Die $(n+1) \times (n+1)$-Matrix in (3.139) enthält identische Einträge auf den Querdiagonalen (die zweite Querdiagonale is rot angedeutet) und ist daher symmetrisch. Sie enthält nur die $2n+1$ Größen s_n (3.140), welche *Momente* genannt werden. Eine Matrix mit konstanten Querdiagonalen heißt *Hankel-Matrix*.

Man kann die Gewichte w_i auch aus der obigen Hankel-Matrix herausziehen. Dann erhalten die Bestimmungsgleichungen der Koeffizienten die Form

$$\mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \mathsf{E} \cdot \vec{a} = \mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \vec{f}. \tag{3.141}$$

[6] Aufgabe: Zeige für $m = n+1$ die Äquivalenz von (3.139) und (3.90).

Hierbei ist W eine $m \times m$-Matrix, welche die Gewichte w_i auf der Diagonale enthält und E eine $m \times (n+1)$-Matrix vom Vandermondeschen Typ (siehe (3.90))[7]

$$\mathsf{W} = \begin{pmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{pmatrix}, \qquad \mathsf{E} = \begin{pmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & & x_2^n \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^n \end{pmatrix}. \tag{3.142}$$

Man kann zeigen (Golub and Ortega, 1996), daß $\mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \mathsf{E}$ für mindestens $n+1$ paarweise verschiedene Punkte positiv definit (s. Anhang A.4.5) ist. Damit ist $\mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \mathsf{E}$ auch regulär und (3.141) besitzt eine eindeutige Lösung.

3.4.4. Allgemeine Ausgleichsprobleme

Man kann die Methode der kleinsten Quadrate verallgemeinern. Es braucht nicht unbedingt ein Polynom zu sein, welches man für die Approximation verwendet. Man kann auch eine Linearkombination von irgendwelchen fest vorgegebenen Funktionen $\phi_0, \dots, \phi_n$ verwenden. Dazu macht man den Ansatz

$$\phi(x) = \sum_{i=0}^{n} a_i \phi_i(x). \tag{3.143}$$

Hermann Hankel 1839–1873

Im Fall des Ausgleichs mittels Polynomen ist $\phi_i = x^i$. Andere Möglichkeiten bestehen in der Wahl von $\phi_i = \cos(i\pi x)$ oder $\phi_i = \exp\{k_i x\}$ mit $k_i \in \mathbb{R}$. Welche Funktionen man am besten verwendet, hängt stark von der erwarteten x-Abhängigkeit der Daten ab.

Im allgemeinen Fall können wir genauso vorgehen wie beim Ausgleich mittels Polynomen. Die Fehlerfunktion lautet im allgemeinen Fall

$$g(a_0, \dots, a_n) = \sum_{k=1}^{m} w_k \left[\sum_{i=0}^{n} a_i \phi_i(x_k) - f_k \right]^2. \tag{3.144}$$

Die notwendigen Bedingungen für ein Minimum lauten damit

$$\frac{1}{2} \frac{\partial g(a_0, \dots, a_n)}{\partial a_j} = \sum_{k=1}^{m} w_k \phi_j(x_k) \left[a_0 \phi_0(x_k) + \dots + a_n \phi_n(x_k) - f_k \right] = 0, \quad j = 0, \dots, n. \tag{3.145}$$

[7] Es ist

$$\mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \mathsf{E} = (E^{\mathrm{T}})_{li} w_j \delta_{ij} E_{jk} = \sum_{i=1}^{m} (E^{\mathrm{T}})_{li} \sum_{j=1}^{m} w_j \delta_{ij} E_{jk} = \sum_{i=1}^{m} (E^{\mathrm{T}})_{li} w_i E_{ik} = \sum_{i=1}^{m} w_i (E^{\mathrm{T}})_{li} E_{ik}$$
$$= \sum_{i=1}^{m} w_i (x_i)^l (x_i)^k = \sum_{i=1}^{m} w_i (x_i)^{l+k}, \; l+k = j.$$

Wenn man die Gleichungen zusammenfaßt, erhält man die Matrix-Form

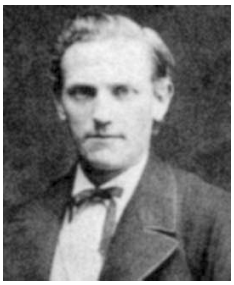
$$\begin{pmatrix} s_{00} & s_{01} & s_{02} & \dots & s_{0n} \\ s_{10} & s_{11} & s_{12} & \dots & s_{1n} \\ s_{20} & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ s_{n0} & s_{n1} & \dots & \dots & s_{nn} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix} \tag{3.146}$$

wobei nun

$$s_{ji} = \sum_{k=1}^{m} w_k \phi_j(x_k)\phi_i(x_k) \qquad \text{und} \qquad c_j = \sum_{k=1}^{m} w_k \phi_j(x_k) f_k. \tag{3.147}$$

Die Matrix ist symmetrisch ($s_{i,j} = s_{j,i}$). Die Querdiagonalen sind aber nicht notwendigerweise konstant ($s_{i,j} \neq s_{i-1,j+1}$). Deshalb ist die Matrix i.a. keine Hankel-Matrix, sondern nur noch eine sogenannte *Gramsche Matrix*. Damit (3.146) eindeutig lösbar ist, muß man bestimmte Forderungen an das System von Ansatzfunktionen $\{\phi_i(x)\}$ stellen.[8]

Die Normalgleichungen (3.139) und (3.146) können in der Regel für kleine Werte von n gut gelöst werden. Wird n jedoch zu groß, kann es sein, daß die Matrix schlecht konditioniert ist (siehe Kap. 2.1.4 und (2.57)). Dies kann zu numerischen Problemen führen. Eine Möglichkeit der Abhilfe besteht darin, die Approximation nicht mit Hilfe der Monome x^n, sondern mit Hilfe eines Systems von *orthogonalen Polynomen* $\{q_k(x)\}$ (bezüglich der Punkte $\{x_k\}$) durchzuführen. Diese haben die Eigenschaft, daß alle Matrixelemente $s_{ij} = \sum_{k=1}^{m} w_k q_i(x_k) q_j(x_k) = a_i \delta_{ij}$ bis auf die Diagonalelemente verschwinden. Dann wird die Matrix des zu lösenden Systems diagonal. Wie man die orthogonalen Polynome systematisch konstruieren kann, ist z.B. in Golub and Ortega (1996) beschrieben.[9]



Jorgen Pedersen Gram
1850–1916

3.5. Bestimmung von Nullstellen

Die Bestimmung von Nullstellen ist ein Problem, das sehr häufig auftritt. Schon das anscheinend einfache mechanische Problem, die Gleichgewichtslage eines Systems

[8] Man kann (3.146) auch wieder in der Form (3.141) schreiben mit entsprechend modifizierten Matrizen E und W. Jedoch ist $\mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \mathsf{E}$ im allgemeinen nicht mehr positiv definit. Für eine eindeutige Lösung muß man die Ansatzfunktionen (und Punkte x_i) so wählen, daß $\mathsf{E}^{\mathrm{T}} \cdot \mathsf{W} \cdot \mathsf{E}$ positiv definit ist.

[9] Polynome $q_0, \dots, q_n$ vom Grade $0, \dots, n$ heißen *orthogonal* bezüglich der Punkte $x_1, \dots, x_m$, wenn gilt

$$\sum_{i=1}^{m} q_k(x_i) q_j(x_i) = 0, \qquad \text{für} \quad k, j = 0, 1, \dots, n, \ k \neq j.$$

Man kann sie sukzessive aus den Daten konstruieren.

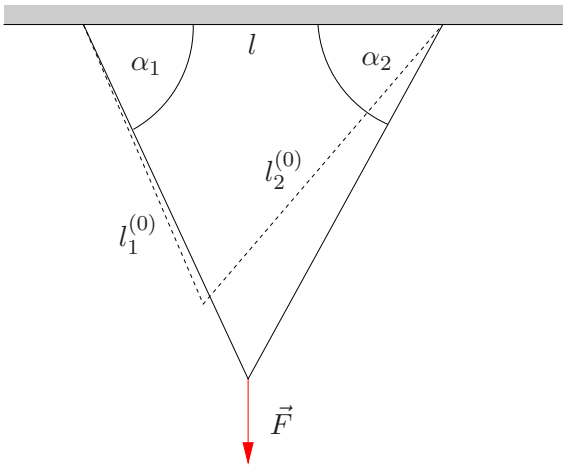


Abbildung 3.12.: Gleichgewicht zweier elastischer Stäbe unterschiedlicher Elastizität bei Belastung.

aus zwei elastischen Stäben unter einer Last (Zweischlag, Abb. 3.12) zu berechnen, führt auf nichtlineare Gleichungen, die nicht mehr in geschlossener Form gelöst werden können. Auch die Navier-Stokes-Gleichung der Strömungsmechanik ist aufgrund des konvektiven Terms $\vec{u} \cdot \nabla \vec{u}$ nichtlinear. Zur Berechnung von Strömungen muß man daher im allgemeinen ein großes System nichtlinearer Gleichungen für die unbekannten Geschwindigkeitskomponenten an allen Punkten eines feinen Gitters lösen.

3.5.1. Newton-Verfahren

Ein einfaches algebraisches Problem könnte lauten: Berechnen Sie die Lösung(en) von

$$\lambda x^2 = \sin(x). \tag{3.148}$$

Die graphische Lösung ist in Abb. 3.13a dargestellt. Eine nichtlineare algebraische Gleichung wie (3.148) kann man immer in der Form eines *Nullstellenproblems*

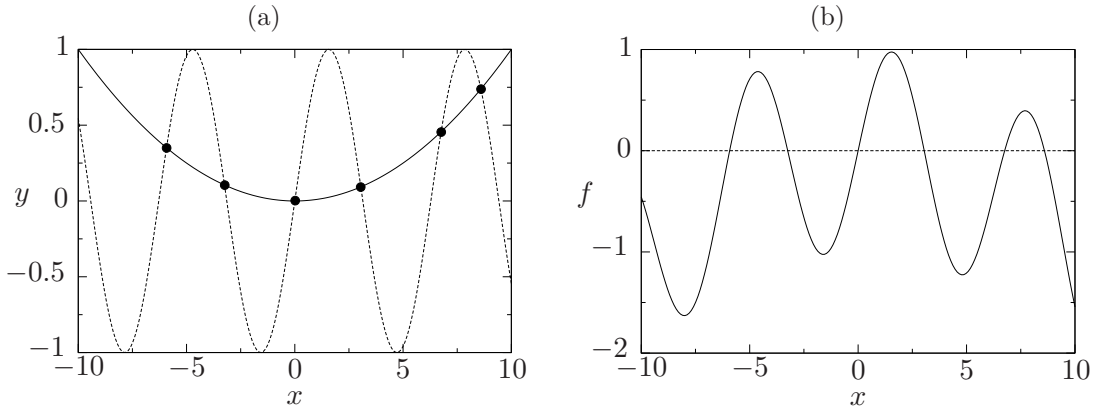


Abbildung 3.13.: (a) Graphische Lösung von (3.148) für $\lambda = 0.01$. (b) Entsprechendes Nullstellenproblem mit $f(x) = \sin(x) - \lambda x^2$.

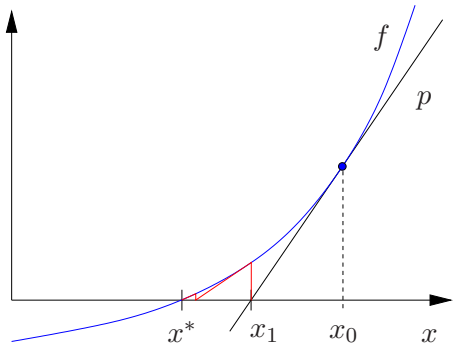


Abbildung 3.14.: Lineare Näherung der Funktion f in der Nähe der Nullstelle x^* durch die Tangente p an f bei x_0. Die roten Linien verdeutlichen die Abfolge der Newton-Iteration anschaulich.

schreiben. Hier lautet das Nullstellenproblem (Abb. 3.13b)

$$f(x) = \sin(x) - \lambda x^2 = 0. \tag{3.149}$$

Wäre $f(x)$ ein Polynom vom Grade n, dann hätte f genau n reelle oder komplexe Nullstellen (Wurzeln). Für eine allgemeine Funktion läßt sich nicht *a priori* sagen, wieviele Nullstellen sie in einem bestimmten Intervall besitzt. Es gilt aber folgendes:

- Ist eine Funktion $f(x)$ in einem Intervall $[a, b]$ streng monoton, dann besitzt $f(x)$ *maximal* eine Nullstelle.
- Ist eine Funktion $f(x)$ in einem Intervall $[a, b]$ stetig und ist $f(a)f(b) < 0$, dann besitzt $f(x)$ im Intervall $[a, b]$ *mindestens* eine Nullstelle.

Um eine Nullstelle x^* von $f(x)$ numerisch zu bestimmen, wird meistens der Funktionsverlauf zwischen den Punkten a und b durch ein Polynom $p(x)$ niedriger Ordnung approximiert und dessen Nullstelle näherungsweise als Nullstelle von $f(x)$ aufgefaßt.

Sei x_0 ein Punkt in der Nähe der Nullstelle. Für die Taylor-Entwicklung von f um den Punkt x_0 gilt

$$f(x) = \underbrace{f(x_0) + (x - x_0)f'(x_0)}_{:=p(x)} + \underbrace{\frac{1}{2}f''(z)(x - x_0)^2}_{\text{Restglied}}. \tag{3.150}$$

Die Taylor-Entwicklung bis zur ersten Ordnung können wir als Polynomapproximation erster Ordnung auffassen. Indem wir f durch p ersetzen, haben wir f linearisiert. Dabei wird die Funktion f lokal durch die Tangente im Punkt x_0 ersetzt (Abb. 3.14). Aus (3.150) ergibt sich die Approximation x_1 der Nullstelle von f durch die Bedingung $p(x_1) = 0$. Diese liefert

$$p(x_1) = f(x_0) + (x_1 - x_0)f'(x_0) = 0, \tag{3.151}$$

also

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{3.152}$$

Wenn man nun vom Punkt x_1 ausgeht, diesen als neuen Entwicklungspunkt wählt und auf dieselbe Art eine zweite Näherung x_2 berechnet, und so weiter, dann erhält man die *Newton-Iteration*

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad n = 0, 1, \ldots . \tag{3.153}$$

Diese Iterationsvorschrift wird auch *Newton-Raphson-Verfahren*[10] genannt.

Um die Konvergenz der Newton-Iteration zu untersuchen, betrachten wir den Fehler $x^* - x_{n+1}$, den wir beim $(n+1)$-ten Iterationsschritt machen. Dazu werten wir die Taylor-Darstellung der Funktion $f(x) = p(x) + (x - x_n)^2 f''(z)/2$ an der Stelle $x = x^*$ aus und erhalten für die Abweichung von $f(x^*) = 0$ von der linearen Näherung $p(x^*)$

$$\underbrace{f(x^*)}_{=0} - p(x^*) = \overbrace{\underline{-f(x_n)} - (x^* - \underline{x_n}) f'(x_n)}^{-p(x^*)} \stackrel{!}{=} \frac{1}{2} f''(z)(x^* - x_n)^2. \tag{3.154}$$

Mit Hilfe eines Newton-Schrittes (3.153) können wir die unterstrichenen Terme durch $x_n f'(x_n) - f(x_n) = x_{n+1} f'(x_n)$ ersetzen und erhalten so

$$x_{n+1} f'(x_n) - x^* f'(x_n) = \frac{1}{2} f''(z)(x^* - x_n)^2. \tag{3.155}$$

Mit $f'(x_n) \neq 0$ ergibt sich für den Fehler

$$x^* - x_{n+1} = \underbrace{-\frac{1}{2} \frac{f''(z)}{f'(x_n)}}_{c_n} (x^* - x_n)^2 = c_n (x^* - x_n)^2. \tag{3.156}$$

Ist nun in einem Intervall um die Nullstelle $|f''(x)| \leq M$ betragsmäßig beschränkt und $|f'(x)| \geq m > 0$, dann kann man $|c_n| \leq M/(2m) := c$ nach oben hin abschätzen. Aus der Beschränktheit des Koeffizienten folgt, daß sich der Fehler in jedem Schritt quadratisch verringert. Ist zum Beispiel der Fehler schon $|x^* - x_n| = 10^{-4}$, dann wird er im nächsten Schritt kleiner sein als $c \times 10^{-8}$. Bei jeder Iteration verdoppelt sich ungefähr die Anzahl der korrekten Dezimalen der Approximation der Nullstelle (siehe Tab. 3.1). Typischerweise wählt man als Abbruchkriterium der Newton-Iteration die Bedingung $|x_{n+1} - x_n| < \epsilon$.

Die *quadratische Konvergenz* der Newton-Iteration ist so ziemlich das schnellste, was man erreichen kann. Die rapide Konvergenz hat ihren Preis, denn die Newton-Iteration konvergiert nur, wenn der Anfangswert x_0 schon hinreichend nahe bei

[10] Joseph Raphson, 1648–1715: Raphson war einer der wenigen, denen es Isaac Newton erlaubte, vorab seine mathematischen Werke zu sehen. 1690 veröffentlichte Raphson sein Buch *Analysis aequationum universalis*, welches das Newton-Raphson-Verfahren zur numerischen Lösung von nichtlinearen Gleichungen enthält. Hierfür wurde er 1691 zum Mitglied der Royal Society ernannt. Außerdem schrieb er das Buch *History of Fluxions*, welches aber erst nach seinem Tod erschien.

n	x_n
0	4
1	2.750343532969441
2	3.062460099178964
3	3.048532919044707
4	3.048523403179332
5	3.048523403174493
6	3.048523403174493

Tabelle 3.1.: Beispiel für die Newton-Iteration von (3.149) mit $\lambda = 0.01$ und Startwert $x_0 = 4$. Man erkennt die quadratische Konvergenz.

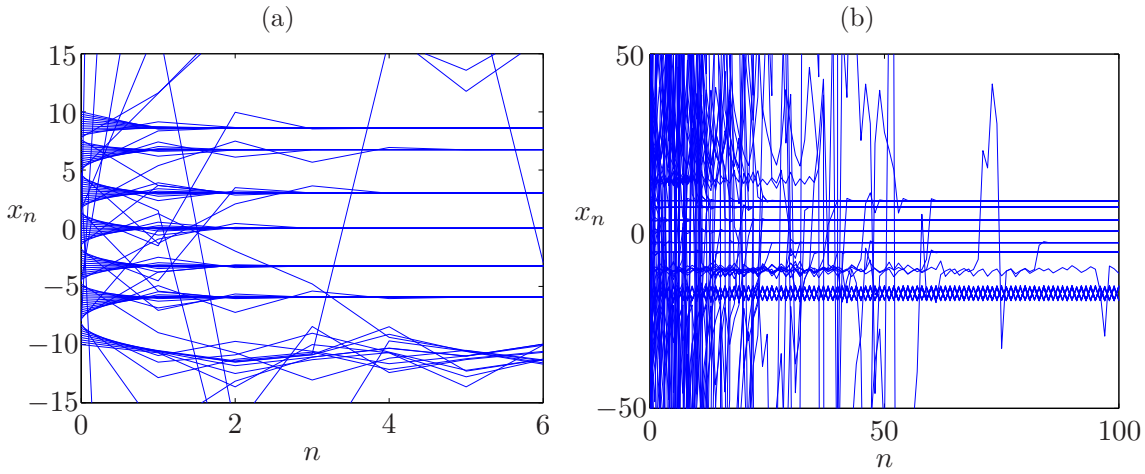


Abbildung 3.15.: Newton-Iteration für das Beispiel (3.149) mit $\lambda = 0.01$. In (a) wurden äquidistanten Anfangswerte $x_0 \in [-10, 10]$ mit $\Delta x_0 = 0.2$ gewählt. Die Iterationen konvergieren sehr schnell für die meisten Anfangsbedingungen nahe einer der sechs Nullstellen. In (b) wurden Startwerte $x_0 \in [-50, 50]$ mit $\Delta x_0 = 1$ gewählt. Für Anfangsbedingungen, die weit von einer Lösung entfernt sind, treten sehr große Schwankungen der Iterierten oder periodisches bzw. quasiperiodisches Verhalten auf.

der wirklichen Nullstelle x^* liegt. Dieses Verhalten wird auch als *lokale Konvergenz* bezeichnet. Für das Beispiel (3.149) ist die schnelle Konvergenz in Abb. 3.15 dargestellt. Falls der Startwert x_0 nicht hinreichend nahe an x^* liegt, kann die Iteration auch divergieren oder in chaotischer Weise schwanken. Auch ist ein periodisches Verhalten der Iterierten möglich.

3.5.2. Sekantenmethode

Grundlage der Newton-Iteration war die Taylor-Entwicklung in der Nähe eines Punktes, der hinreichend dicht an der Nullstelle liegt. Anstelle der Taylor-Entwicklung kann man zur Konstruktion einer verbesserten Näherung der Nullstelle auch die Sekante verwenden, die durch die Punkte $f(x_0)$ und $f(x_1)$ definiert ist, wobei x_0 und x_1 die beiden vorhergehenden Iterierten sind (Abb. 3.16).

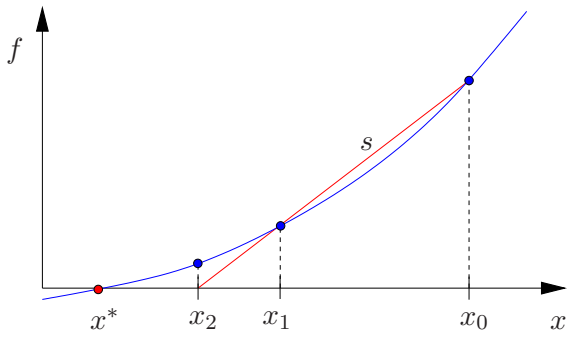


Abbildung 3.16.: Approximation der Nullstelle von f durch die Nullstelle der Sekante s.

Die Gleichung für die Sekante mit Stützpunkten x_0 und x_1 lautet

$$s = f_1 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_1) \tag{3.157}$$

Für die Nullstelle x_2 der Sekante gilt dann mit $s(x_2) = 0$

$$0 = f_1 + \frac{f_1 - f_0}{x_1 - x_0}(x_2 - x_1) \quad \Rightarrow \quad x_2 = x_1 - f_1 \frac{x_1 - x_0}{f_1 - f_0}. \tag{3.158}$$

Wenn man diese Methode systematisch fortsetzt, kommt man auf die Iteration (*Sekantenmethode*)

$$x_{n+1} = x_n - f_n \frac{x_n - x_{n-1}}{f_n - f_{n-1}}, \qquad n = 1, 2, \ldots. \tag{3.159}$$

Die Sekantenmethode hat eine ähnliche Struktur wie die Newton-Methode. Der Unterschied besteht darin, daß bei der Sekantenmethode die Ableitung diskret gebildet wird, und zwar mit Hilfe der beiden vorhergehenden Iterationspunkte

$$f'_n \approx \frac{f_n - f_{n-1}}{x_n - x_{n-1}}. \tag{3.160}$$

Die Konvergenzrate der Sekanten-Methode ist im allgemeinen geringer als diejenige der Newton-Methode. Man kann zeigen, daß im Limes $x_n \to x^*$ die Konvergenzordnung $\left(1 + \sqrt{5}\right)/2 \approx 1.62$ beträgt, daß also gilt

$$|x^* - x_{n+1}| \sim c\,|x^* - x_n|^{1.62}. \tag{3.161}$$

Dieser Nachteil wird aber unter Umständen dadurch kompensiert, daß man für die Sekanten-Methode pro Iteration nur *einen* Funktionswert f berechnen muß, während man bei der Newton-Iteration sowohl f als auch f' bestimmen muß.

3.5.3. Bisektion und Regula Falsi

Die einfachste Möglichkeit, eine Nullstelle zu bestimmen, besteht in der Halbierung des Intervalls (*Bisektion*). Angenommen, in dem Intervall $[a, b]$ mit $f(a)f(b) < 0$

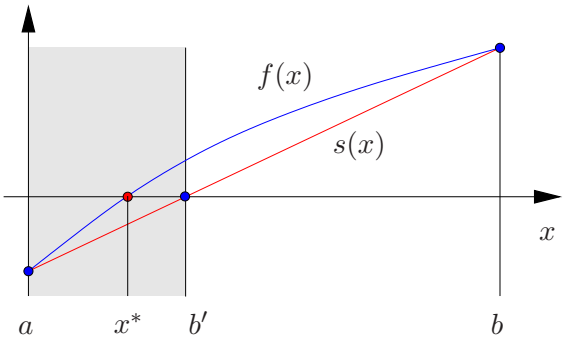


Abbildung 3.17.: Regula falsi: Die Nullstelle von $f(x)$ befindet sich im grauen Teilintervall $[a, b']$, welches durch die Nullstelle b' der Sekante $s(x)$ und $f(a)f(b') < 0$ definiert ist.

existiert nur eine Nullstelle von $f(x)$, dann berechnet man $f(x_1)$ an der Stelle $x_1 = (a+b)/2$. Die Nullstelle befindet sich dann in einem der beiden Teilintervalle

$$\begin{array}{lll} [a, x_1], & \text{falls} & f(a)f(x_1) < 0, \\ [x_1, b], & \text{falls} & f(x_1)f(b) < 0. \end{array} \tag{3.162}$$

Diese Intervallhalbierung wird nun iteriert. Bei der Intervallhalbierung kann man den Fehler mit Hilfe der ursprünglichen Intervallänge $b - a$ abschätzen. Nach m Bisektionen beträgt er

$$|x^* - x_m| \leq \frac{b-a}{2^m}. \tag{3.163}$$

Um den Fehler um 6 Zehnerpotenzen zu verringern ($2^m = 10^6$), benötigt man daher $m = 6\log_2(10) \approx 20$ Iterationen. Die Konvergenz ist also sehr langsam.

Um eine Verbesserung der Konvergenzrate zu erreichen, kann man anstelle der Intervallhalbierung die neue Intervallgrenze x_1 auch mit Hilfe der Sekanten-Methode (3.159) berechnen. Das resultierende Verfahren heißt dann *regula falsi* (Abb. 3.17).

3.5.4. Fehler und Kondition

Numerisch kann man die Funktion, deren Nullstelle zu bestimmen ist, nur mit endlicher Genauigkeit auswerten. Da die Funktionswerte in der Nähe der Nullstelle sehr klein werden, kann es sein, daß bei einer bestimmten Iteration der Bisektion das Vorzeichen von f falsch ist. Weitere Iterationen machen dann keinen Sinn mehr und die Fehlerabschätzung (3.163) verliert ihre Gültigkeit. Dieses Problem äußert sich dann in einem irregulären Verhalten der Iteration. Dies trifft auch für die Newton-Iteration und die Sekanten-Methode zu, denn auch bei diesen Verfahren muß die kleine Größe f in der Nähe der Nullstelle berechnet werden.

Das Problem ist offenbar umso größer, je flacher die Funktion f durch die Nullstelle geht. Am Beispiel einer linearen Funktion ist dieser Sachverhalt in Abb. 3.18 dargestellt. Ein anderes bekanntes Beispiel sind die n entarteten Nullstellen $x_j = 0$ von $x^n = 0$. Falls nun die rechte Seite nicht Null ist, sondern nur ein wenig davon abweicht, erhält man

$$x^n = \epsilon, \tag{3.164}$$

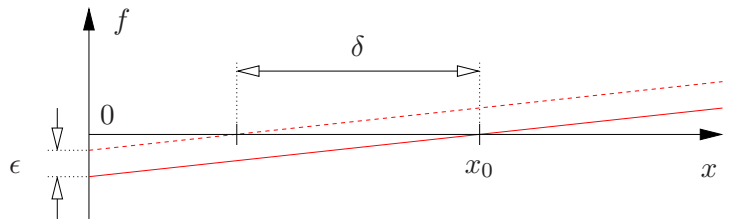


Abbildung 3.18.: Ungenauigkeit der Nullstellenbestimmung bei schlechter Kondition: Bei geringer Steigung $f'(x_0)$ hat ein kleiner additiver Fehler ϵ eine große Auswirkung auf die Verschiebung δ der Nullstelle.

mit den Lösungen

$$x_j = \epsilon^{1/n} \exp\left\{2\pi \mathrm{i} \frac{j}{n}\right\}, \quad j = 1, \ldots n. \tag{3.165}$$

Falls $n = 10$ und $\epsilon = 10^{-10}$ ist, dann sind die Wurzeln betragsmäßig gleich $\epsilon^{1/n} = (10^{-10})^{1/10} = 10^{-1}$. In diesem Fall hat eine Änderung der Eingangsdaten um 10^{-10} eine Änderung der Ausgangsdaten von 10^{-1} bewirkt. Der Fehler wurde demnach um einen Faktor 10^9 verstärkt!

Ganz allgemein bezeichnet man ein Problem, bei dem kleine Ungenauigkeiten in den Eingangsdaten (Parametern) zu sehr großen Ungenauigkeiten der Ausgangsdaten (Lösungen) führt, als schlecht konditioniert. Man sagt auch: die *Kondition eines Problems* ist schlecht. Diesen Sachverhalt kann man noch präzisieren und durch Zuweisung einer *Konditionszahl* quantifizieren (siehe auch Kap. 2.1.4).

3.5.5. Fixpunktiteration

Die Nullstelle einer Funktion $f(x)$ wie (3.149) kann man auch mittels *Fixpunktiteration* finden. Dazu addiert man x zu beiden Seiten der Gleichung

$$x = f(x) + x. \tag{3.166}$$

Um zu einer Iteration zu kommen, wird die linke Seite bei $n+1$ und die rechte bei n ausgewertet. So erhält man

$$x_{n+1} = g(x_n). \tag{3.167}$$

In unserem Beispiel ist $g(x) = \sin(x) - \lambda x^2 + x$. Die beiden Seiten der Gleichung (3.166) sind in Abb. 3.19 dargestellt. An den Schnittpunkten beider Graphen gilt $x_n = g(x_n) = x^*$. Sie heißen *Fixpunkte*. Für die Fixpunkte gilt offenbar $f(x^*) = 0$. Damit sind die Fixpunkte identisch mit den Nullstellen von $f(x)$. Es gibt stabile und instabile Fixpunkte. Ob ein Fixpunkt *stabil* oder *instabil* ist, hängt davon ab, ob die Iteration eines Startwertes in der Nähe des Fixpunktes auf den Fixpunkt konvergiert, oder ob sie sich von ihm entfernt. Der linke Fixpunkt in Abb. 3.19b ist instabil, während der rechte stabil ist. Für einen stabilen Fixpunkt muß in einer Umgebung des Fixpunktes gelten $|g'(x)| < 1$. Wenn wir nun ein Intervall $x \in [a, b]$ betrachten, so gilt

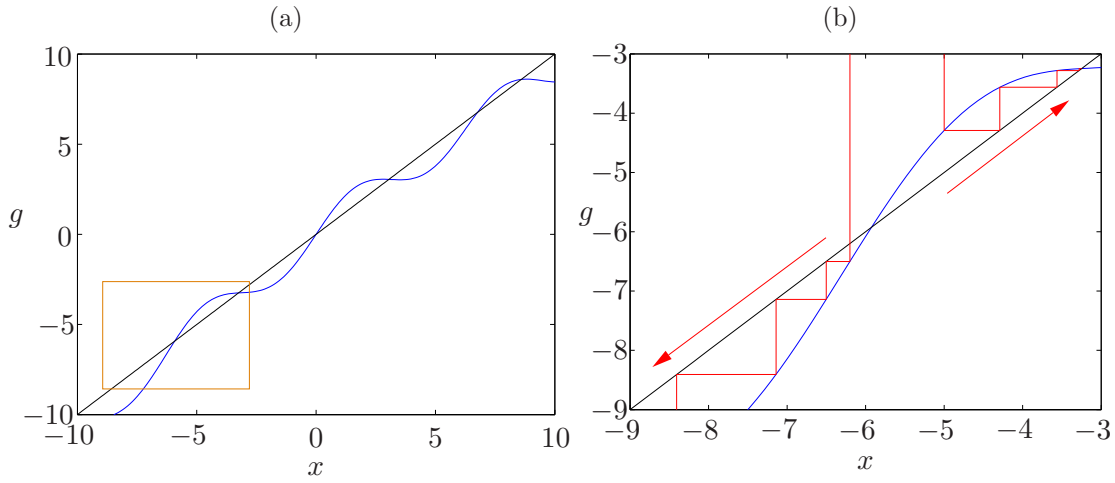


Abbildung 3.19.: (a) Die beiden Seiten x (schwarz) und $g(x)$ (blau) aus (3.167) mit $\lambda = 0.01$. (b) Zoom in das Gebiet der beiden linken Schnittpunkte (Fixpunkte) in (a). Die roten Linien zeigen den Verlauf der Fixpunktiteration für die Startwerte $x_0 = -5$ und $x_0 = -6.2$.

Falls

1. $\forall_{x\in[a,b]} \quad g(x) \in [a,b]$,
2. $g(x)$ ist in $[a,b]$ diffbar,
3. $\exists_{K<1} \quad \forall_{x\in[a,b]} \quad |g'(x)| \leq K < 1$,

dann besitzt $g(x)$ *genau einen* Fixpunkt in $[a,b]$ und für alle Anfangswerte x_0 konvergiert die Iteration (3.167) gegen den Fixpunkt x^*, wobei

$$\lim_{n\to\infty} \frac{x_{n+1} - x^*}{x_n - x^*} = g'(x^*). \tag{3.168}$$

Um die Bedingung (3) zu erfüllen,[11] kann es unter Umständen nötig sein, die Funktion $f(x)$ in (3.167) geeignet zu skalieren. Die schnellste Konvergenz erhält man, wenn $|g'(x^*)| \to 0$ möglichst klein wird. Die instabilen Fixpunkte werden zu stabilen Fixpunkten, wenn man das Vorzeichen $f(x) \to -f(x)$ umkehrt und, falls erforderlich, geeignet skaliert.

Die babylonische Methode die Wurzel aus einer beliebigen positiven reellen Zahl

[11] Die dritte Bedingung nennt man auch *Lipschitz-Bedingung* mit *Lipschitz-Konstante* K.

a zu ziehen, bestand aus der Fixpunktiteration[12]

$$x_{n+1} = \frac{1}{2}\left(\frac{a}{x_n} - x_n\right) + x_n = \frac{1}{2}\left(\frac{a}{x_n} + x_n\right) = g(x_n). \tag{3.169}$$

Im Falle der Konvergenz für $n \to \infty$ gilt $x^{*2} = a$. Wegen $g'(x = x^* = \sqrt{a}) = 0$ gilt $|g'(x)| \leq 1$ in einer Umgebung von $x^* = \sqrt{a}$. Deshalb ist die Fixpunktiteration stabil und konvergiert optimal.

Auch das Newton-Verfahren (3.153) kann als Fixpunktiteration aufgefaßt werden, wenn man die Iterationsfunktion $g(x) := x - f(x)/f'(x)$ definiert. Mit einer Modifikation der Fixpunktiteration (*Aitken-* oder auch *Steffensen-Verfahren*) kann man die Konvergenz beschleunigen (siehe Quarteroni and Saleri, 2006).

3.5.6. Newton-Verfahren für nichtlineare Gleichungssysteme

Wir wollen nun das Newton-Verfahren zu Bestimmung der Nullstelle einer nichtlinearen Funktion $f(x)$ auf nichtlineare Gleichungssysteme aus I Gleichungen in Abhängigkeit von I Unbekannten x_i übertragen. Diese Art von Problemstellung tritt häufig auf, wenn man z.B. stationäre Lösungen (Gleichgewichtslagen) nichtlinearer dynamischer Systeme wie etwa (5.285) oder (5.351) sucht (siehe auch Abb. 3.12).

Die simultan zu lösenden Gleichungen kann man in der Form schreiben

$$f_i(x_1, \ldots, x_I) = \vec{f}(\vec{x}) = 0, \qquad i = 1, \ldots, I. \tag{3.170}$$

Wir suchen nun den Vektor $\vec{x}^* = (x_1^*, \ldots, x_I^*)^{\mathrm{T}}$, so daß (3.170) erfüllt ist. Die gesuchte Nullstelle ist also ein Punkt $\vec{x}^*$ im I-dimensionalen Raum.

Wie im eindimensionalen Fall sei $\vec{x}^{(0)}$ eine anfängliche Approximation der Nullstelle $\vec{x}^*$. Dann ergibt die Taylor-Entwicklung von $\vec{f}$ im I-dimensionalen Raum um den Punkt $\vec{x}^{(0)}$

$$\vec{f}(\vec{x}) = \underbrace{\vec{f}(\vec{x}^{(0)}) + \left(\vec{x} - \vec{x}^{(0)}\right) \cdot \nabla \vec{f}(x^{(0)})}_{:=\vec{p}(\vec{x})} + \text{ Terme höherer Ordnung.} \tag{3.171}$$

Die Linearisierung der Vektorfunktion $\vec{f}$ um den Punkt $\vec{x}^{(0)}$ lautet $\vec{f} = \vec{p}$. In zwei Dimensionen ($I = 2$) wird dann jede Komponente $f_i(x_1, x_2)$ der Funktion $\vec{f}(x_1, x_2)$ durch eine Ebene $p_i(x_1, x_2)$ approximiert, die im Punkt $\vec{x}^{(0)}$ tangential zur jeweiligen Fläche $f_i(x_1, x_2)$ ist (Abb. 3.20). Entsprechend wird in I Dimensionen jede Hyperfläche $f_i(x_1, \ldots, x_I)$ durch je eine Hyperebene approximiert, die im Punkt $\vec{x}^{(0)}$ tangential zur Hyperfläche $f_i(x_1, \ldots, x_I)$ ist.

[12] Aus $x^2 = a$ ergibt sich $a/x - x = 0$. Wenn man dies skaliert und x addiert, erhält man $\frac{1}{2}(a/x - x) + x = x$.

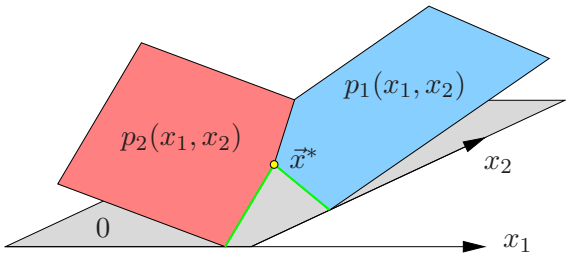
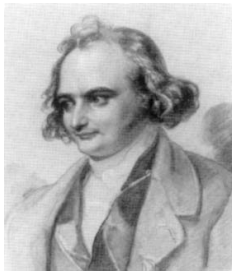


Abbildung 3.20.: Graphische Darstellung der Bestimmung der Nullstelle $\vec{x}^*$ in zwei Dimensionen als Schnittpunkt zweier bilinearer Funktionen p_1 und p_2 mit der Ebene $p_1 = p_2 = 0$.



Carl Gustav Jacob Jacobi 1804–1851

Wenn wir nun in (3.171) $\vec{p}\left(\vec{x}^{(1)}\right) = 0$ setzen, erhalten wir

$$\vec{f}^{(0)} + \left(\vec{x}^{(1)} - \vec{x}^{(0)}\right) \cdot \nabla \vec{f}^{(0)} = 0. \tag{3.172}$$

Mit der Identifikation $x^{(0)} \to \vec{x}^{(n)}$ und $x^{(1)} \to \vec{x}^{(n+1)}$ ergibt sich in Komponentenschreibweise

$$-f_i^{(n)} = \underbrace{\left(x_j^{(n+1)} - x_j^{(n)}\right)}_{\delta_j^{(n)}} \frac{\partial f_i^{(n)}}{\partial x_j} = \left(\frac{\partial f_i^{(n)}}{\partial x_j}\right) \delta_j^{(n)}. \tag{3.173}$$

Wenn wir die *Jacobi-Matrix* definieren

$$\mathsf{J}^{(n)} = J_{ij}^{(n)} = \left.\frac{\partial f_i}{\partial x_j}\right|_{\vec{x}^{(n)}} = \begin{pmatrix} \dfrac{\partial f_1\left(\vec{x}^{(n)}\right)}{\partial x_1} & \cdots & \dfrac{\partial f_1\left(\vec{x}^{(n)}\right)}{\partial x_I} \\ \vdots & & \vdots \\ \dfrac{\partial f_I\left(\vec{x}^{(n)}\right)}{\partial x_1} & \cdots & \dfrac{\partial f_I\left(\vec{x}^{(n)}\right)}{\partial x_I} \end{pmatrix}, \tag{3.174}$$

können wir (3.173) auch kompakt vektoriell schreiben als

$$\mathsf{J}^{(n)} \cdot \vec{\delta}^{(n)} = -\vec{f}^{(n)}. \tag{3.175}$$

In der Praxis wird man diese Gleichung nie nach $\vec{x}^{(n+1)}$ auflösen, sondern mit dem *Korrekturvektor* $\vec{\delta}^{(n)} = \vec{x}^{(n+1)} - \vec{x}^{(n)}$ rechnen. Der Algorithmus lautet dann:

Newton-Algorithmus

- Berechne $\vec{\delta}^{(n)}$ aus $\mathsf{J}^{(n)} \cdot \vec{\delta}^{(n)} = -\vec{f}^{(n)}$,
- Setze $\vec{x}^{(n+1)} = \vec{x}^{(n)} + \vec{\delta}^{(n)}$

Im Fall der Konvergenz verschwinden sowohl die Korrektur $\vec{\delta} = 0$ als auch das *Residuum* $\vec{f} = 0$.

Hinsichtlich der Konvergenz des mehrdimensionalen Newton-Verfahrens kann man den folgenden Satz beweisen: Ist $\vec{f}(\vec{x})$ in einer Umgebung der Nullstelle $\vec{x}^*$,

d.h. $\vec{f}(\vec{x}^*) = 0$, zweimal stetig differenzierbar und ist $\mathsf{J} = \nabla \vec{f}|_{\vec{x}^*}$ regulär, dann konvergiert die Newton-Iteration (3.175) bzw. (3.173) gegen $\vec{x}^*$, d.h. $\vec{x}^{(n)} \to \vec{x}^*$, falls $\vec{x}^{(0)}$ genügend nahe bei $\vec{x}^*$ liegt. Die Konvergenz ist quadratisch, d.h.,

$$\|\vec{x}^{(n+1)} - \vec{x}^*\|_2 \leq c\|\vec{x}^{(n)} - \vec{x}^*\|_2^2, \tag{3.176}$$

wobei c eine Konstante ist. Hierbei ist wichtig, daß J regulär ist. Ansonsten geht die quadratische Konvergenz verloren, obwohl die Iteration immer noch gegen einen Grenzwert konvergieren kann. Die Bedingung, daß J regulär ist, entspricht im eindimensionalen Fall der Bedingung $f'(x^*) \neq 0$.

In der Praxis genügen meist nur 1–4 Iterationen (vgl. Tab. 3.1). Ein Nachteil der Newton-Iteration besteht darin, daß alle I^2 Elemente der Jacobi-Matrix für jede Iteration neu berechnet werden müssen. Dies wird besonders dann teuer, wenn die Jacobi-Matrix dicht besetzt ist. Nach dem Aufstellen der Jakobi-Matrix muß man außerdem noch in jedem Schritt ein lineares Gleichungssystem lösen.

Wenn man die Jacobi-Matrix nicht analytisch berechnen kann, muß man sie numerisch bestimmen. Eine Option besteht darin, die exakten Ableitungen durch finite Vorwärts-Differenzen zu approximieren

$$\mathsf{J}(\vec{x}) = \frac{\partial f_i(\vec{x})}{\partial x_j} \approx \frac{f_i(\vec{x} + h\vec{e}_j) - f_i(\vec{x})}{h}. \tag{3.177}$$

Hierbei ist h eine kleine Schrittweite und $\vec{e}_j$ der Einheitsvektor in j-Richtung.

Eine weitere Modifikation des Newton-Verfahrens besteht darin, die Jacobi-Matrix nicht in jeden Iterationsschritt neu zu berechnen, sondern für einige Schritte mit derselben Jacobi-Matrix zu rechnen. Dies entspricht der sukzessiven Nullstellenapproximation mit einer konstanten Steigung (vgl. Abb. 3.14). Dieses Verfahren verringert jedoch die Konvergenzrate.

Andere Modifikationen des Newton-Verfahren sind sogenannte *Quasi-Newton-Verfahren* der Form

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} - \omega^{(n)}\mathsf{H}^{(n)} \cdot \vec{f}^{(n)}. \tag{3.178}$$

Hierbei wurde die inverse Jacobi-Matrix $\left[\mathsf{J}^{(n)}\right]^{-1}$ durch eine einfachere Matrix $\mathsf{H}^{(n)}$ und einen Schrittweitenfaktor $\omega^{(n)}$ ersetzt, so daß sie im Limes der Konvergenz mit der inversen Jacobi-Matrix übereinstimmt (Ortega and Rheinboldt, 1970).

Wie für das eindimensionale Newton-Verfahren benötigt man auch für das mehrdimensionale Newton-Verfahren einen guten Anfangswert $\vec{x}^{(0)}$. Liegt der Anfangswert zu weit von $\vec{x}^*$ entfernt, kann ein komplexes Verhalten auftreten. Die Abhängigkeit der gefundenen Nullstelle vom Anfangswert ist in Abb. 3.21 am Beispiel des zweidimensionalen Systems $f(z) = z^3 - 1$ mit $z = x + \mathrm{i}y \in \mathbb{C}$ gezeigt. Aufgespalten in Real- und Imaginärteil gilt

$$f_{\text{real}} = x^3 - 3xy^3 - 1 = 0, \tag{3.179}$$

$$f_{\text{imag}} = 3x^2y - y^3 = 0. \tag{3.180}$$

Die Nullstellen ergeben sich sofort aus $z^3 = 1$ als die drei Einheitswurzeln in der komplexen Ebene $z_1^* = \mathrm{e}^0$, $z_2^* = \mathrm{e}^{2\mathrm{i}\pi/3}$ und $z_3^* = \mathrm{e}^{4\mathrm{i}\pi/3}$.

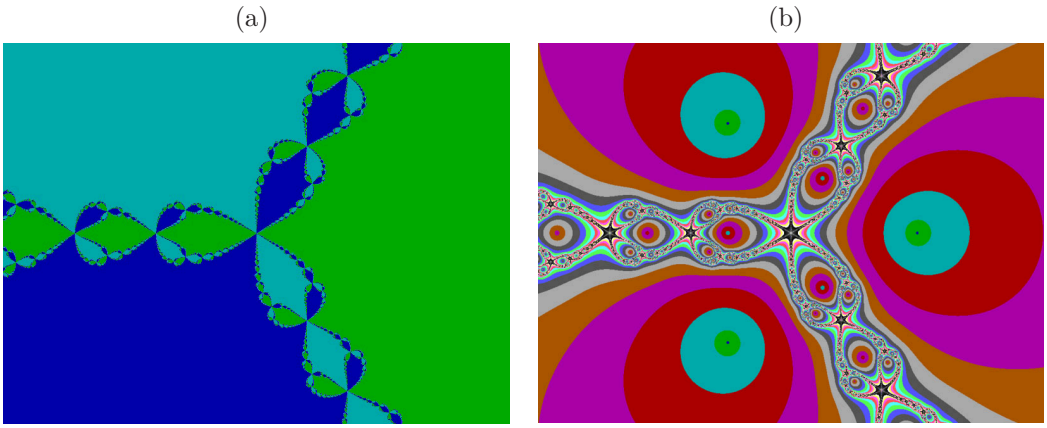


Abbildung 3.21.: Newton-Fraktal für die Newton-Iteration der komplexen Funktion $p(z) = z^3 - 1$. Gezeigt ist die komplexe Ebene der Startwerte z_0. In (a) zeigt die Farbe an, zu welcher der drei Nullstellen die Newton-Iteration konvergiert. Die Grenzen der Attraktionsbereiche der einzelnen Nullstellen sind fraktal. In (b) ist die Farbe ist ein Maß für die Anzahl der Iterationen, die erforderlich sind, um für einen gegebenen Startwert z_0 eine bestimmte Genauigkeit $\|z_n - z_i^*\|_2 < \epsilon$ zu erreichen. Die Abbildungen wurden mit dem Programm FRACTINT erstellt.

3.5.7. Verfolgung einer Lösung bei Parametervariation

In vielen Fällen hängt die Funktion, deren Nullstelle man sucht, noch von mindestens einem unabhängigen Parameter ab. In unserem Beispiel (3.149) ist dies der Parameter λ. Ein nicht-triviales Beispiel ist die Berechnung einer stationären Strömung. Gesucht wird dann das Geschwindigkeitsfeld und seine Abhängigkeit von der Reynoldszahl $\lambda = \mathrm{Re}$. Die Komponenten des unbekannten Vektors $\vec{x}$ bestehen dann aus allen 3 Geschwindigkeitskomponenten an allen Knotenpunkten eines geeigneten Rechengitters. Die Vektorfunktion $\vec{f}(\vec{x}, \mathrm{Re})$, die sich aus der *Navier-Stokes-Gleichung* durch die Diskretisierung ergibt, lebt dann in einem sehr hochdimensionalen Raum, der durch die Komponenten von $\vec{x}$ aufgespannt wird. Die stationären Lösungen entsprechen dann den Nullstellen $\vec{f}(\vec{x}^*, \mathrm{Re}) = 0$ im hochdimensionalen Raum von $\vec{x}$.

Natürliche Parameterfortsetzung

Sei eine Lösung $\vec{x}^*$ für $\lambda = 1$ bekannt. Für sie gilt $\vec{f}(\vec{x}^*, \lambda = 1) = 0$. Dann kann es schwierig sein, mit diesem Startwert $\vec{x}^*(\lambda = 1)$ die Lösung von $\vec{f}(\vec{x}^*, \lambda = 10^3) = 0$ für einen anderen Parameter zu finden, da sich durch die Änderung des Parameters λ (z.B. der Reynoldszahl) die Lösung $\vec{x}^*(\lambda = 10^3)$ zu weit vom ursprünglichen Wert $\vec{x}^*(\lambda = 1)$ entfernt hat. In diesem Fall kann man λ in mehreren hinreichend kleinen Schritten $\Delta\lambda$ erhöhen und in jedem Schritt eine Newton-Iteration durchführen, wobei man als Startwert jeder Newton-Iteration die konvergente Lösung der jeweils vorherigen Newton-Iteration verwendet. Mit dieser Strategie kann man die Lösung

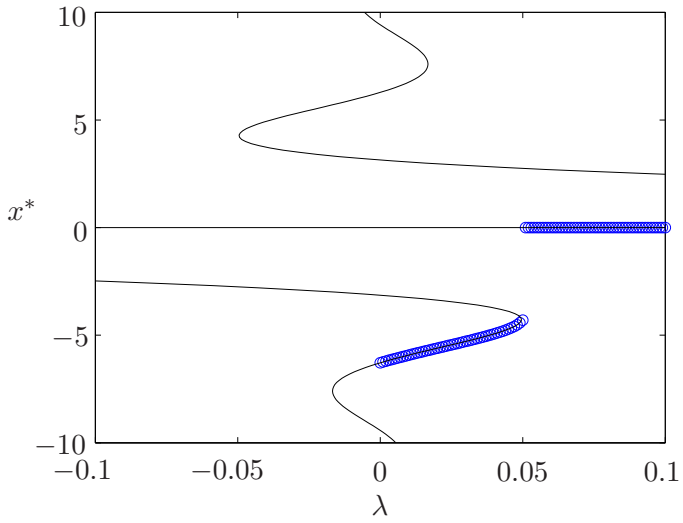


Abbildung 3.22.: Exakte Nullstellen $x^*(\lambda)$ des Problems (3.149) (schwarze Kurven) und Resultat der Newton-Iteration bei natürlicher Parametervariation für ansteigende Werte von λ beginnend bei $\lambda = 0$ (blaue Kreise).

$\vec{x}^*(\lambda)$ als Funktion des Parameters λ verfolgen.

Diese *natürliche Parameterfortsetzung* funktioniert, wenn sich die Lösung $\vec{x}^*(\lambda)$ kontinuierlich als Funktion von λ entwickelt und keine Singularitäten auftreten. Es kann jedoch sein, daß sich die Lösungskurve $\vec{x}^*(\lambda)$ *zurückbiegt* und $x^*(\lambda)$ keine eindeutige Funktion von λ ist. Als Beispiel für dieses Verhalten betrachten wir unser eindimensionales Beispiel (3.149). Die Lösung $x^*(\lambda)$ als Funktion des Parameters λ ist in Abb. 3.22 gezeigt.[13] Wenn man nun einen Lösungsast verfolgt, indem man λ in kleinen Schritten erhöht, wird die Newton-Iteration an dem Punkt, an dem sich der Lösungsast zurückbiegt (*kritischer Punkt*), im günstigsten Fall auf eine andere Lösung springen. Dieses Verhalten ist in Abb. 3.22 dargestellt. Hier springt die Lösung bei Erhöhung von λ von dem nichttrivialen Lösungsast mit $x^* < 0$ auf den Ast $x^* = 0$. Damit hat man aber den ursprünglichen Lösungsast verloren.

Am kritischen Punkt wird bei einer Parametervariation ein Paar neuer Lösungen kreiert oder vernichtet. Dies nennt man auch *Sattel-Knoten-Verzweigung*. Die Situation ist noch einmal schematisch in Abb. 3.23 gezeigt. In solchen Fällen kann man den ursprünglichen Lösungsast mit der natürlichen Parameter-Fortsetzung nicht weiter verfolgen und man muß andere Methoden der Verfolgung verwenden.

Pseudo-Bogenlängen-Fortsetzung

Eine bessere Möglichkeit der Verfolgung von Lösungen ist die *Pseudo-Bogenlängen-Fortsetzung* (*pseudo-arclength continuation*). Dieses Verfahren wurde zuerst von Keller (1977) vorgeschlagen. Hierbei wird der Parameter λ als zusätzliche Variable

[13] Hier ist einfach $\lambda = \sin(x^*)/{x^*}^2$.

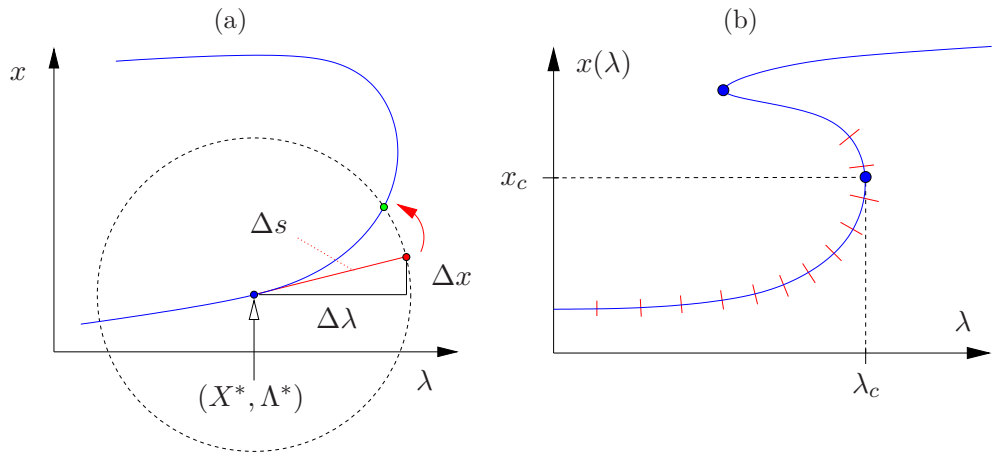


Abbildung 3.23.: Idee der Bogenlängen-Fortsetzung: (a) Ausgehend von einer konvergierten Lösung (X^*, Λ^*) (blauer Punkt) wird von der neuen Newton-Iteration verlangt, daß der Abstand Δs vom der alten konvergierten Lösung konstant bleibt. Dann verläuft die Iteration (angedeutet durch den roten Pfeil) von einer geeigneten ersten Schätzung (roter Punkt) zur neuen Lösung grüner Punkt. Wenn die Schrittweite Δs (rote Linie) hinreichend klein ist, kann der blaue Lösungsast relativ glatt verfolgt werden. (b) Lösung x von $f(x, \lambda) = 0$ als Funktion des Parameters λ. Mit Hilfe der Pseudo-Bogenlängen-Fortsetzung kann man die Lösung auch über kritische Punkte (Sattel-Knoten-Verzweigungen, blaue Punkte) hinaus verfolgen. Die roten Balken symbolisieren die Schritte mit konstanter quasi-Bogenlänge.

aufgefaßt. Im $(I+1)$-dimensionalen Raum $(\vec{x}, \lambda)$ definieren dann die I Gleichungen $\vec{f}(\vec{x}, \lambda) = 0$ eine eindimensionale Untermenge des $\mathbb{R}^{I+1}$: die Lösungskurve $\vec{x}^*(\lambda)$. Um einen Punkt auf dieser Lösungskurve festzulegen, der das Ziel der Newton-Iteration sein soll, brauchen wir eine weitere Gleichung. Diese Gleichung besteht in der Forderung, daß sich der iterierte Punkt $(\vec{x}^{(n)}, \lambda^{(n)})$ im $(\vec{x}, \lambda)$-Raum in einem fest vorgegebenen Abstand Δs (Pseudo-Bogenlänge) von dem vorherigen, iterierten und konvergierten Punkt $(\vec{X}^*, \Lambda^*)$ auf der Lösungskurve befindet (Abb. 3.23). Dann gilt für die Werte von $\vec{x}^{(n)}$ und $\lambda^{(n)}$ die Einschränkung

$$\left\| \vec{x} - \vec{X}^* \right\|_2^2 + (\lambda - \Lambda^*)^2 = (\Delta s)^2 \,. \tag{3.181}$$

Wir fassen nun $\vec{x}$ und λ als die gesuchten Variablen auf und betrachten das erweiterte Nullstellenproblem im $I+1$-dimensionalen Raum

$$\vec{f}\,(\vec{x}, \lambda) = 0, \tag{3.182a}$$

$$g(\vec{x}, \lambda) := (\lambda - \Lambda^*)^2 + \left\| \vec{x} - \vec{X}^* \right\|_2^2 - (\Delta s)^2 = 0. \tag{3.182b}$$

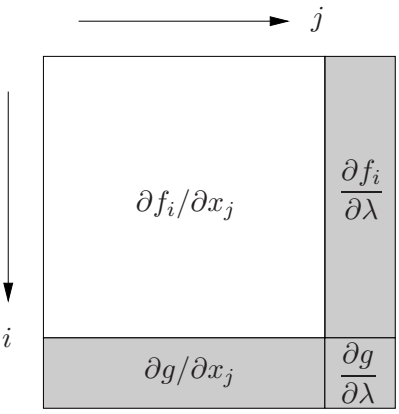


Abbildung 3.24.: Struktur der erweiterten Jacobi-Matrix bei Verwendung der Pseudo-Bogenlängen-Fortsetzung.

Die Newton-Iteration für dieses erweiterte System können wir dann formulieren als (vgl. den Newton-Algorithmus auf Seite 64)

$$\underbrace{\begin{pmatrix} \frac{\partial f_i}{\partial x_j} & \frac{\partial f_i}{\partial \lambda} \\ \frac{\partial g}{\partial x_j} & \frac{\partial g}{\partial \lambda} \end{pmatrix}^{(n)}}_{\text{Jacobi-Matrix}} \cdot \underbrace{\begin{pmatrix} x_j^{(n+1)} - x_j^{(n)} \\ \lambda^{(n+1)} - \lambda^{(n)} \end{pmatrix}}_{\text{Korrektur}} = - \begin{pmatrix} f_i \\ g \end{pmatrix}^{(n)} . \tag{3.183}$$

Wenn wir die partiellen Ableitungen von g ausführen (Summenkonvention)

$$\left.\frac{\partial g}{\partial x_j}\right|^{(n)} = 2\left(x_i - X_i^*\right) \underbrace{\left.\frac{\partial x_i}{\partial x_j}\right|^{(n)}}_{=\delta_{i,j}} = 2\left(x_j - X_j^*\right)^{(n)} = 2\left(x_j^{(n)} - X_j^*\right), \tag{3.184a}$$

$$\left.\frac{\partial g}{\partial \lambda}\right|^{(n)} = 2\left(\lambda - \Lambda^*\right)^{(n)} = 2\left(\lambda^{(n)} - \Lambda^*\right), \tag{3.184b}$$

erhalten wir

$$\begin{bmatrix} \frac{\partial f_i}{\partial x_j} & \frac{\partial f_i}{\partial \lambda} \\ 2\left(x_j - X_j^*\right) & 2\left(\lambda - \Lambda^*\right) \end{bmatrix}^{(n)} \cdot \begin{pmatrix} x_j^{(n+1)} - x_j^{(n)} \\ \lambda^{(n+1)} - \lambda^{(n)} \end{pmatrix} = - \begin{pmatrix} f_i \\ g \end{pmatrix}^{(n)} . \tag{3.185}$$

Die Struktur der erweiterten Jacobi Matrix ist in Abb. 3.24 illustriert. Beachte, daß die reduzierte Jacobi-Matrix $\partial f_i/\partial x_j$ am kritischen Punkt singulär wird. Dies liegt daran, daß sich am kritischen Punkt $\lambda = \lambda_c$ bei Variation von λ zwei Nullstellen gegenseitig vernichten. Dies geschieht in unserem Beispiel bei $\lambda \approx 0.05$ (Abb. 3.22). Gleichung (3.185) muß nun iteriert werden. Dazu benötigen wir möglichst gute Startwerte.

Vorhersage-Schritt

Einen guten Startwert $\left(\vec{x}^{(0)}, \lambda^{(0)}\right)$ (roter Punkt in Abb. 3.23a bzw. rote Quadrate in Abb. 3.25) kann man erhalten, indem man von dem konvergenten Punkt $(\vec{X}^*, \Lambda^*)$

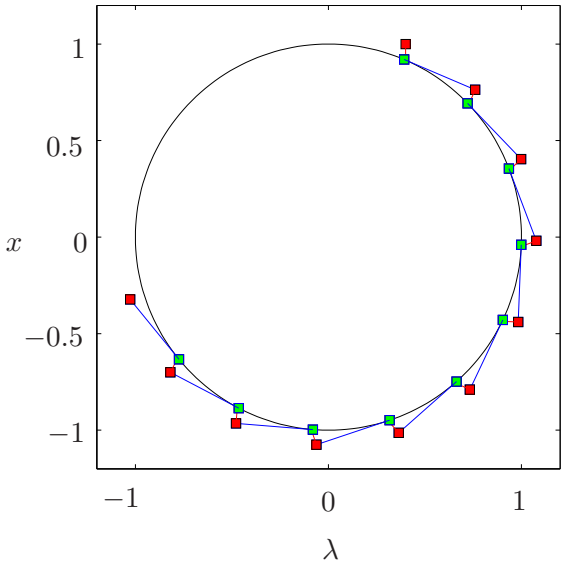


Abbildung 3.25.: Quasi-Bogenlängen-Verfolgung mit $\Delta s = 0.4$ am Beispiel der Funktion $f(x,\lambda) = x^2 + \lambda^2 - 1$. Die Nullstelle $f(x,\lambda) = 0$ ist als schwarzer Kreis dargestellt. Die Anfangswerte einer jeden Newton-Iteration sind die roten Punkte. Der Verlauf jeder Newton Iteration ist als rote Linie gezeigt. Die konvergierten Nullstellen sind mit grünen Punkten symbolisiert. Die erste Iterierte liegt schon innerhalb der Größe der grünen Symbole. Als Abbruchkriterium wurde $\epsilon = 10^{-10}$ gewählt, was hier typischerweise 5 Newton-Iterationen erforderte.

auf dem Lösungsast $\vec{f}(\vec{x},\lambda) = 0$ ausgeht (blauer Punkt in Abb. 3.23a bzw. grüne Quadrate in Abb. 3.25) und einen Schritt der Länge Δs tangential zum Lösungsast $\vec{f}(\vec{x},\lambda) = 0$ vorwärts geht (rote Linie in Abb. 3.23a bzw. blaue Linien in Abb. 3.25). Dazu muß man im $(I+1)$-dimensionalen Raum (I Variablen $\vec{x}$ und 1 Parameter λ) die tangentiale Richtung an die Lösungskurve $\vec{f} = 0$ im Punkte $(\vec{X}^*, \Lambda^*)$ berechnen. Sei der Tangentialvektor $\vec{T} = \left(\vec{t}, t^{(\lambda)}\right)^{\mathrm{T}}$. Um seine Komponenten zu bestimmen, beachten wir, daß die tangentiale Richtung dadurch gekennzeichnet ist, daß sich $\vec{f}$ in Richtung von $\vec{T}$ nicht ändert. Die Änderungsrate irgendeiner Funktion in $\vec{T}$-Richtung ist proportional zu $\vec{T}\cdot\tilde{\nabla}$ wobei $\tilde{\nabla} = (\nabla, \partial_\lambda)^{\mathrm{T}}$ der Nabla-Operator im $(I+1)$-dimensionalen Raum ist.[14]

Daher fordern wir vom Tangentenvektor $\vec{T} = \left(\vec{t}, t^{(\lambda)}\right)^{\mathrm{T}}$ im $(I+1)$-dimensionalen Raum

$$\vec{T}\cdot\tilde{\nabla}\vec{f} = \vec{t}\cdot\nabla\vec{f} + t^{(\lambda)}\partial_\lambda\vec{f} \stackrel{!}{=} 0. \tag{3.186}$$

Dies sind I Bedingungen für die $I+1$ Komponenten von $\vec{T}$. Der Tangentenvektor ist durch (3.186) nicht eindeutig bestimmt, da sein Betrag noch frei ist. Deshalb kann man zum Beispiel $T_1 = 1$ setzen und so (3.186) eindeutig lösen. Da wir jedoch

[14] Wenn $\|\vec{T}\|_2 = 1$ auf eins normiert ist, dann ist die Richtungsableitung in $\vec{T}$-Richtung gegeben durch $\vec{T}\cdot\tilde{\nabla}$.

um die Länge Δs in $\vec{T}$-Richtung gehen wollen, setzen wir durch die Normierung

$$\vec{T} \longrightarrow \frac{\vec{T}}{\left\|\vec{T}\right\|_2}\Delta s \tag{3.187}$$

die Länge von $\vec{T}$ auf Δs fest. Wenn man $\vec{T} = \left(\vec{t}, t^{(\lambda)}\right)^{\mathrm{T}}$ auf diese Weise bestimmt hat, wird die nächste Newton-Iteration nach (3.185) mit dem Startwert

$$\begin{pmatrix}\vec{x}^{(0)} \\ \lambda^{(0)}\end{pmatrix} = \begin{pmatrix}\vec{X}^* \\ \Lambda^*\end{pmatrix} + \begin{pmatrix}\vec{t} \\ t^{(\lambda)}\end{pmatrix} \tag{3.188}$$

initialisiert.

Beispiel Als Beispiel betrachten wir

$$f(x, \lambda) = x^2 + \lambda^2 - 1. \tag{3.189}$$

Die gesuchte Lösung von $f(x, \lambda) = 0$ ist der schwarze Kreis in Abb. 3.25. Ausgehend von der bekannten Lösung $(X^*, \Lambda^*) = (1, 0)$, einer Pseudo-Bogenlänge $\Delta s = 0.4$ und einer Anfangsschätzung $\left(x^{(0)}, \lambda^{(0)}\right) = (1, 0.4)$ (erster roter Punkt in Abb. 3.25) wird nach (3.185) iteriert. Die Jacobi-Matrix aus (3.185) lautet dann

$$\begin{bmatrix} \dfrac{\partial f_i}{\partial x_j} & \dfrac{\partial f_i}{\partial \lambda} \\ 2\left(x_j - X_j^*\right) & 2\left(\lambda - \Lambda^*\right) \end{bmatrix}^{(n)} \rightarrow \begin{bmatrix} 2x & 2\lambda \\ 2\left(x - X^*\right) & 2\left(\lambda - \Lambda^*\right) \end{bmatrix}^{(n)}. \tag{3.190}$$

Die Iteration wird als konvergent erachtet, wenn für die Korrektur von x und λ gilt $\left(x^{(n+1)} - x^{(n)}\right)^2 + \left(\lambda^{(n+1)} - \lambda^{(n)}\right)^2 \leq \epsilon^2$ (erster grüner Punkt in Abb. 3.25). Damit haben wir unsere neue Nullstelle (X^*, Λ^*) berechnet.

Nun benötigen wir einen neuen Vorhersage-Schritt. Dafür betrachten wir die eindimensionale Version von (3.186)

$$t^{(x)} \left.\frac{\partial f}{\partial x}\right|_{X^*,\Lambda^*} + t^{(\lambda)} \left.\frac{\partial f}{\partial \lambda}\right|_{X^*,\Lambda^*} = 0. \tag{3.191}$$

Wenn wir willkürlich $t^{(x)} = 1$ setzen, ergibt sich

$$t^{(\lambda)} = -\left(\frac{\partial f/\partial x}{\partial f/\partial \lambda}\right)_{X^*,\Lambda^*}. \tag{3.192}$$

Die Länge des Tangentialvektors ist dann

$$N = \left\|\vec{T}\right\|_2 = \sqrt{\left(t^{(x)}\right)^2 + \left(t^{(\lambda)}\right)^2} = \sqrt{1 + \left(\frac{\partial f/\partial x}{\partial f/\partial \lambda}\right)^2_{X^*,\Lambda^*}}. \tag{3.193}$$

Damit erhalten wir einen Tangentialvektor der Länge Δs mit den Komponenten

$$t^{(x)} = \frac{\Delta s}{N}, \tag{3.194a}$$

$$t^{(\lambda)} = -\left(\frac{\partial f/\partial x}{\partial f/\partial \lambda}\right)_{X^*,\Lambda^*} \frac{\Delta s}{N} \overset{(3.190)}{=} -\frac{X^*}{\Lambda^*}\frac{\Delta s}{N}. \tag{3.194b}$$

In unserem Beispiel muß man noch $t^{(x)} \to -\text{sign}\,(\partial f/\partial\lambda)\, t^{(x)}$ einfügen, damit die Lösung wie in Abb. 3.25 im Uhrzeigersinn durchlaufen wird.

Man kann nun mit $\left(x^{(0)}, \lambda^{(0)}\right) = \left(X^* + t^{(x)}, \Lambda^* + t^{(\lambda)}\right)$ (zweiter roter Punkt in Abb. 3.25) die nächste Newton-Iteration starten. Damit ist es möglich, die gesamte Lösung (Kreis) zu verfolgen, obwohl die Lösung $x^*(\lambda)$ als Funktion von λ nicht eindeutig ist.

Man erkennt leicht, daß die Quasi-Bogenlänge $s = n\Delta s$ der wirklichen Bogenlänge der Lösungskurve umso näher kommt, desto geringer die Schrittweite Δs gewählt wird.

4. Numerische Integration

In vielen Fällen möchte man das bestimmte Integral

$$I(f) = \int_a^b f(x)\,\mathrm{d}x \tag{4.195}$$

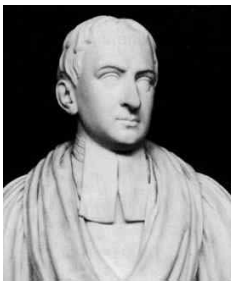
näherungsweise bestimmen. Dies kann notwendig werden, wenn die Berechnung der exakten Lösung zu aufwendig ist oder wenn die exakte Lösung nicht in geschlossener Form dargestellt werden kann. Ein Beispiel ist

$$\int_0^\pi \cos(4x)\cos\left[3\sin(x)\right]\,\mathrm{d}x = \pi\left(\frac{3}{2}\right)^4 \sum_{i=0}^{\infty} \frac{(-9/4)^i}{i!(i+4)!}. \tag{4.196}$$

Eine Näherung des exakten Ergebnisses erhält man, wenn man die unendliche Reihe bei $i = n$ abbricht.

Bei der numerischen Integration sucht man typischerweise Näherungen der Form

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} a_i f(x_i), \tag{4.197}$$



Roger Cotes
1682–1716

wobei $n+1$ Stützpunkte $x_i \in [a, b]$ innerhalb des abgeschlossenen Intervalls verwendet werden und a_i zugehörige Gewichtskoeffizienten sind. Damit hat man das kontinuierliche Problem (4.195) in ein diskretes Problem (4.197) überführt (diskretisiert). Diese Art der Approximation wird *Quadratur* genannt.

Zur Bestimmung der zu x_i gehörigen Koeffizienten a_i wird die Funktion $f(x)$ unter dem Integral in (4.197) typischerweise durch andere Funktionen $p(x)$ ersetzt, die an den vorab gewählten Stützstellen x_i mit der Funktion f identisch sind $[p(x_i) = f(x_i)]$, aber leicht integriert werden können. Die Koeffizienten a_i ergeben sich dann aus Forderungen der Art $\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b p(x)\,\mathrm{d}x \stackrel{!}{=} \sum_{i=0}^n a_i p(x_i) = \sum_{i=0}^n a_i f(x_i)$ für möglichst viele Funktionen $p(x)$. Für $p(x)$ werden oft Polynome oder Exponentialfunktionen verwendet.

4.1. Newton-Cotes-Formeln

Die einfachste Möglichkeit besteht darin, $n + 1 = 1$ zu wählen und die Funktion f durch eine Konstante zu ersetzen, die mit dem Funktionswert am linken Rand $f(a)$

übereinstimmt (man kann auch $f(b)$ nehmen). Dann erhält man die *Rechteck-Regel*

$$\int_a^b f(x)\mathrm{d}x \approx \int_a^b f(a)\mathrm{d}x = (b-a)f(a). \tag{4.198}$$

Als Stützpunkt kann man auch einen Wert aus dem Innern des Intervalls verwenden. Meist wird der Mittelpunkt $(a+b)/2$ genommen. Man gewinnt dann die *Mittelpunktsregel* (*mid-point rule*)

$$\int_a^b f(x)\mathrm{d}x \approx \int_a^b f\left(\frac{a+b}{2}\right)\mathrm{d}x = (b-a)f\left(\frac{b+a}{2}\right). \tag{4.199}$$

Ein besseres Ergebnis erhält man, wenn man zwei Stützpunkte $(n+1=2)$ wählt und die Funktion durch eine lineare Funktion approximiert, die durch die beiden Punkte $f(a)$ und $f(b)$ geht. Dies führt auf die *Trapez-Regel*

$$\int_a^b f(x)\mathrm{d}x \approx \int_a^b \left[f(a) + \frac{f(b)-f(a)}{b-a}(x-a)\right]\mathrm{d}x = (b-a)\frac{f(a)+f(b)}{2}. \tag{4.200}$$

Hierbei wird das Integral durch eine Fläche in Trapez-Form ersetzt. Als weitere Verbesserung ist es naheliegend, ein interpolierendes Polynom zweiten Grades zu verwenden, das mit den $n+1=3$ Stützpunkten a, b und dem Mittelpunkt $(a+b)/2$ mit den Funktionswerten von f übereinstimmt. Man findet dann die *Simpson-Regel*[1]

Thomas Simpson
1710–1761

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{(b-a)}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{4.201}$$

Die genannten Approximationen sind in Abb. 4.1 graphisch dargestellt.

Man kann nun so weitermachen und das Intervall $[a,b]$ in n Teilintervalle unterteilen. Inklusive a und b hat man dann $n+1$ Stützpunkte x_i. Eine Möglichkeit, die $n+1$ unbekannten Koeffizienten a_i in der Quadratur-Formel zu bestimmen, besteht darin, zu verlangen, daß die Näherung (4.197) exakt erfüllt ist, wenn die zu integrierende Funktion $f(x)$ ein beliebiges Polynom vom maximalen Grade n ist. Am einfachsten wählt man einfache Polynome $f = x^j$ mit $j = 0,\ldots,n$. Dann lauten die $n+1$ Bedingungen für die Koeffizienten a_i zu den Stützstellen x_i

$$\int_a^b x^j\,\mathrm{d}x = \frac{b^{j+1}-a^{j+1}}{j+1} \stackrel{!}{=} \sum_{i=0}^{n} a_i x_i^j, \qquad j = 0,1,\ldots,n. \tag{4.202}$$

[1] Thomas Simpson (1710–1761): Englischer Mathematiker. Neben der nach ihm benannten Quadraturformel gehen die heutzutage üblichen Bezeichnungen Sinus, Cosinus, Tangens und Cotangens auf ihn zurück sowie auch die differentielle Form des Newton-Verfahrens.

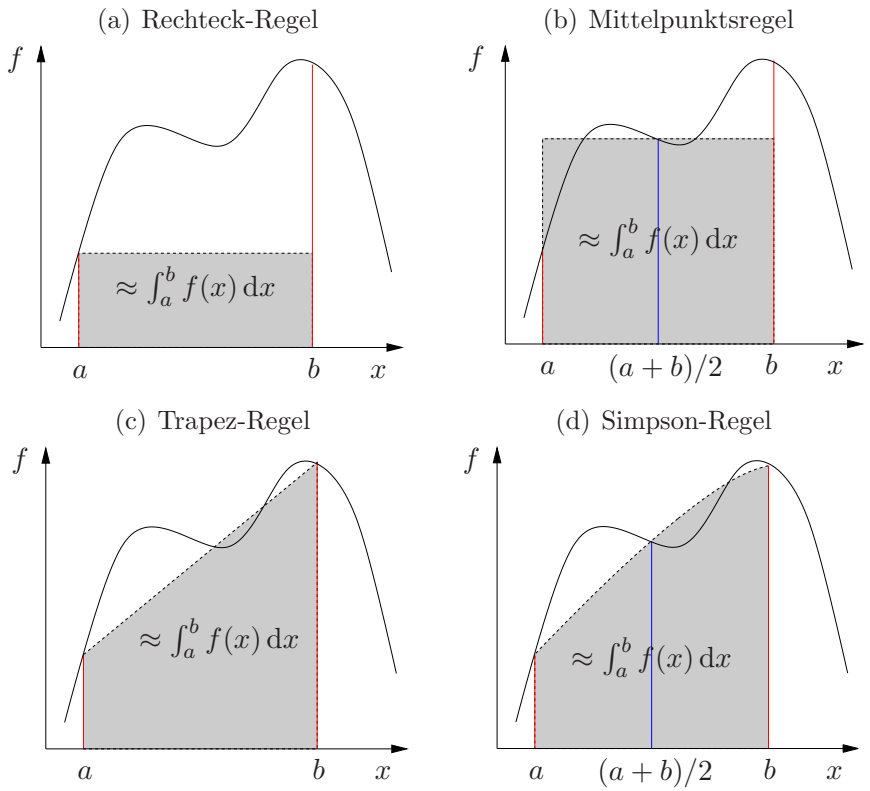


Abbildung 4.1.: Approximation des bestimmten Integrals durch Approximation des Integranden mittels verschiedener Interpolationen. Die quadratische Interpolation in (d) ist nur schematisch gezeigt.

Diese $n+1$ Bedingungen führen auf das lineare Problem

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ x_0 & x_1 & x_2 & & x_n \\ \vdots & & & \vdots & \\ x_0^n & x_1^n & x_2^n & \dots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} b-a \\ (b^2-a^2)/2 \\ \vdots \\ (b^{n+1}-a^{n+1})/(n+1) \end{pmatrix}, \tag{4.203}$$

wobei die Matrix die schon bekannte Vandermondesche Gestalt (3.90) (transponiert) hat. Sie ist regulär, wenn die Punkte x_i paarweise verschieden sind. Wenn man die Punkte x_i darüber hinaus auch noch äquidistant wählt, nennt man die resultierende Formel (4.197) *Newton-Cotes-Formel*.

Eine alternative Möglichkeit, die Koeffizienten a_i zu bestimmen (in Analogie zu den oben beschriebenen Trapez- und Simpson-Regeln), besteht darin, $f(x)$ durch die Lagrange-Interpolation (3.95) der n Punkte $[x_i, f(x_i)]$ zu approximieren

$$f(x) \approx p(x) = \sum_{i=0}^{n} f(x_i)\, L_i^{(n)}(x), \tag{4.204}$$

mit den Lagrange-Polynomen vom Grade n

$$L_i^{(n)}(x) = \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{x - x_k}{x_i - x_k}, \qquad i = 0, 1, \ldots, n. \tag{4.205}$$

Das Integral wird dann approximiert als

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b p(x)\,\mathrm{d}x = \int_a^b \sum_{i=0}^{n} f(x_i)\, L_i^{(n)}(x)\,\mathrm{d}x = \sum_{i=0}^{n} f(x_i) \underbrace{\int_a^b L_i^{(n)}(x)\,\mathrm{d}x}_{a_i}. \tag{4.206}$$

Hieran kann man die Gewichte a_i direkt ablesen. Sie sind identisch mit denjenigen aus (4.203), denn die Polynom-Interpolation ist eindeutig.

Als Beispiel betrachten wir $n = 1$ mit $x_0 = a$ und $x_1 = b$ und den zugehörigen Funktionswerten $f_0 = f(a)$ und $f_1 = f(b)$. Die beiden Lagrangeschen Polynome sind

$$L_0^{(1)} = \frac{x - b}{a - b} \qquad \text{und} \qquad L_1^{(1)} = \frac{x - a}{b - a}. \tag{4.207}$$

Damit erhalten wir die Approximation von $f(x)$

$$p(x) = L_0^{(1)}(x) f_0 + L_1^{(1)}(x) f_1. \tag{4.208}$$

Für die Gewichte der Quadraturformel erhält man dann nach (4.206)

$$a_0 = \int_a^b L_0^{(1)}(x)\,\mathrm{d}x = \int_a^b \frac{x - b}{a - b}\mathrm{d}x = \left[\frac{1}{2}\frac{(x - b)^2}{a - b}\right]_a^b = \frac{b - a}{2}. \tag{4.209}$$

Analog erhält man $a_1 = a_0 = (b - a)/2$. Dies liefert genau die Trapezregel (4.200).

4.2. Summierte Formeln

An Abb. 4.1 ist ersichtlich, daß die Newton-Cotes-Formeln für Polynome niedriger Ordnung sehr fehlerbehaftet sein können. Daher könnte man versuchen, bessere Approximationen zu erhalten, indem man den Grad der Polynome sukzessive erhöht. Dieses Verfahren ist aber nicht praktikabel, da die Vandermonde-Matrix schlecht konditioniert ist. Ab der Ordnung $n \geq 8$ können auch negative Koeffizienten a_i auftreten, was zu Auslöschungseffekten führen kann (siehe Kap. 1.2). Außerdem ist nicht sichergestellt, daß die Folge der Approximationen für $n \to \infty$ tatsächlich gegen das gesuchte Integral konvergiert.[2]

Aus diesem Grund ist es meist wesentlich besser, das Intervall $[a, b]$, über welches integriert wird, in viele Teilintervalle $[x_{i-1}, x_i]$ mit $x_0 = a$ und $x_n = b$ zu zerlegen

[2] Siehe die Problematik der Punkteverteilung innerhalb des Intervalls bei der Lagrange-Interpolation; Kap. 3.2.1.

und die Polynom-Approximation stückweise zu verwenden, wobei die Ordnung der Polynome relativ niedrig sein kann. Dazu schreibt man

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x. \tag{4.210}$$

Wenn man nun die Integrale über die Teilintervalle mittels Rechteck-Regel approximiert, erhält man die *summierte Rechteck-Regel* (SR)

$$I_{\mathrm{SR}}(f) = \sum_{i=1}^{n} h_i f(x_{i-1}), \tag{4.211a}$$

wobei $h_i = x_i - x_{i-1}$ die Länge des i-ten Teilintervalls bezeichnet. In analoger Weise erhält man die summierten Mittelpunkts- (SM), Trapez- (ST) und Simpson-Regeln (SS)

$$I_{\mathrm{SM}}(f) = \sum_{i=1}^{n} h_i f\left(\frac{x_{i-1}+x_i}{2}\right), \tag{4.211b}$$

$$I_{\mathrm{ST}}(f) = \sum_{i=1}^{n} \frac{h_i}{2}\left[f(x_{i-1}) + f(x_i)\right], \tag{4.211c}$$

$$I_{\mathrm{SS}}(f) = \sum_{i=1}^{n} \frac{h_i}{6}\left[f(x_{i-1}) + 4f\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_i)\right]. \tag{4.211d}$$

Bei diesen Approximationen wurde der Integrand durch stückweise konstante (SR,SM), lineare (ST) oder quadratische Polynome (SS) approximiert.

4.3. Fehlerbetrachtungen

Bei der Approximation der Integrale treten zwei Fehlerquellen auf. Einerseits ist dies der Fehler, den man bei der Approximation der Funktion durch ein Polynom macht (Diskretisierungsfehler). Hinzu kommt andererseits der Rundungsfehler, der durch die diskrete Darstellung der Zahlen und die diskreten Rechenoperationen entsteht.

4.3.1. Diskretisierungsfehler

Der *Diskretisierungsfehler* lautet

$$E = \int_a^b \left[f(x) - p(x)\right]\,\mathrm{d}x. \tag{4.212}$$

Diesen Fehler können wir mit Hilfe des Fehlers der Polynom-Interpolation von Grade n (3.106) schreiben als

$$E = \frac{1}{(n+1)!}\int_a^b (x-x_0)\ldots(x-x_n)\, f^{(n+1)}\left[z(x)\right]\,\mathrm{d}x, \tag{4.213}$$

wobei $x_0, \ldots, x_n$ die Interpolationspunkte sind und $z(x) \in [a,b]$ eine von x abhängige Zwischenstelle ist.

Für die Rechteck-Regel (R) ($n = 0$) können wir den Diskretisierungsfehler betragsmäßig abschätzen (das Approximationspolynom ist hier vom Grade Null)

$$|E_\mathrm{R}| = \left| \int_a^b (x-a) f'\left[z(x)\right] \,\mathrm{d}x \right| \le \underbrace{\max_z |f'(z)|}_{M_1} \int_a^b |x-a| \,\mathrm{d}x = \frac{M_1}{2} (b-a)^2 \,. \tag{4.214}$$

Für die Mittelpunktsregel (M) erhält man mit $x_0 = (a+b)/2$

$$|E_\mathrm{M}| = \left| \int_a^b \left(x - \frac{a+b}{2} \right) f'\left[z(x)\right] \,\mathrm{d}x \right| \le M_1 \int_a^b \left| x - \frac{a+b}{2} \right| \mathrm{d}x = \frac{M_1}{4} (b-a)^2 \,, \tag{4.215}$$

ähnlich wie für die Rechteck-Regel. Bei der Mittelpunktsregel ist diese Abschätzung allerdings zu grob. Denn man kann den Diskretisierungsfehler zu $M_2 (b-a)^3/24$ abschätzen.[3]

Für die Trapez-Regel (T) sind die Interpolationspunkte $x = a$ und $x = b$ und wir erhalten die Abschätzung (das Approximationspolynom ist hier vom Grade 1)

$$|E_\mathrm{T}| = \frac{1}{2} \left| \int_a^b (x-a)(x-b) f''\left[z(x)\right] \mathrm{d}x \right| \le M_2 \int_a^b |(x-a)(x-b)| \,\mathrm{d}x = \frac{M_2}{12} (b-a)^3 \,, \tag{4.216}$$

wobei $M_2 = \max_z |f''(z)|$ (für den Integranden gilt $(x-a)(x-b) \le 0$).

[3] Eine bessere Abschätzung des Fehlers der Mittelpunktsregel erhält man, wenn man den Integranden $f(x) - p(x)$ im Fehler (4.212) in eine Taylor-Reihe um den Mittelpunkt $x_m = (a+b)/2$ des Intervalls entwickelt. Mit $p(x) = f(x_m) = \text{const.}$ erhalten wir

$$-p(x) + f(x) = \underbrace{-f(x_m) + f(x_m)}_{} + (x - x_m) f'(x_m) + \frac{1}{2}(x-x_m)^2 f''(x_m) + \ldots$$

Damit erhalten wir den Betrag des Diskretisierungsfehlers

$$\begin{aligned}
|E_\mathrm{M}| &= \left| \int_a^b \left[(x-x_m) f'(x_m) + \frac{1}{2}(x-x_m)^2 f''(x_m) + \ldots \right] \mathrm{d}x \right| \\
&\le \left| \int_a^b (x-x_m) f'(x_m) \,\mathrm{d}x \right| + \frac{1}{2} \left| \int_a^b (x-x_m)^2 f''(x_m) \,\mathrm{d}x \right| + \ldots \\
&\le |f'(x_m)| \left| \underbrace{\int_a^b (x-x_m) \,\mathrm{d}x}_{=0} \right| + \frac{1}{2} \underbrace{|f''(x_m)|}_{\le M_2} \int_a^b (x-x_m)^2 \,\mathrm{d}x + \ldots \\
&\le \frac{M_2}{24} (b-a)^3 + \ldots .
\end{aligned}$$

Die durch … angedeuteten Terme sind für $b - a \ll 1$ zu vernachlässigen. Der Fehler ist also geringer (höhere Potenz von $(b-a)$) als derjenige, den man durch die einfache Abschätzung (4.215) erhält.

Für die Simpson-Regel erhält man in ähnlicher Weise die Fehlerabschätzung

$$|E_{\mathrm{S}}| \leq \frac{M_4}{2880}\,(b-a)^5\,, \tag{4.217}$$

wobei M_4 eine obere Schranke für den Betrag der vierten Ableitung ist.

Bei all diesen Abschätzungen geht eine Potenz der Intervallänge $(b-a)^k$ ein. Wenn $b-a = O(1)$ groß ist, wird auch der Fehler groß sein. Wenn aber $b-a \ll 1$ klein ist gegenüber 1, dann wird der Fehler umso kleiner, je höher die Potenz k ist. Dies ist der Fall bei den summierten Formeln, für welche ja eine sehr feine Intervallunterteilung angestrebt wird. Jedoch addieren sich bei der Summenbildung auch die Fehler, so daß wir für die summierte Rechteck-Regel beispielsweise den Fehler

$$|E_{\mathrm{SR}}| \leq \frac{M_1}{2}\sum_{i=1}^{n} h_i^2 \tag{4.218}$$

erhalten, wobei $h_i = x_i - x_{i-1}$ die i-te Intervallänge ist. Hierbei haben wir M_1 über das gesamte Intervall $[a,b]$ maximiert, obwohl es eine günstigere Abschätzung gäbe, wenn wir die Abschätzung intervallweise durchgeführt hätten. Für die Fehlerordnung spielt dies aber keine Rolle.

Im wichtigen Spezialfall einer homogenen Unterteilung $h_i = h = (b-a)/n$ des Gesamtintervalls erhalten wir

$$|E_{\mathrm{SR}}| \leq \frac{M_1}{2}(b-a)h = O\left(h\right). \tag{4.219a}$$

Der Fehler der summierten Rechteck-Regel ist also von erster Ordnung in h. Man sagt auch: Die *Fehlerordnung* ist $O(h^1)$. Dies bedeutet, daß der Fehler bei einer Verringerung der Länge h der Teilintervalle linear mit h kleiner wird. Für äquidistante Intervallunterteilungen erhält man in analoger Weise die Fehler der summierten Mittelpunkts-, Trapez- und Simpson-Regeln zu

$$|E_{\mathrm{SM}}| \leq \frac{M_2}{24}(b-a)h^2 = O\left(h^2\right), \tag{4.219b}$$

$$|E_{\mathrm{ST}}| \leq \frac{M_2}{12}(b-a)h^2 = O\left(h^2\right), \tag{4.219c}$$

$$|E_{\mathrm{SS}}| \leq \frac{M_4}{2880}(b-a)h^4 = O\left(h^4\right). \tag{4.219d}$$

Die Simpson-Formel ist noch relativ einfach, aber trotzdem schon von vierter Ordnung. Daher wird sie häufig verwendet.

Die Konstanz der Sub-Intervallänge h stellt eine gewisse Einschränkung dar. Denn es gibt Integranden, die in einem Gebiet stark variieren und in anderen Gebieten nur langsam mit x variieren. Dann kann es sinnvoll sein, die Intervallänge adaptiv zu wählen. Eine mögliche Strategie besteht darin, die Intervallänge h zu halbieren und das Integral über h mit der Summe der beiden Integrale über die Teilintervalle mit jeweils $h/2$ zu vergleichen. Falls der Unterschied kleiner ist als der

tolerierte Fehler, dann werden die beiden Intervalle nicht weiter verfeinert. Andernfalls werden die Intervalle der Länge $h/2$ wieder halbiert, jeweils auf die Länge $h/4$. Dieser Prozeß wird fortgesetzt, bis man überall die gewünschte lokale Genauigkeit erreicht hat.

4.3.2. Rundungsfehler

Falls wir eine exakte Zahlendarstellung hätten, könnten wir ein Integral beliebig genau berechnen. Die Computer-Arithmetik stellt die Zahlen jedoch mit einer endlichen Genauigkeit dar. Wenn wir den Rundungsfehler von $f(x_i)$ mit ϵ_i bezeichnen und den Fehler der kombinierten numerischen Addition und Multiplikation mit η_i, dann gilt zum Beispiel für die summierte Trapez-Regel (4.211c)

$$I_{\mathrm{ST}}^{\mathrm{num}} = \sum_{i=1}^{n} \left\{ \frac{h}{2} \left[f\left(x_{i-1}\right) + \epsilon_{i-1} + f\left(x_i\right) + \epsilon_i \right] + \eta_i \right\} . \tag{4.220}$$

Für $h \to 0$ strebt der Diskretisierungsfehler gegen Null ($E_{\mathrm{ST}} \sim h^2$), aber der summierte Rundungsfehler durch ϵ_i und η_i steigt an. Wenn wir die Summen separat nehmen, können wir schreiben

$$I_{\mathrm{ST}}^{\mathrm{num}} = \underbrace{\sum_{i=1}^{n} \frac{h}{2} \left[f\left(x_{i-1}\right) + f\left(x_i\right) \right]}_{I_{\mathrm{ST}}} + \underbrace{\sum_{i=1}^{n} \frac{h}{2} \left(\epsilon_{i-1} + \epsilon_i \right) + \sum_{i=1}^{n} \eta_i}_{E} . \tag{4.221}$$

Damit können wir den Betrag des Fehlers $|E|$ folgendermaßen abschätzen (ungünstigster Fall)

$$|E| = |I_{\mathrm{ST}}^{\mathrm{num}} - I_{\mathrm{ST}}| \leq \sum_{i=1}^{n} h \max |\epsilon_i| + \sum_{i=1}^{n} \max |\eta_i| . \tag{4.222}$$

Mit $h = (b-a)/n$ ist dann

$$|E| \leq \underbrace{(b-a) \max |\epsilon_i|}_{\text{beschränkt}} + \underbrace{\overbrace{\frac{b-a}{h}}^{n} \max |\eta_i|}_{\substack{\to\infty \\ \text{für } h\to 0}} . \tag{4.223}$$

Der Fehler, der von den diskreten Operationen (Multiplikation und Addition) herrührt, kann für $h \to 0$ im ungünstigsten Fall $\sim h^{-1}$ akkumulieren. Daher muß der gesamte Fehler (Diskretisierungsfehler und Rundungsfehler) wie in Abb. 4.2 schematisch dargestellt von h abhängen. Das Minimum des Fehlers liegt meist bei sehr viel kleineren Werten von h, als sie in der Praxis verwendet werden.[4]

[4] Squire and Trapp (1998) beschreiben eine Methode zur Approximation von Ableitungen einer reellen Funktion mit Hilfe komplexer Variablen, wodurch die fatale Auslöschung (subtraktiver Auslöschungsfehler) vermieden wird.

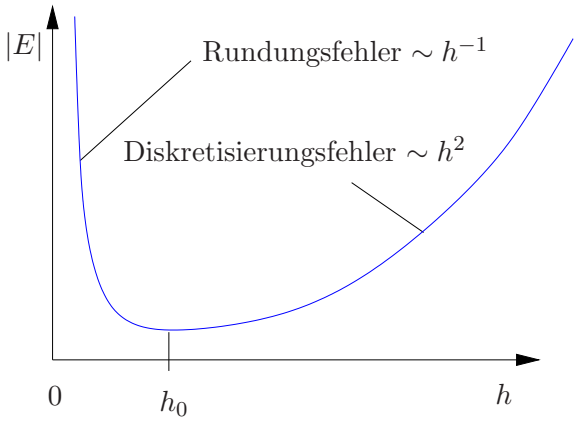
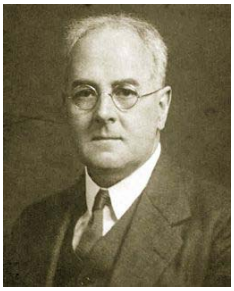


Abbildung 4.2.: Qualitatives Verhalten des gesamten Fehlerbetrags $|E|$ (Diskretisierungsfehler und Rundungsfehler) als Funktion der Intervallänge h. Bei der summierten Trapezregel hat man einen Fehler $\sim h^2$, der für $h \to 0$ in ein Verhalten $\sim h^{-1}$ übergeht.

4.4. Romberg-Verfahren

Mit Hilfe des *Romberg-Verfahrens* lassen sich Integrale genauer berechnen als mit den Newton-Cotes-Formeln. Die Idee besteht darin, das Intervall sukzessive äquidistant zu verfeinern und dabei nicht nur das Ergebnis auf dem feinsten Gitter zu verwenden sondern auch noch zusätzliche Informationen aus den Ergebnissen, die man auf den gröberen Gittern erhalten hat. Die Nutzung dieser Zusatzinformation beruht auf der *Richardson-Extrapolation*.



Lewis Fry Richardson 1881–1953

Wir nehmen an, daß wir das gesuchte Integral auf einem groben Gitter der Weite H und auf einem feinen Gitter der Weite h berechnet haben. Weiter sei der Fehler der verwendeten Approximation in beiden Fällen von m-ter Ordnung. Dann gilt

$$\text{grob:} \quad I(f) = I_H + c_H H^m, \tag{4.224a}$$

$$\text{fein:} \quad I(f) = I_h + c_h h^m. \tag{4.224b}$$

Diese Gleichungen enthalten die drei Unbekannten I, c_H und c_h. Die beiden Koeffizienten c_H und c_h sind verschieden, aber fast identisch. Wenn wir nun näherungsweise $c := c_h \approx c_H$ annehmen, dann können wir (4.224) lösen und c sowie $I(f)$ berechnen. Durch Bilden der Differenz erhalten wir

$$0 \approx I_H - I_h + c\,(H^m - h^m)\,, \tag{4.225}$$

woraus

$$c \approx \frac{I_h - I_H}{H^m - h^m} \tag{4.226}$$

folgt. Eingesetzt erhalten wir so

$$I(f) \approx I_h + \frac{I_h - I_H}{H^m - h^m} h^m = I_h + \frac{I_h - I_H}{(H/h)^m - 1}. \tag{4.227}$$

Dies nennt man *Richardson-Extrapolation*. Die Relation gilt natürlich nicht exakt, weil nur näherungsweise gilt $c_H \approx c_h$. Aber man kann erwarten, daß (4.227) ein besseres Ergebnis liefert als I_h allein.

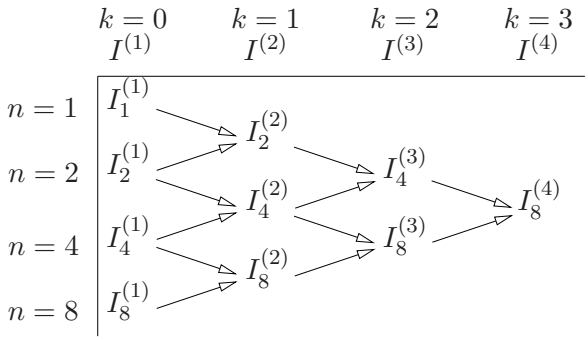


Abbildung 4.3.: Schematische Darstellung des Romberg-Verfahrens.

Als Beispiel betrachten wir nun die Trapezregel und berechnen die Richardson-Extrapolation basierend auf den beiden Intervallteilungen

$$h = \frac{b-a}{n} \qquad \text{und} \qquad H = 2h = \frac{b-a}{n/2}. \tag{4.228}$$

Der Fehler der summierten Trapezregel ist nach (4.219c) von zweiter Ordnung $m = 2$. Aus (4.227) folgt dann

$$I(f) \approx I_h + \frac{I_h - I_H}{(H/h)^2 - 1} = I_n + \frac{I_n - I_{n/2}}{2^2 - 1}. \tag{4.229}$$

Im letzten Schritt haben wir die Anzahl der Intervalle (n bzw. $n/2$), welche zur Berechnung der Integrale verwendet werden, als Index verwendet. Da das Ergebnis nur eine verbesserte Approximation ist, schreiben wir die Richardson-Extrapolation als

$$I_n^{(2)} := I_n^{(1)} + \frac{I_n^{(1)} - I_{n/2}^{(1)}}{2^2 - 1}. \tag{4.230}$$

Der hochgestellte Index in Klammern kennzeichnet die Ausgangsnäherung (1) und die erste Richardson-Extrapolation (2).

Man kann diese erste Richardson-Extrapolation nun für verschiedene Auflösungen (Anzahl von Intervallen bei der Aufteilung des Integrationsgebiets $[a, b]$) berechnen, die sich jeweils um einen Faktor 2 unterscheiden. Die benachbarten Paare der Anzahl von Intervallen sind $(1, 2)$, $(2, 4)$, $(4, 8)$, und so weiter. Dies ist in Abb. 4.3 skizziert. Aus den beiden Integralen eines jeden Paares erhält man je eine Richardson-Extrapolation. Diese verbesserten Näherungen bezeichnen wir mit $I_n^{(2)}$. Jetzt kann man einen Schritt weiter gehen und aus den ersten Extrapolationen $I_n^{(2)}$ wiederum in der gleichen Art den Fehler eliminieren. Wenn man dies systematisiert, erhält man das *Romberg-Verfahren*.

Werner Romberg
1909–2003

Um das Romberg-Verfahren durchzuführen, benötigen wir aber zunächst die Fehlerordnung der Näherungen $I^{(2)}$! Um sie zu erhalten, betrachten wir zunächst nur ein Intervall der Länge H mit den beiden Teilintervallen der Länge $h = H/2$ (Abb. 4.4) am Beispiel der Trapez-Regel. Nach (4.200) gilt dann

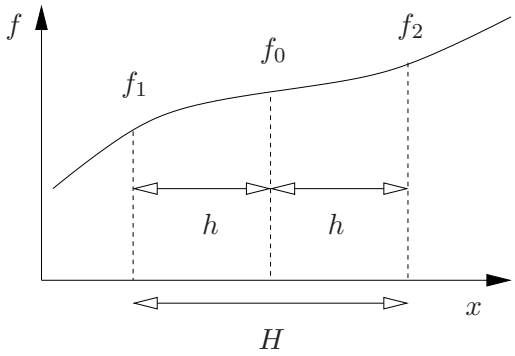


Abbildung 4.4.: Zur Berechnung des Fehlers der Richardson-Extrapolierten bei Verwendung der Trapezregel.

$$H: \qquad I_{n/2}^{(1)} = \frac{H}{2}(f_1 + f_2), \tag{4.231a}$$

$$h = \frac{H}{2}: \qquad I_n^{(1)} = \frac{h}{2}(f_1 + f_0) + \frac{h}{2}(f_0 + f_2) = \frac{H}{4}(f_1 + 2f_0 + f_2). \tag{4.231b}$$

Eingesetzt in (4.230) erhalten wir die Richardson-Extrapolation

$$\begin{aligned} I_n^{(2)} &= \frac{H}{4}(f_1 + 2f_0 + f_2) + \frac{\frac{H}{4}(f_1 + 2f_0 + f_2) - \frac{H}{2}(f_1 + f_2)}{3} \\ &= \frac{H}{4}\left[f_1 + 2f_0 + f_2 + \frac{(f_1 + 2f_0 + f_2) - 2(f_1 + f_2)}{3}\right] \\ &= \frac{H}{4}\left(\frac{2f_1}{3} + \frac{8f_0}{3} + \frac{2f_2}{3}\right) = \frac{H}{6}\left(f_1 + 4f_0 + f_2\right). \end{aligned} \tag{4.232}$$

Dies ist gerade die Simpson-Formel (4.201) und deren Fehler ist, wie wir in (4.217) gesehen hatten, von fünfter Ordnung. Wir benötigen hier aber die summierte Formel und diese ist nach (4.219c) von vierter Ordnung. Damit gilt für den Fehler

$$E_n^{(2)} = Mh^4 \sim \left(\frac{1}{n}\right)^4. \tag{4.233}$$

In Analogie zu (4.230) würde dann die zweite Richardson-Extrapolation (nun mit $m = 4$) lauten

$$I_n^{(3)} = I_n^{(2)} + \frac{I_n^{(2)} - I_{n/2}^{(2)}}{2^4 - 1}. \tag{4.234}$$

Dieses Verfahren führt man nun bis zu einer maximalen Anzahl von Intervallteilungen fort. Dabei nutzt man aus, daß für den Fehler in der $(k-1)$-ten Richardson-Extrapolation gilt (ohne Beweis)

$$E_n^{(k)} \sim \left(\frac{1}{n}\right)^{2k}. \tag{4.235}$$

Die Fehlerordnung (der Exponent) erhöht sich also bei jedem Schritt um 2, was zu einer rapiden Konvergenz führt. In Verallgemeinerung des Resultats lautet dann

die k-te Richardson-Extrapolation

$$I_n^{(k+1)} = I_n^{(k)} + \frac{I_n^{(k)} - I_{n/2}^{(k)}}{2^{2k} - 1}. \tag{4.236}$$

In Abb. 4.3 ist dies bis zur dritten Ordnung ($k = 3$) dargestellt.

Als Beispiel betrachten wir die Romberg-Integration von x^4 auf $[0, 1]$. Die exakte Lösung lautet

$$I(f) = \int_0^1 x^4 \, \mathrm{d}x = \left[\frac{x^5}{5}\right]_0^1 = 0.2. \tag{4.237}$$

Wir werden die Integration bis $n = 4$ bzw. $k = 2$ durchführen. Mit Hilfe der Trapezregel (4.200) verschaffen wir uns zunächst die Ausgangsintegrale

$$I_1^{(1)} = \frac{1}{2}\left[f(0) + f(1)\right] = 0.5, \tag{4.238a}$$

$$I_2^{(1)} = \frac{1}{4}\left[f(0) + 2f(0.5) + f(1)\right] = \frac{1}{4}\left[0 + 2 \times 0.5^4 + 1\right] = 0.28125, \tag{4.238b}$$

$$\begin{aligned} I_4^{(1)} &= \frac{1}{8}\left[f(0) + 2f(0.25) + 2f(0.5) + 2f(0.75) + f(1)\right] \\ &= \frac{1}{8}\left[0 + 2 \times 0.25^4 + 2 \times 0.5^4 + 2 \times 0.75^4 + 1\right] = 0.220703125 \end{aligned} \tag{4.238c}$$

Die erste Richardson-Extrapolation liefert dann

$$I_2^{(2)} = 0.28125 + \frac{0.28125 - 0.5}{3} = 0.208333333, \tag{4.239a}$$

$$I_4^{(2)} = 0.220703125 + \frac{0.220703125 - 0.28125}{3} = 0.200520833. \tag{4.239b}$$

Die zweite Richardson-Extrapolation ergibt

$$I_4^{(3)} = 0.200520833 + \frac{0.200520833 - 0.208333333}{2^4 - 1} = 0.200000000. \tag{4.240}$$

Dieses Ergebnis stimmt in mindestens 9 Dezimalstellen mit dem exakten Ergebnis überein.

4.5. Gauß-Quadratur

Bei den Quadraturen nach Newton-Cotes sind wir von fest vorgegebene Stützstellen x_i ausgegangen. Das Problem bestand darin, die $n + 1$ Gewichtsfaktoren a_i in (4.197) geeignet zu bestimmen. Sie wurden so festgelegt, daß Polynome bis zum Höchstgrade n *exakt* integriert werden. Dies führte auf ein lineares System zur Bestimmung der Koeffizienten a_i.

Hier betrachten wir wieder eine Approximation des Integrals in der Form[5]

$$I_{\mathrm{G}}(f) = \sum_{i=1}^{n} a_i f\left(x_i\right). \tag{4.241}$$

Johann Carl Friedrich Gauß 1777–1855

Die Idee der *Gauß-Quadratur* ist es, neben den n Gewichten a_i auch die n Stützstellen x_i als Unbekannte aufzufassen und diese so zu bestimmen, daß das resultierende Ergebnis möglichst genau ist. Damit hat man doppelt so viele freie Parameter $(2n)$ wie beim Newton-Cotes-Ansatz. Ähnlich wie in (4.202) kann man nun x_i und a_i so wählen, daß Polynome bis zum Grade $2n-1$ exakt integriert werden. Daher wird eine wesentlich höhere Genauigkeit erwartet. Allgemein bezeichnet man alle Integrationsformeln, bei denen sowohl alle Gewichte wie auch alle Stützstellen variiert werden, als *Gauß-Quadratur*.

Um die Gauß-Methode zu nutzen, ist es sinnvoll, das Integrationsintervall $x \in [a,b]$ mit Hilfe von

$$x = \frac{b+a}{2} + \frac{b-a}{2}\xi \tag{4.242}$$

auf das Intervall $\xi \in [-1,1]$ abzubilden. Dies ist keine Einschränkung, denn mit $\mathrm{d}x = (b-a)\mathrm{d}\xi/2$ gilt

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left[x(\xi)\right]\,\mathrm{d}\xi. \tag{4.243}$$

4.5.1. 2-Punkt-Gauß-Quadratur

Bevor wir uns dem allgemeinen Fall zuwenden, betrachten wir $n = 2$. Die 2-Punkt-Näherung (4.241) des Integrals lautet

$$I_2(f) = a_1 f(x_1) + a_2 f(x_2). \tag{4.244}$$

Mit dieser Formel sollten wir Polynome bis zur Ordnung $(2n-1)$ exakt integrieren können. Für $n = 2$ hat das allgemeine Polynom die Form

$$f(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3, \tag{4.245}$$

mit beliebigen Koeffizienten c_i. Das exakte Integral über $[-1,1]$ lautet einerseits

$$I(f) = \int_{-1}^{1} f(x)\,\mathrm{d}x = \left[c_1 x + \frac{c_3}{3}x^3\right]_{-1}^{1} = 2c_1 + \frac{2}{3}c_3. \tag{4.246}$$

[5] Die Summe beginnt hier bei 1 und nicht bei 0 wie in (4.197).

Wenn wir andererseits f mittels der Quadraturformel (4.244) integrieren, erhalten wir formal

$$\underbrace{c_1 I_2(1) + c_2 I_2(x) + c_3 I_2(x^2) + c_4 I_2(x^3)}_{\text{Approximation}} \stackrel{!}{=} \underbrace{2c_1 + \frac{2}{3}c_3}_{\text{exakt}}. \tag{4.247}$$

Hierbei haben wir gefordert $I_2(f) = I(f)$! Die Quadraturformeln für die einzelnen Potenzfunktionen lauten

$$I_2(1) = a_1 + a_2, \tag{4.248a}$$
$$I_2(x) = a_1 x_1 + a_2 x_2, \tag{4.248b}$$
$$I_2(x^2) = a_1 x_1^2 + a_2 x_2^2, \tag{4.248c}$$
$$I_2(x^3) = a_1 x_1^3 + a_2 x_2^3. \tag{4.248d}$$

Ein Koeffizientenvergleich in (4.247) (die Koeffizienten c_i sind beliebig) ergibt damit das *nichtlineare* Gleichungssystem für die Unbekannten a_1, a_2, x_1 und x_2

$$a_1 + a_2 = 2, \tag{4.249a}$$
$$a_1 x_1 + a_2 x_2 = 0, \tag{4.249b}$$
$$a_1 x_1^2 + a_2 x_2^2 = \frac{2}{3}, \tag{4.249c}$$
$$a_1 x_1^3 + a_2 x_2^3 = 0. \tag{4.249d}$$

Das System kann man eindeutig lösen. Man erhält[6]

$$a_1 = 1 \qquad a_2 = 1, \tag{4.250a}$$
$$x_1 = -\frac{1}{\sqrt{3}} \qquad x_2 = \frac{1}{\sqrt{3}}. \tag{4.250b}$$

Damit lautet die 2-Punkt-Gauß-Quadratur

$$I_2(f) = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right). \tag{4.251}$$

4.5.2. n-Punkt-Gauß-Quadratur

Für $n \geq 2$ kann man das resultierende *nichtlineare* Gleichungssystem nicht mehr so einfach lösen. Dann benötigt man einen anderen Zugang. Man kann nun zeigen, daß die n-Punkt-Gauß-Quadratur (4.241) ein beliebiges Polynom $(2n-1)$-ter Ordnung exakt integriert, wenn die n Stützstellen x_i die Nullstellen eines Polynoms n-ter Ordnung

$$P_n(x) = (x - x_1)(x - x_2)\ldots(x - x_n) \tag{4.252}$$

[6] Aus (4.249b) und (4.249d) folgt $a_1/a_2 = -x_2/x_1$ und $a_1/a_2 = -x_2^3/x_1^3$ was auf $x_1^2 = x_2^2$ und damit auf $a_1/a_2 = \pm 1$ führt. Wegen (4.249a) muß aber das Pluszeichen gelten, so daß $a_1 = a_2 = 1$ und $x_1 = -x_2$. Aus (4.249c) folgt damit $x_{1,2} = \pm 1/\sqrt{3}$.

sind, welches orthogonal ist zu jedem beliebigen Polynom $Q(x)$ niedrigerer Ordnung, also von höchstens $(n-1)$-ter Ordnung.

Orthogonalität bedeutet in diesem Zusammenhang, daß

$$\langle P_n|Q\rangle := \int_{-1}^{1} P_n(x)Q(x)\,\mathrm{d}x = 0. \tag{4.253}$$

Wenn wir jetzt Polynome der Struktur $f(x) = P_n(x)Q(x)$ betrachten, mit Ordnung zwischen n und $2n-1$, so gilt offenbar

$$0 \overset{\substack{(4.253)\\ \text{exakt}}}{=} \int_{-1}^{1} \underbrace{P_n(x)Q(x)}_{f(x)}\,\mathrm{d}x \overset{\substack{(4.241)\\ \text{Gauss-Quadr.}}}{=} \sum_{i=1}^{n} a_i \underbrace{P_n(x_i)Q(x_i)}_{f(x_i)} \overset{(4.252)}{=} 0, \tag{4.254}$$

da $P_n(x_i) = 0$ für $i = 1,\ldots,n$. Im Anhang B wird gezeigt, daß *alle* Polynome der Ordnung $m \le 2n-1$ exakt integriert werden.

Von zentraler Bedeutung für die Gauß-Quadratur sind damit *orthogonale Polynome*. Die Stützstellen der Gauß-Quadratur der Ordnung n sind gerade die Nullstellen eines orthogonalen Polynoms der Ordnung n. Wenn die Stützstellen bekannt sind, kann man die Gewichtsfunktionen a_i aus der Lösung eines (dann linearen) Gleichungssystems wie (4.249) erhalten.

Die Gewichte a_i kann man bei gegebenen Stützstellen x_i aber auch wie in (4.206) durch die Integration des Lagrangeschen Interpolationspolynoms (4.204) vom Grade n erhalten (mit Summe über $i = 1,\ldots,n$), denn

$$I(f) \approx \int_{-1}^{1} \left(\sum_{i=1}^{n} L_i^{(n)}(x) f(x_i)\right) \mathrm{d}x = \sum_{i=1}^{n} f(x_i) \underbrace{\int_{-1}^{1} L_i^{(n)}(x)\,\mathrm{d}x}_{a_i} \overset{!}{=} \sum_{i=1}^{n} a_i f(x_i) = I_n(f). \tag{4.255}$$

Die Gewichte a_i ergeben sich demnach als Integrale über die zu den Stützstellen x_i gehörigen Lagrange-Polynome $L_i^{(n)}(x)$

$$a_i = \int_{-1}^{1} L_i^{(n)}(x)\,\mathrm{d}x. \tag{4.256}$$

4.5.3. Orthogonale Polynome

Orthogonale Polynome $\{f_n(x)\}_{n\in\mathbb{N}_0}$ sind charakterisiert durch die *Orthogonalitätsrelation*

$$\langle f_n|f_m\rangle = \int_a^b w(x) f_n(x) f_m(x)\,\mathrm{d}x = h_n \delta_{n,m}. \tag{4.257}$$

Hierbei nennt man $\langle\ldots\rangle := \int_a^b w(x)\ldots\mathrm{d}x$ auch *Skalarprodukt*.[7] Es gibt verschiedene Systeme von orthogonalen Polynomen. Das jeweilige System wird durch die

[7] Durch Vergleich von (4.257) mit $\vec{e}_n \cdot \vec{e}_m = \delta_{n,m}$ erkennt man die formale Analogie zwischen der *Orthogonalität von Funktionen* $\{f_n(x)\}$ und der *Orthogonalität von Einheitsvektoren* $\{\vec{e}_n\}$ im $\mathbb{R}^N$.

Gewichtsfunktion $w(x)$ bis auf einen Normierungsfaktor eindeutig festgelegt. Die Polynome sind ganz allgemein von der Form

$$f_n(x) = k_n x^n + k'_n x^{n-1} + \dots \tag{4.258}$$

mit gewissen Koeffizienten k_n und k'_n. Außerdem genügen alle orthogonalen Polynome f_n einer *Rekursion* der Form

$$f_{n+1} = (a_n + x b_n) f_n - c_n f_{n-1}. \tag{4.259}$$

Die Koeffizienten a_n, b_n und c_n der Rekursionsformel stehen mit den Koeffizienten k_n und k'_n und den Normierungsfaktoren h_n im Zusammenhang

$$b_n = \frac{k_{n+1}}{k_n}, \quad a_n = b_n \left(\frac{k'_{n+1}}{k_{n+1}} - \frac{k'_n}{k_n} \right), \quad c_n = \frac{k_{n+1} k_{n-1} h_n}{k_n^2 h_{n-1}}. \tag{4.260}$$

Legendre-Polynome

Adrien-Marie Legendre 1752–1833

Im einfachsten Fall ist $[a, b] = [-1, 1]$ und $w(x) = 1$. Die resultierenden Polynome heißen *Legendre-Polynome* und werden mit $P_n(x)$ bezeichnet. Mit den ersten Polynomen $P_0(x) = 1$ und $P_1(x) = x$ kann man aus der Rekursionsformel

$$(n+1) P_{n+1}(x) = (2n+1) x P_n(x) - n P_{n-1}(x), \qquad n = 1, 2, \dots \tag{4.261}$$

alle Legendre-Polynome sukzessive konstruieren. Sie sind entsprechend (4.257) orthogonal. Der Normierungsfaktor ist $h_n = 2/(2n+1)$. Die Legendre-Polynome niedriger Ordnung lauten[8]

$$P_0(x) = 1, \tag{4.262a}$$
$$P_1(x) = x, \tag{4.262b}$$
$$P_2(x) = \frac{1}{2}(3x^2 - 1), \tag{4.262c}$$
$$P_3(x) = \frac{1}{2}(5x^3 - 3x), \tag{4.262d}$$
$$P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3), \tag{4.262e}$$
$$P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x). \tag{4.262f}$$

Sie sind in Abb. 4.5 dargestellt. Man sieht, daß die oben in (4.250) berechneten Stützstellen der 2-Punkt-Gauß-Quadratur gerade die Nullstellen von $P_2(x)$ sind.

[8] Numerisch können die Legendre-Polynome mit Hilfe des Gram-Schmidt'schen Orthogonalisierungsverfahren (s.u.) ausgehend von den Monomen $(x^n)_{n \in \mathbb{N}}$ iterativ erzeugt werden.

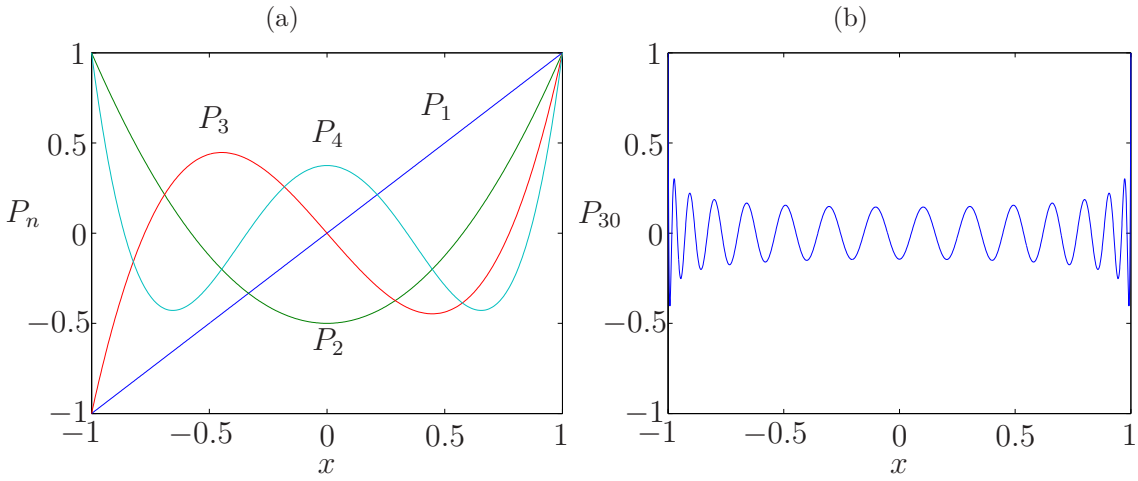


Abbildung 4.5.: Legendre-Polynome $P_n(x)$. Im Limes hoher Ordnung n fallen die Funktionen sehr schnell vom Randwert ab und besitzen im Innern eine fast konstante Oszillationsamplitude. Die Oszillationen sind in Randnähe schneller als im Zentrum.

Die Legendre-Polynome[9] bilden ein *vollständiges Orthogonalsystem*. Das heißt, man kann jede reelle Funktion $f(x)$ auf $[-1, 1]$ nach Legendre-Polynomen entwickeln

$$f(x) = \sum_{n=0}^{\infty} c_n \, P_n(x). \tag{4.263}$$

Da die Legendre Polynome orthogonal sind, ergeben sich die Koeffizienten c_m durch Projektion von $f(x)$ mittels des Skalarprodukts (4.257) auf die Legendre-Polynome P_m

$$\langle P_m | f \rangle = \int_{-1}^{1} P_m(x) f(x) \, \mathrm{d}x = \int_{-1}^{1} \sum_{n=0}^{\infty} c_n P_m(x) P_n(x) \, \mathrm{d}x$$

[9] Die Legendre-Polynome $P_n(x)$ sind Lösungen der Legendreschen Differentialgleichung

$$(1 - x^2) f'' - 2x f' + n(n+1) f = 0, \qquad n \in \mathbb{N}_0.$$

Diese Gleichung ist ein Grenzfall ($m = 0$) der allgemeinen Legendre-Differentialgleichung

$$(1 - x^2) f'' - 2x f' + \left(\ell[\ell + 1] - \frac{m^2}{1 - x^2} \right) f = 0,$$

mit ℓ und m ganzzahlig und $0 \le m \le \ell$. Ihre Lösungen sind die sogenannten *zugeordneten Legendre-Polynome* $f = P_l^{(m)}(x)$. Die allgemeine Legendresche Differentialgleichung resultiert, wenn man Lösungen der Laplace-Gleichung in Kugelkoordinaten sucht. Die Differentialgleichung beschreibt dann den meridionalen Anteil ($x = \cos\theta$) der Lösung. Der azimutale Anteil ist $\sim \mathrm{e}^{\mathrm{i}m\varphi}$. Dies spielt z.B. eine Rolle in der quantenmechanischen Berechnung des Wasserstoffatoms oder bei anderen Problemen mit Kugelsymmetrie (Elekrostatik, Geodäsie). Für den Fall $m = 0$ sind die zugeordneten Legendre-Polynome identisch mit den Legendre-Polynomen $P_l^{(m=0)}(x) = P_l(x)$.

$$= \sum_{n=0}^{\infty} c_n \underbrace{\int_{-1}^{1} P_m(x) P_n(x)\,\mathrm{d}x}_{2\delta_{n,m}/(2m+1)} = \frac{2c_m}{2m+1}. \tag{4.264}$$

Chebyshev-Polynome

Pafnuty Lvovich Chebyshev 1821–1894

Ein anderes populäres System orthogonaler Polynome erhält man für $[a,b] = [-1,1]$ und $w(x) = (1-x^2)^{-1/2}$. Dann ergeben sich die *Chebyshev-Polynome* $T_n(x)$. Das Skalarprodukt lautet dann

$$\begin{aligned} \langle T_n | T_m \rangle &= \int_{-1}^{1} \frac{T_n(x) T_m(x)}{\sqrt{1-x^2}} \mathrm{d}x \\ &= h_n \delta_{n,m} = \delta_{n,m} \begin{cases} \pi, & n = 0, \\ \pi/2, & n \neq 0. \end{cases} \end{aligned} \tag{4.265}$$

Mit $T_0 = 1$ und $T_1 = x$ erhält man die übrigen Chebyshev-Polynome mit der einfachen Rekursion

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \qquad n = 1, 2, \ldots \tag{4.266}$$

Die ersten Polynome lauten[10]

$$T_0(x) = 1, \tag{4.267a}$$
$$T_1(x) = x, \tag{4.267b}$$
$$T_2(x) = 2x^2 - 1, \tag{4.267c}$$
$$T_3(x) = 4x^3 - 3x, \tag{4.267d}$$
$$T_4(x) = 8x^4 - 8x^2 + 1, \tag{4.267e}$$
$$T_5(x) = 16x^5 - 20x^3 + 5x. \tag{4.267f}$$

Sie sind in Abb. 4.6 gezeigt. Mit Hilfe von $x = \cos\theta$ kann man die Chebyshev-Polynome auch schreiben als[11]

$$T_n(x) = \cos(n\theta) = \cos\left[n \arccos(x)\right]. \tag{4.268}$$

[10] Die Chebyshev-Polynome sind Lösung der Chebyshev-Differentialgleichung

$$(1-x^2)f'' - xf' + n^2 f = 0.$$

[11] Dies kann man mittels der Rekursionsformel prüfen. Eingesetzt in (4.266) ergibt sich

$$T_{n+1}(x) = \cos[(n+1)\theta] = \underbrace{2\overbrace{\cos\theta}^{x}\cos(n\theta)}_{\cos[(n-1)\theta]+\cos[(n+1)\theta]} - \cos[(n-1)\theta] = 2xT_n(x) - T_{n-1}(x). \quad \checkmark$$

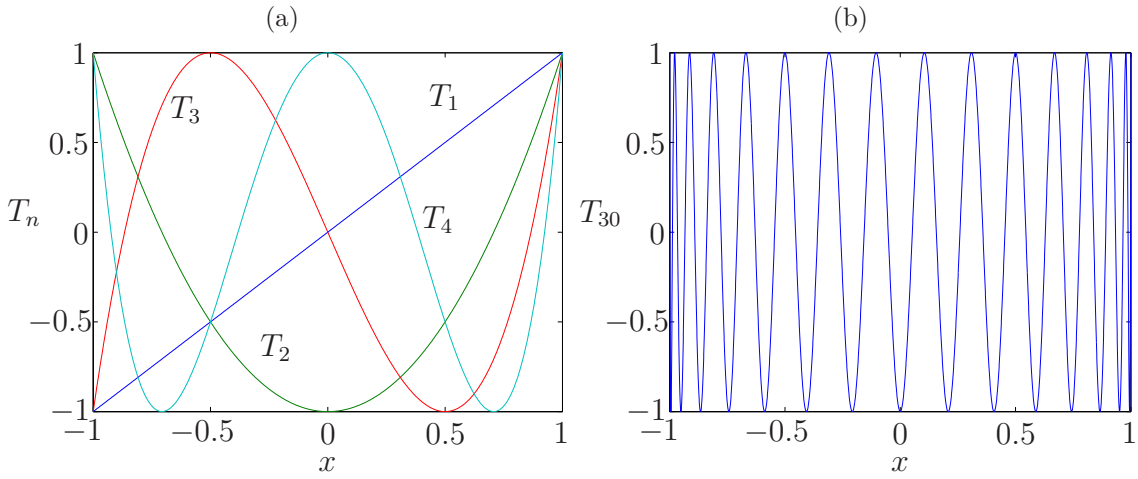


Abbildung 4.6.: Chebyshev-Polynome $T_n(x)$. Sie oszillieren am Rand sehr viel schneller als im Innern. Alle lokalen Extrema sind vom Betrage eins.

Auch die Chebyshev-Polynome bilden ein vollständiges Orthogonalsystem. Man kann jede reelle Funktion $f(x)$ auf $[-1, 1]$ darstellen als

$$f(x) = \sum_{i=0}^{\infty} c_i T_i(x). \tag{4.269}$$

Wenn man die Reihe bei $i = n$ abbricht, kann man die $n+1$ Koeffizienten c_i erhalten, indem man $f(x)$ mittels des Skalarprodukts (4.257) auf die Chebyshev Polynome projiziert, wie in (4.264). Dieses Verfahren wird *Galerkin-Methode* genannt.

Rehuel Lobatto
1797–1866

Alternativ dazu kann man die Funktion $f(x)$ an den Extremstellen x_j von $T_n(x)$ auswerten. Dann ergeben sich die Koeffizienten c_i aus den $n + 1$ linearen Gleichungen

$$f(x_j) = \sum_{i=0}^{n} c_i T_i(x_j). \tag{4.270}$$

Dies wird *Kollokations-Methode* genannt. Die Extremstellen von $T_n(x)$ inklusive der Randpunkte heißen *Chebyshev-Gauß-Lobatto-Punkte*. Für die Extremstellen gilt $T_n'(x_j) = 0$, d.h.

$$T_n'(x_j) = [\cos(n\theta_j)]' = -n \sin(n\theta_j)\, \theta_j' = 0. \tag{4.271}$$

Dies gilt bei $n\theta_j = j\pi$,[12] oder

$$x_j = \cos\left(\frac{j\pi}{n}\right), \qquad j = 0, 1, \ldots, n. \tag{4.272}$$

Diese Chebyshev-Gauß-Lobatto-Punkte hatten wir schon oben in (3.107) zur Vermeidung des Runge-Phänomens verwendet.[13] Für weitere Systeme von orthogonalen Polynomen sei auf Abramowitz and Stegun (1972) verwiesen.

[12] Für die beiden Randpunkte $x = \pm 1$ ist $T_n'(x_j) \neq 0$, da $\theta_j' \to -\infty$ divergiert.
[13] Bis auf die Reihenfolge der Numerierung der Punkte.

4.5.4. Bezug zur Gauß-Quadratur

Für die Gauß-Quadratur werden die Nullstellen z.B. der Legendre-Polynome benötigt. Die Nullstellen für die N-Punkt-*Legendre-Gauß-Quadratur* kann man aus der Rekursion (4.261) gewinnen. Sei $x_i \in [-1, 1]$ irgend ein Punkt. Dann gilt nach (4.261)

$$(n+1)P_{n+1}(x_i) = (2n+1)x_i P_n(x_i) - nP_{n-1}(x_i), \qquad n = 1, 2, \ldots, \tag{4.273}$$

oder

$$\frac{n+1}{2n+1}P_{n+1}(x_i) + \frac{n}{2n+1}P_{n-1}(x_i) = x_i P_n(x_i), \qquad n = 1, 2, \ldots \tag{4.274}$$

Dies kann man verallgemeinern. Die allgemeine Rekursion für orthogonale Polynome (4.259) kann man in der Form schreiben

$$a_n p_{n+1}(x_i) + b_n p_n(x_i) + c_n p_{n-1}(x_i) = x_i p_n(x_i), \qquad n = 1, 2, \ldots \tag{4.275}$$

Wenn wir nun den Vektor $\vec{p} = [p_0(x_i), p_1(x_i), ..., p_{N-1}(x_i)]^{\mathrm{T}}$ der Länge N definieren, können wir die Gleichungen für $n = 0, 1, 2, \ldots, N-1$ in Matrixform schreiben

$$\begin{pmatrix} b_0 & a_0 & & & & \\ c_1 & b_1 & a_1 & & & \\ & c_2 & b_2 & a_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{N-2} & b_{N-2} & a_{N-2} \\ & & & & c_{N-1} & b_{N-1} \end{pmatrix} \cdot \vec{p} = x_i \vec{p}. \tag{4.276}$$

In der ersten Zeile taucht c_0 nicht auf, da p_{-1} nicht existiert. In der letzten Zeile müßte eigentlich noch a_{N-1} auftauchen. Dies ist aber nicht der Fall, genau dann wenn x_i gerade eine Nullstelle von $p_N(x)$ ist. Denn der Vorfaktor a_{N-1} vor $p_N(x_i) = 0$ wird dann mit Null multipliziert und der Term verschwindet aus den Gleichungen. Die Rekursion bricht dann ab!

Gleichung (4.276) definiert über die tridiagonale Matrix ein *Eigenwertproblem*. Die N Eigenwerte (reell, nicht entartet) der Matrix sind gerade die N Nullstellen x_i von $p_N(x)$, d.h. die gesuchten Stützstellen der Gauß-Quadratur. Die Komponenten der Eigenvektoren repräsentieren die Werte der anderen orthogonalen Polynome $p_n(x_i)$ mit $n < N$ an den Stützstellen. Wie man Eigenwert-Probleme wie (4.276) lösen kann, wird in Kap. 6 behandelt.

Hat man die Nullstellen x_i des Legendre-Polynoms $P_N(x)$ auf diese Weise gefunden, ergeben sich die Gewichtskoeffizienten für die Legendre-Gauß-Quadratur als (ohne Beweis, siehe Abramowitz and Stegun, 1972)

$$a_i = \frac{2}{(1 - x_i^2)\left[P_N'(x_i)\right]^2}. \tag{4.277}$$

Für niedrige Werte von N sind diese Größen in Tab. 4.1 aufgelistet.

Anstelle der Nullstellen der Legendre-Polynome kann man als Stützstellen für die Gauß-Quadratur (4.241) auch Nullstellen der Chebyshev-Polynome oder anderer orthogonaler Polynome verwenden. Dazu bringt man zunächst das zu berechnende Integral in eine Form, in der auch die Gewichtsfunktion $w(x)$ für das Skalarprodukt der betreffenden Polynome auftritt. Für Chebyshev-Polynome mit $w(x) = 1/\sqrt{1-x^2}$ wäre also

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \int_{-1}^{1} w(x)\,\underbrace{\sqrt{1-x^2}f(x)}_{g(x)}\,\mathrm{d}x. \tag{4.278}$$

Als Stützstellen fungieren dann die Nullstellen des n-ten Chebyshev-Polynoms. Für Chebychev-Polynome kann man die Nullstellen sehr einfach mit Hilfe der Beziehung $T_n(x_i) = \cos(n\theta_i) = 0$ berechnen. Hieraus folgt $n\theta_i = (2i-1)(\pi/2)$ mit $i = 1, \ldots, n$. Wegen $x = \cos\theta$ folgt

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \qquad i = 1, \ldots, n. \tag{4.279}$$

Diese Punkte heißen *Chebyshev-Knoten*. Das resultierende Verfahren wird *Chebyshev-Gauß-Quadratur* genannt. Mit diesen Stützpunkten gilt dann

$$\int_{-1}^{1} w(x)g(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} a_i g(x_i) = \sum_{i=1}^{n} a_i f(x_i)\sqrt{1-x_i^2}. \tag{4.280}$$

Tabelle 4.1.: Stützstellen x_i und Gewichte a_i für die Gauß-Legendre-Quadratur.

Punktezahl n	Stützstellen x_i	Gewichte a_i
1	0	2
2	$\pm\sqrt{1/3}$	1
3	0	$8/9$
	$\pm\sqrt{3/5}$	$5/9$
4	$\pm\sqrt{\left(3-2\sqrt{6/5}\right)/7}$	$\dfrac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\left(3+2\sqrt{6/5}\right)/7}$	$\dfrac{18-\sqrt{30}}{36}$
5	0	$128/225$
	$\pm\dfrac{1}{3}\sqrt{5-2\sqrt{10/7}}$	$\dfrac{322+13\sqrt{70}}{900}$
	$\pm\dfrac{1}{3}\sqrt{5+2\sqrt{10/7}}$	$\dfrac{322-13\sqrt{70}}{900}$

Die Gewichtskoeffizienten $a_i = \pi/n$ sind konstant. Damit kann man explizit schreiben

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \frac{\pi}{n}\sum_{i=1}^{n} f\left[\cos(...)\right]\sqrt{1-\cos^2(...)} = \frac{\pi}{n}\sum_{i=1}^{n} f\left[\cos(...)\right]\sin(...)$$
$$= \frac{\pi}{n}\sum_{i=1}^{n} f\left[\cos\left(\frac{2i-1}{2n}\pi\right)\right]\sin\left(\frac{2i-1}{2n}\pi\right). \tag{4.281}$$

Bei den genannten Verfahren (Legendre-Gauß und Chebyshev-Gauß) werden nur Stützpunkte im Innern des Intervalls für die Quadratur verwendet. Manchmal ist es aber wichtig, auch die Randpunkte selbst zu verwenden. Hierfür existieren gewissen Modifikationen (siehe z.B. Canuto et al., 1988), die nur geringfügig ungenauer sind als die Verfahren ohne Verwendung der Randpunkte. Wird ein einziger Randpunkt verwendet, spricht man von *Gauß-Radau-Quadratur*, werden beide Randpunkte verwendet heißt das Verfahren *Gauß-Lobatto-Quadratur*. Diese Verfahren integrieren dann nur noch Polynome bis zur Ordnung $2N-2$ (Gauß-Radau) bzw. $2N-3$ (Gauß-Lobatto) exakt, anstelle von $2N-1$ für die Formeln ohne die Randpunkte.

In Matlab ist eine adaptive Gauß-Lobatto-Quadratur im Befehl `q = quadl(fun,a,b)` implementiert. Hierbei muß die Funktion `fun` definiert sein. Sie wird über das Intervall `[a,b]` integriert. Der Befehl `q = quad(fun,a,b)` verwendet eine adaptive Simpson-Regel.

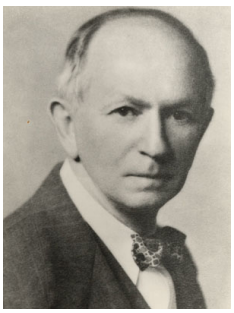
5. Anfangs- und Randwert-Probleme

Eine sehr wichtige Klasse von Problemen hat die mathematische Struktur

$$y_i'(x) = f_i(x, \vec{y}). \tag{5.282}$$

Hierbei ist x eine unabhängige Variable (z.B. der Ort oder die Zeit) und die Vektorfunktion f_i ein vorgegebener (bekannter) funktionaler Zusammenhang zwischen den Komponenten y_i des unbekannten Vektors $\vec{y}$. Man kennt also die funktionale Abhängigkeit der einzelnen Änderungsraten $f_i(x, \vec{y})$ und sucht die Lösung $y_i(x)$.

Vito Volterra
1860–1940



Alfred James
Lotka
1880–1949

Die Problemstellung (5.282) nennt man auch oft *dynamisches System*. Dynamische Systeme treten in der Technik sehr häufig auf. Ein einfaches Beispiel ist ein (mathematisches) *Pendel*, dessen Aufhängepunkt $z(t) = a\sin(\omega t)$ vertikal mit Frequenz ω und Amplitude a oszilliert (Abb. 5.1). Die Bewegungsgleichung für dieses *parametrisch angetriebene System* lautet

$$ml\ddot{\varphi} = -m\left[g - a\omega^2 \sin(\omega t)\right] \sin\varphi. \tag{5.283}$$

Mit der Definition $\psi := \dot{\varphi}$ kann diese Gleichung in die Form

$$\dot{\psi} = -\left[\frac{g}{l} - \frac{a\omega^2}{l}\sin(\omega t)\right] \sin\varphi, \tag{5.284a}$$

$$\dot{\varphi} = \psi, \tag{5.284b}$$

gebracht werden, woran man leicht die Struktur von (5.282) erkennt: $\dot{y}_1(t) = f_1(t, y_1, y_2)$ und $\dot{y}_2(t) = f_2(t, y_1, y_2)$.

Ähnliche dynamische Systeme treten auch in der Biologie bei der mathematischen Beschreibung der Entwicklung von Populationen auf. Ein bekanntes Beispiel für ein System von zwei gewöhnlichen Differentialgleichungen erster Ordnung ist das *Lotka-Volterra-Modell*

$$\dot{X}(t) = -b_X X(t) + c_X X(t) Y(t), \tag{5.285a}$$

$$\dot{Y}(t) = -b_Y Y(t) + c_Y X(t) Y(t). \tag{5.285b}$$

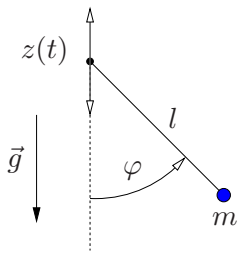


Abbildung 5.1.: Mathematisches Pendel, dessen Aufhängepunkt vertikal harmonisch mit $z(t) = a\sin(\omega t)$ oszilliert.

Es wird auch als *Räuber-Beute-Modell* bezeichnet, da die abhängigen Größen $X(t)$ und $Y(t)$ die Anzahl der Räuber- bzw. Beutetiere repräsentieren. Die Dynamik hängt von den Koeffizienten ab.[1]

In der Technik kann z.B. das Schwingungsverhalten von Fahrzeugen oder die Dynamik von Strömungen mittels dynamischer Modelle beschrieben werden. Sie stehen auch in direktem Zusammenhang mit der linearen und nichtlinearen Regelung von Systemen. Selbst bei der Prognose von Börsenkursen kommen dynamische Systeme zum Einsatz.[2]

5.1. Anfangswertprobleme für gewöhnliche Differentialgleichungen

Augustin Louis Cauchy
1789–1857

Wir betrachten zunächst den eindimensionalen Fall. Die Integration einer Funktion $f(x)$ über dem Intervall $[a, b]$ kann man auch in differentieller Form als das Problem

$$y'(x) = f(x), \qquad \text{mit} \quad y(a) = 0, \tag{5.286}$$

auffassen. Denn durch Integration erhalten wir die Lösung

$$y(x) = \int_a^x f(s)\,\mathrm{d}s. \tag{5.287}$$

Das Problem (5.286) kann man verallgemeinern zu

$$y'(x) = f\left[x, y(x)\right], \qquad \text{mit} \quad y(a) = \alpha \quad \text{und} \quad x > a. \tag{5.288}$$

Dies ist ein einfaches *Anfangswertproblem* für *gewöhnliche Differentialgleichungen*. Man nennt es auch *Cauchy-Problem*. Gewöhnliche Differentialgleichungen sind Differentialgleichungen, die nur von einer einzigen unabhängigen Variable (hier x) abhängen. Die Aufgabe besteht darin, aus dem Anfangswert $y(a) = \alpha$ den Verlauf von $y(x)$ für $x > a$ zu berechnen. Gleichung (5.288) ist von erster Ordnung in x, da die höchste vorkommende Ableitung $y^{(1)} = y'$ ist.

Eine weitere Verallgemeinerung besteht darin, ein System von n gewöhnlichen Differentialgleichungen erster Ordnung zu betrachten

$$y_i'(x) = f_i\left[x, y_1(x), \ldots, y_n(x)\right], \quad \text{für} \quad i = 1, \ldots, n \quad \text{und} \quad x \geq a, \tag{5.289}$$

welches den Anfangswerten $y_i(a) = \alpha_i$ genügt.[3]

[1] Es bedeuten b_X die Sterberate der Räuber, wenn keine Beute vorhanden ist, $-b_Y$ die Reproduktionsrate der Beute ohne Störung und bei großem Nahrungsangebot, c_X die Reproduktionsrate der Räuber pro Beutelebewesen und $-c_Y$ die Freßrate der Räuber pro Beutelebewesen, d.h. die Sterberate der Beute pro Räuber.

[2] Entscheidende Impulse erhielt die mathematische Behandlung nichtlinearer dynamischer Systeme durch Poincaré im Zusammenhang mit Problemen der Planetenbewegung.

[3] Wenn man eine gewöhnliche Differentialgleichung n-ter Ordnung zu lösen hat, in welcher als

Rudolf Otto Sigismund Lipschitz 1832–1903

Für die Lösung des eindimensionalen Anfangswertproblems (5.288) und dessen numerische Approximation ist es wichtig zu klären, ob die Lösung eindeutig ist. Diese Eigenschaft stellt die Lipschitz-Bedingung sicher, welche eine Forderung an die Funktion $f(x,y)$ ist. Eine Funktion $f(x,y)$ genügt einer *Lipschitz-Bedingung* auf dem Definitionsgebiet $D \in \mathbb{R} \times \mathbb{R}$, wenn gilt[4]

$$\|f(x,y_1) - f(x,y_2)\| \leq L(x)\,\|y_1 - y_2\|\,, \quad \forall\, x, y_1, y_2, \tag{5.290}$$

mit einer stetigen Funktion $L(x)$ (Lipschitz-Funktion). Anschaulich bedeutet dies, daß der Betrag der Steigung von f bezüglich y nach oben beschränkt ist.

Wenn die Funktion $f(x,y)$ aus (5.288) bzgl. x und y stetig ist und bzgl. y die Lipschitz-Bedingung (5.290) erfüllt, dann kann man beweisen, daß die Lösung des Cauchyschen Anfangswertproblems (5.288) eine *eindeutige* Lösung $y(x)$ besitzt. Im folgenden gehen wir immer von einer eindeutigen Lösung aus.

5.1.1. Euler-Verfahren

Wir wenden uns zunächst der eindimensionalen gewöhnlichen Differentialgleichung (5.288) zu. Um die Gleichung numerisch zu lösen, könnte man den aufretenden Differentialquotienten y' durch den Differenzenquotienten approximieren

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \lim_{h\to 0} \frac{y(x+h)-y(x)}{h} \approx \frac{y(x+h)-y(x)}{h}, \tag{5.291}$$

wobei h eine kleine Schrittweite in x-Richtung ist. Durch Einsetzen der Näherung in (5.288) erhalten wir

$$\frac{y(x+h)-y(x)}{h} = f\,[x, y(x)] \tag{5.292}$$

oder

$$y(x+h) = y(x) + hf\,[x, y(x)]\,. \tag{5.293}$$

Wenn man nun bei $x = a$ mit dem Startwert $y(a) = \alpha$ beginnt, kann man den Wert $y(a+h) = y(a) + hf\,[a, y(a)]$ berechnen. Dies kann man fortsetzen und eine Folge $y(x_k)$ berechnen, wobei $x_{k+1} = x_k + h$ ist. Mit $y_k = y(x_k)$ lautet die diskrete Folge

$$y_{k+1} = y_k + hf\,(x_k, y_k)\,, \qquad k = 0, 1, \ldots. \tag{5.294}$$

höchste Ableitung $y^{(n)}(x)$ auftritt, dann läßt sich immer ein System von n gewöhnlichen Differentialgleichungen erster Ordnung konstruieren. Dazu definiert man zusätzlich zu $y_1 = y^{(n)}$ noch weitere Funktionen als $y_2 = y^{(n-1)}$, $y_3 = y^{(n-2)}$, etc. und betrachtet nur deren erste Ableitungen.

[4] Mit $y \in \mathbb{R}^n$ und $D \in \mathbb{R} \times \mathbb{R}^n$ gilt diese Definition auch für Systeme von gewöhnlichen Differentialgleichungen erster Ordnung.

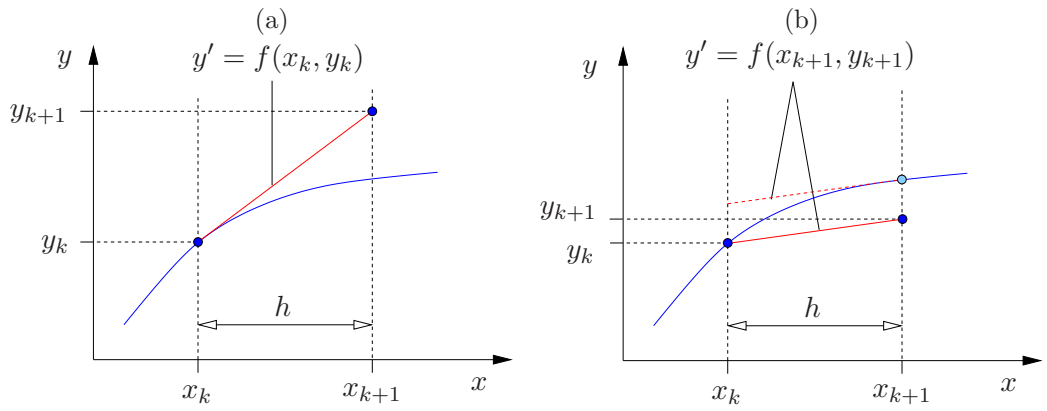


Abbildung 5.2.: Geometrische Interpretation des Eulerverfahrens. Die exakte Lösung ist als blaue Kurve dargestellt. (a) Beim Vorwärts-Euler-Verfahren (5.294) wird die Steigung y'_k am Punkt x_k verwendet, um den Zuwachs $y_{k+1} - y_k = \int_{x_k}^{x_{k+1}} y'(x)\,\mathrm{d}x \approx y'(x_k) \int_{x_k}^{x_{k+1}} \mathrm{d}x$ zu approximieren; (b) beim Rückwärts-Euler-Verfahren (5.298) wird dazu die Steigung y'_{k+1} am Punkt x_{k+1} verwendet.

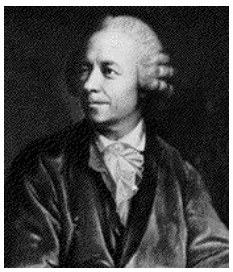
Die Folge $y_k = y(x_k)$ ist eine diskrete Approximation der exakten kontinuierlichen Lösung $y(x)$. Dieses Methode nennt man *Euler-Verfahren*, genauer gesagt, Vorwärts-Euler-Verfahren. Das Vorwärts-Euler-Verfahren bezeichnet man als *explizit*, da der neu zu berechnende Wert y_{k+1} ausschließlich von explizit bekannten (schon vorher berechneten) Größen bei $x_k, \ldots, x_1$ abhängt.

Man kann das Euler-Verfahren bzw. y' auch aus der Taylor-Entwicklung von $y(x)$ um den Punkt x_k erhalten. Dazu betrachten wir

$$y(x) = y(x_k) + (x - x_k)y'(x_k) + O\left[(x - x_k)^2\right]. \tag{5.295}$$

Für $x = x_{k+1}$ und $h = x_{k+1} - x_k$ erhalten wir daraus

$$y'_k = \frac{y_{k+1} - y_k}{h} + O\left(h\right). \tag{5.296}$$



Leonhard Euler
1707–1783

Hieran erkennt man, daß das Euler-Verfahren *von erster Ordnung* ist. D.h., der Fehler beim Ersetzen des Differentialquotienten durch den Differenzenquotienten ist von der Größenordnung $O\left(h^1\right)$. Geometrisch bedeutet das Euler-Verfahren die Approximation der exakten (variierenden) Ableitung y' im Intervall $[x_k, x_{k+1}]$ durch die Steigung der Tangente im Punkte x_k (Abb. 5.2a).

Wenn man anstelle von (5.292) die rechte Seite $f[x, y(x)]$ der Gleichung bei x_{k+1} auswertet, erhält man das *Rückwärts-Euler-Verfahren*

$$\frac{y(x+h) - y(x)}{h} = f\left[x + h, y(x+h)\right] \tag{5.297}$$

bzw.

$$y_{k+1} = y_k + hf\left(x_{k+1}, y_{k+1}\right). \tag{5.298}$$

Offenbar ist auch das Rückwärts-Euler-Verfahren von erster Ordnung. Im Gegensatz zum Vorwärts-Euler-Verfahren (5.294) ist das Rückwärts-Euler-Verfahren (5.298) *implizit*: Man kann (5.298) nicht einfach nach y_{k+1} auflösen, da auch die rechte Seite über $f(x_{k+1}, y_{k+1})$ noch von y_{k+1} abhängt. Das implizite Verfahren ist deshalb nicht so einfach zu verwenden, denn man muß für jeden Integrationsschritt eine (i.a. nichtlineare) Gleichung lösen. Dafür hat das implizite Verfahren aber bessere Stabilitätseigenschaften (Kap. 5.2). Beide Euler-Verfahren gehören zur Klasse der *Einschritt-Verfahren*, da man für die Berechnung von y_{k+1} nur die Informationen von einem einzigen vorhergehenden Punkt (x_k) benötigt.[5]

Im allgemeinen ist die numerisch berechnete Lösung fehlerbehaftet. Normalerweise spielt der Rundungsfehler selbst bei den kleinsten in der Praxis verwendeten Schrittweiten h keine Rolle. Der dominante Fehler ist der Diskretisierungsfehler.

Der Fehlerterm der Größenordnung $O(h)$ in (5.296) ist ein *lokaler Diskretisierungsfehler*, der bei jedem Integrationsschritt entsteht. Man kann zeigen, daß sich die Fehlerordnung des lokalen Diskretisierungsfehlers auf einem *endlichen*(!) Intervall auf den *globalen Diskretisierungsfehler*

$$e(h) = \max_i \left| y_i^{\text{Gitter}} - y^{\text{exakt}}(x_i) \right| = O(h) \tag{5.299}$$

überträgt, wobei $i = 1, \ldots, N$ über alle Gitterpunkte des endlichen Intervalls $x \in [a, b]$ läuft, y_i^{Gitter} die mittels Euler-Verfahren berechnete Lösung an der Stelle x_i ist und $y^{\text{exakt}}(x_i)$ die exakte Lösung darstellt.[6]

5.1.2. Verbesserungen des Euler-Verfahren

John Crank
1916–2006

Der relativ große Fehler $O(h)$ des Euler-Verfahrens ist in vielen Anwendungen nicht akzeptabel. Wenn man das explizite und das implizite Euler-Verfahren gleichgewichtig kombiniert, erhält man das *Crank-Nicolson-Verfahren*

$$y_{k+1} = y_k + \frac{h}{2}\left[f\left(x_k, y_k\right) + f\left(x_{k+1}, y_{k+1}\right)\right]. \tag{5.300}$$

[5] Als Übungsbeispiel kann man ja einmal

$$y'(x) = y^2(x) + 2x - x^4 \qquad \text{mit} \quad y(0) = 0$$

mit Hilfe des Euler-Verfahrens berechnen und mit der exakten Lösung $y(x) = x^2$ vergleichen.

[6] Voraussetzung für dieses Verhalten des globalen Fehlers ist, daß $f[x, y(x)]$ eine beschränkte Ableitung bzgl. y besitzt und die zweite Ableitung von y im Intervall beschränkt ist. Außerdem gilt die Aussage für ein endliches Intervall $[a, b]$. Für den Strahl $x \in [a, \infty]$ kann der Fehler für $x \to \infty$ durchaus über alle Grenzen wachsen (Beispiel: Harmonischer Oszillator $y'' + y = 0$, Abb. 5.7b).

Es ist implizit. Aber durch die symmetrische Bildung des finiten Differenzenquotienten $(y_{k+1} - y_k)/h$ und Auswertung von f bezüglich $k + 1/2$ ist dieses Verfahren von zweiter Ordnung $O(h^2)$ genau. Für gewöhnliche Differentialgleichungen ist dies leider immer noch nicht genau genug.

Um explizite Verfahren höherer Genauigkeit systematisch zu konstruieren, kann man versuchen, mehr Informationen über die Funktion $f(x)$ in das Integrationsschema einzubeziehen. Dazu wird die Kenntnis der höheren Ableitungen von $f(x)$ ausgenutzt. Wir betrachten zunächst die Taylor-Entwicklung von $y(x)$ um den Punkt x_k bis zur zweiten Ordnung

$$y_{k+1} = y_k + hy'_k + \frac{h^2}{2} y''_k + O(h^3). \tag{5.301}$$

Wenn wir den Term zweiter Ordnung berücksichtigen wollen, müssen wir y''_k bereitstellen. Formal ist das nicht schwierig, denn

$$y''(x_k) = \frac{\mathrm{d}}{\mathrm{d}x} f[x_k, y(x_k)] = f^{(1)}(x_k, y_k) = f_x[x_k, y(x_k)] + f_y[x_k, y(x_k)]y'(x_k). \tag{5.302}$$

Damit erhalten wir das Verfahren

$$y_{k+1} = y_k + hf(x_k, y_k) + \frac{h^2}{2}\big[f_x(x_k, y_k) + f_y(x_k, y_k)f(x_k, y_k)\big] + O(h^3). \tag{5.303}$$

Phyllis Nicolson
1917– 1968

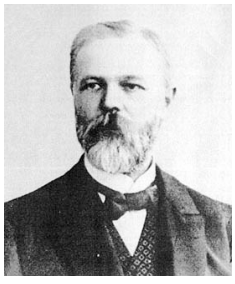
Es ist von zweiter Ordnung, da der Fehler in der hieraus gebildeten Ableitung y'_k von der Größenordnung $O(h^2)$ ist.

Man kann dieses Verfahren verallgemeinern und das R-stufige *Taylor-Verfahren* ansetzen als Taylor-Entwicklung

$$\begin{aligned} y_{k+1} &= \sum_{r=0}^{R} \frac{h^r}{r!} y^{(r)}(x_k, y_k) + O(h^{R+1}) \\ &= y_k + h \sum_{r=1}^{R} \frac{h^{r-1}}{r!} f^{(r-1)}(x_k, y_k) + O(h^{R+1}). \end{aligned} \tag{5.304}$$

Die Fehlerordnung des R-stufigen Taylor-Verfahrens ist $O(h^R)$. Für $R > 2$ wird dieses Konstruktionsprinzip jedoch mühsam, da höhere totale Ableitungen von $f(x_k, y_k)$ berechnet werden müssen.

5.1.3. Runge-Kutta-Verfahren



Karl Heun
1859–1929

Um die Berechnung der Ableitungen von f zu vermeiden, kann man die Ableitungen durch Differenzenquotienten ersetzen. Für $R = 2$ sind dann nur Auswertungen von f bei y_k erforderlich. Dazu diskretisiert man $y^{(2)}(x_k, y_k)$

$$\begin{aligned} y^{(2)}(x_k, y_k) = f^{(1)}(x_k, y_k) &\approx \frac{1}{h}\big[f(x_{k+1}, y_{k+1}) - f(x_k, y_k)\big] \\ &\approx \frac{1}{h}\Big\{f\left[x_{k+1}, y_k + hf(x_k, y_k)\right] - f(x_k, y_k)\Big\}. \end{aligned} \tag{5.305}$$

Damit erhält man anstelle von (5.303)

$$y_{k+1} = y_k + hf(x_k, y_k) + \frac{h^2}{2} f^{(1)}(x_k, y_k) \tag{5.306}$$

$$\approx y_k + hf(x_k, y_k) + \frac{h}{2}\Big\{ f\left[x_{k+1}, y_k + hf(x_k, y_k)\right] - f(x_k, y_k)\Big\}. \tag{5.307}$$

Man kann zeigen, daß dieses Verfahren ebenfalls von zweiter Ordnung ist. Der lokale Fehler beträgt also $O(h^2)$. Man nennt es das *Heun-Verfahren*. Es ist ein *Runge-Kutta-Verfahren zweiter Ordnung*

$$y_{k+1} = y_k + \frac{h}{2}\Big\{ f(x_k, y_k) + f\left[x_{k+1}, y_k + hf(x_k, y_k)\right] \Big\}. \tag{5.308}$$

Im Vergleich zum Euler-Verfahren (5.294) wurde $f(x_k, y_k)$ durch den Mittelwert von $f(x_k, y_k)$ und $f\left[x_{k+1}, y_k + hf(x_k, y_k)\right]$ ersetzt. Dies ist so ähnlich wie beim Crank-Nicholson-Verfahren (5.300), hier nur explizit formuliert.

Allgemein setzt man die Runge-Kutta-Verfahren an als

$$y_{k+1} = y_k + hF\left(x_k, y_k\right) = y_k + h\sum_{r=1}^{R} c_r K_r(x_k, y_k), \tag{5.309}$$

wobei

$$K_1 = f(x_k, y_k), \tag{5.310a}$$

$$K_r = f\left(x_k + a_r h, y_k + h\sum_{s=1}^{r-1} b_{rs} K_s\right), \quad r = 2, \ldots, R \tag{5.310b}$$

Funktionsauswertungen an gewissen Zwischenstellen sind. Die unbekannten Parameter c_r, a_r und b_{rs} werden so bestimmt, daß dieses Runge-Kutta-Verfahren dem R-stufigen Taylor-Verfahren (5.304) entspricht

$$F\left(x_k, y_k\right) = \sum_{r=1}^{R} c_r K_r(x_k, y_k) \overset{!}{=} \sum_{r=1}^{R} \frac{h^{r-1}}{r!} f^{(r-1)}(x_k, y_k) + O(h^R), \tag{5.311}$$

bis zu einer möglichst hohen Fehlerordnung m. Idealerweise ist $m = R$. Um einer Koeffizientenvergleich in (5.311) durchzuführen, werden alle Funktionswerte von f (an den verschobenen Stellen) im Runge-Kutta-Ansatz (5.309) mittels Taylor-Entwicklung durch Funktionswerte am Punkt (x_k, y_k) ausgedrückt.

Betrachte zum Beispiel $R = 2$. Dann ist

$$\begin{aligned} F\left(x, y\right) &= c_1 K_1 + c_2 K_2 = c_1 f(x_k, y_k) + c_2 f\left[x_k + a_2 h, y_k + h b_{21} f(x_k, y_k)\right] \\ &\overset{\text{Taylor}}{=} c_1 f(x_k, y_k) + c_2\left[f\left(x_k, y_k\right) + a_2 h f_x\left(x_k, y_k\right) + h b_{21} f(x_k, y_k) f_y\left(x_k, y_k\right)\right] \\ &\overset{(5.311)}{=} f\left(x_k, y_k\right) + \frac{h}{2}\left[f_x\left(x_k, y_k\right) + f\left(x_k, y_k\right) f_y\left(x_k, y_k\right)\right]. \end{aligned} \tag{5.312}$$

Der Koeffizientenvergleich für das letzte Gleichheitszeichen liefert

$$c_1 + c_2 = 1, \tag{5.313a}$$

$$c_2 a_2 = \frac{1}{2}, \tag{5.313b}$$

$$c_2 b_{21} = \frac{1}{2}. \tag{5.313c}$$

Martin Wilhelm Kutta
1867–1948

Dies sind drei Gleichungen für vier Koeffizienten. Es gibt also mehr als eine Lösung. Eine mögliche Lösung ist das obige Heun-Verfahren (5.308). Es entspricht der Lösung $c_1 = c_2 = \frac{1}{2}$ und $a_2 = b_{21} = 1$. Ein anderes Runge-Kutta-Verfahren erhält man mit der Wahl $c_1 = 0$, $c_2 = 1$ und $a_2 = b_{21} = \frac{1}{2}$. Sie führt auf

$$y_{k+1} = y_k + hf\left[x_{k+1/2}, y_k + \frac{h}{2} f\left(x_k, y_k\right)\right]. \tag{5.314}$$

Dies ist eine Art modifiziertes Euler-Verfahren. Wie das Heun-Verfahren ist es ebenfalls von zweiter Ordnung.

Auf diese Weise kann man nun weitere Verfahren mit höherer Genauigkeit konstruieren. Für $R = 4$ ergeben sich 13 freie Parameter. Für diese erhält man 11 Bestimmungsgleichungen zur Konstruktion eines Verfahrens vierter Ordnung. Unter den vielen Möglichkeiten, die sich daraus ergeben, wird meistens das folgende *Runge-Kutta-Verfahren vierter Ordnung* verwendet

$$y_{k+1} = y_k + \frac{h}{6}\left(F_1 + 2F_2 + 2F_3 + F_4\right) \tag{5.315a}$$

mit

$$F_1 = f(x_k, y_k), \tag{5.315b}$$

$$F_2 = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} F_1\right), \tag{5.315c}$$

$$F_3 = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} F_2\right), \tag{5.315d}$$

$$F_4 = f\left(x_{k+1}, y_k + hF_3\right). \tag{5.315e}$$

Es ist explizit, wie auch das Euler-Verfahren (5.294) und das Heun-Verfahren 2-ter Ordnung (5.308). Denn wenn man y_k berechnet hat, kann man sukzessive $F_1 \to F_2 \to F_3 \to F_4$ berechnen und in (5.315a) einsetzen. Es sind lediglich vier Auswertungen von f erforderlich. Die so konstruierten Runge-Kutta-Verfahren sind *Einschritt-Verfahren*, da zur Berechnung von y_{k+1} lediglich die Kenntnis von y_k erforderlich ist.

Allgemein kann man zeigen: Sei p die Ordnung eines Runge-Kutta-Verfahrens. Dann sind für $p \leq 4$ genau p Auswertungen von f erforderlich. Für $5 \leq p \leq 6$ sind $p + 1$ Auswertungen nötig, für $p = 7$ sind es $p + 2$ Auswertungen und für

$p \geq 8$ mindestens $p+3$. In vielen Fällen stellt das Runge-Kutta-Verfahren vierter Ordnung einen guten Kompromiß zwischen Aufwand und Genauigkeit dar.

Bemerkung: Wenn $f = f(x)$ wie in (5.286) nicht von y abhängt, dann lautet das Euler-Verfahren (die variable Schrittweite ist $h_{k+1} = x_{k+1} - x_k$)

$$y_{k+1} = y_k + h_{k+1} f(x_k). \tag{5.316}$$

Mit der Anfangsbedingung $y_0 = 0$ wird dann

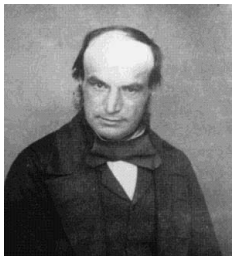
$$y_2 = y_1 + h_2 f(x_1) = \overbrace{\underbrace{y_0}_{=0} + h_1 f(x_0)}^{y_1} + h_2 f(x_1). \tag{5.317}$$

Für $k = N$ erhalten wir deshalb

$$y_N = \sum_{n=1}^{N} h_n f(x_{n-1}) = I_{\mathrm{SR}}(f). \tag{5.318}$$

Dies ist gerade die summierte Rechteck-Regel (4.211a). In analoger Weise kann man für $f = f(x)$ und der Anfangsbedingung $y_0 = 0$ (d.h. für (5.286)) zeigen, daß das Runge-Kutta-Verfahren zweiter Ordnung (5.308) der Trapez-Regel entspricht und das Runge-Kutta-Verfahren vierter Ordnung (5.315) der Simpson-Regel.

5.1.4. Adams-Bashforth-Verfahren



John Couch Adams
1819–1892

Eine andere Möglichkeit zur Konstruktion von Verfahren höherer Ordnung besteht darin, Informationen (y'_k) an zuvor berechneten Punkten x_k zu verwenden. Damit gelangt man zur Klasse der *Adams-Bashforth-Verfahren*.[7] Dahinter steht die Idee, eine genauere Extrapolation von y_{k+1} zu erhalten, indem man die Ableitung y' im Intervall $[x_k, x_{k+1}]$ durch ein Polynom approximiert, welches man durch Interpolation aus den zuvor berechneten Werten $y'_n = f(x_n, y_n)$ mit $n \leq k$ erhält.

Um diese Strategie anzuwenden, setzen wir den Zuwachs an

$$y_{k+1} - y_k = \int_{x_k}^{x_{k+1}} y'(x)\,\mathrm{d}x = \int_{x_k}^{x_{k+1}} f[x, y(x)]\,\mathrm{d}x \overset{!}{\approx} \int_{x_k}^{x_{k+1}} p(x)\,\mathrm{d}x, \tag{5.319}$$

wobei $p(x)$ das Polynom ist, welches einige vorherige Funktionswerte $f_k = f[x_k, y_k]$ interpoliert (Abb. 5.3a). Man erhält so das Adams-Bashforth-Verfahren

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} p(x)\,\mathrm{d}x. \tag{5.320}$$

[7] Francis Bashforth, 1819–1912: Mathematiker und Ballistik-Experte. Veröffentlichte zusammen mit Adams 1883 einen Aufsatz (*An Attempt to Test the Theories of Capillary Action*, Cambridge University Press, 1883), in welchem die Methode von Adams zur Berechnung von Tropfenformen verwendet wird.

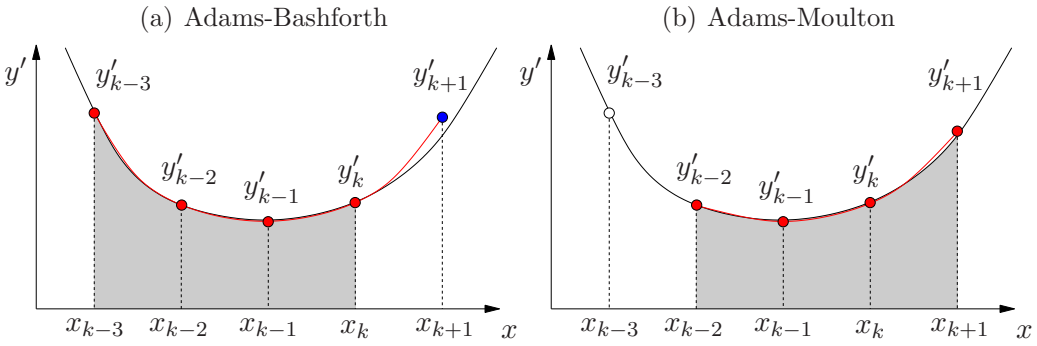


Abbildung 5.3.: Schematische Darstellung der Adams-Bashforth- (a) und der Adams-Moulton-Strategie (b). Bei Adams-Bashforth-Verfahren 4. Ordnung (a) werden die vier zuletzt berechneten Funktionswerte y_k verwendet, um die Ableitungen $y'_k = f(x_k, y_k)$ (rote Punkte) zu gewinnen und daraus y'_{k+1} (blauer Punkt) zu prognostizieren. Aus der Integration über $[x_k, x_{k+1}]$ gewinnt man dann $y_{k+1} - y_k$. Beim Adams-Moulton-Verfahren 4. Ordnung (b) werden die vier roten Punkte verwendet, um y'_{k+1} *implizit* vorherzusagen. Der Bereich, über den sich die Interpolation (rote Kurve) erstreckt, ist grau angedeutet. Die dünne schwarze Kurve soll die exakte Funktion $y' = f(x, y)$ andeuten.

Die einfachste Möglichkeit der Interpolation ist es, das Polynom vom Grade Null zu wählen, nämlich die Konstante $p = f(x_k, y_k) = \text{const}$. Dies liefert $\int_{x_k}^{x_{k+1}} p(x)\,\mathrm{d}x = hf(x_k, y_k)$. Mit dieser Wahl erhält man gerade das Euler-Verfahren (5.294). In der nächsthöheren Ordnung würde man für $p(x)$ die lineare Funktion verwenden, welche die Funktionswerte $f_{k-1} = f(x_{k-1}, y_{k-1})$ und $f_k = f(x_k, y_k)$ interpoliert. Um diese Interpolation zu konstruieren, ist es am bequemsten, die Newton-Interpolation (3.100) zu verwenden. Im vorliegenden Fall liegt es nahe, die finiten Differenzen nicht wie in (3.96) in Vorwärtsrichtung zu bilden, sondern in Rückwärtsrichtung. Damit wird

$$\Delta f_k := f_{k-1} - f_k. \tag{5.321}$$

In Analogie zu (3.100) erhalten wir damit das lineare (Newton-) Interpolationspolynom

$$p(x) = p_1 = f_k - \frac{x - x_k}{h}\Delta f_k, \tag{5.322}$$

wobei das Minuszeichen erforderlich wurde, da wir die positive Schrittweite $h = |x_{k-1} - x_k|$ verwenden. Wenn wir nun das Integral in (5.320) ausführen, erhalten wir

$$y_{k+1} = y_k + \left[f_k x - \frac{1}{2}\frac{(x - x_k)^2}{h}\Delta f_k \right]_{x_k}^{x_{k+1}} = y_k + hf_k - \frac{h}{2}\Delta f_k. \tag{5.323}$$

Dies ist das *Adams-Bashforth-Verfahren zweiter Ordnung*. Wenn man die Differenz

Δf_k einsetzt, erhalten wir

$$y_{k+1} = y_k + hf_k - \frac{h}{2}\left(f_{k-1} - f_k\right) = y_k + \frac{h}{2}\left(3f_k - f_{k-1}\right). \tag{5.324}$$

Wir können nun so fortfahren und durch das Einbeziehen weiterer Punkte in die Newton-Interpolation (3.100) Adams-Bashforth-Verfahren höherer Ordnung entwickeln. Das Newton-Polynom zweiter Ordnung, welches die Punkte (x_{k-2}, f_{k-2}), (x_{k-1}, f_{k-1}) und (x_k, f_k) interpoliert, lautet

$$p_2 = p_1 + \frac{(x-x_k)(x-x_{k-1})}{2h^2}\Delta^2 f_k = f_k - \frac{x-x_k}{h}\Delta f_k + \frac{(x-x_k)(x-x_{k-1})}{2h^2}\Delta^2 f_k. \tag{5.325}$$

Durch Integration erhalten wir gemäß (5.320)[8]

$$\begin{aligned} y_{k+1} = y_k + \Bigg\{ & f_k x - \frac{1}{2}\frac{(x-x_k)^2}{h}\Delta f_k \\ & + \frac{1}{2h^2}\left[\frac{1}{3}(x-x_{k-1})^3 - \frac{1}{2}(x_k - x_{k-1})(x-x_{k-1})^2\right]\Delta^2 f_k \Bigg\}_{x_k}^{x_{k+1}}. \end{aligned} \tag{5.326}$$

Dies führt auf

$$\begin{aligned} y_{k+1} &= y_k + hf_k - \frac{h}{2}\Delta f_k + \frac{1}{2h^2}\left(\frac{1}{3}8h^3 - \frac{1}{3}h^3 - \frac{h}{2}4h^2 + \frac{h}{2}h^2\right)\Delta^2 f_k \\ &= y_k + hf_k - \frac{h}{2}\Delta f_k + \frac{5h}{12}\Delta^2 f_k \end{aligned} \tag{5.327}$$

Wenn wir die Rückwärtsdifferenz zweiter Ordnung

$$\begin{aligned} \Delta^2 f_k &= \Delta\left(f_{k-1} - f_k\right) = \left(f_{k-2} - f_{k-1}\right) - \left(f_{k-1} - f_k\right) \\ &= f_{k-2} - 2f_{k-1} + f_k \end{aligned} \tag{5.328}$$

einsetzen, erhalten wir

$$y_{k+1} = y_k + \frac{h}{2}\left(3f_k - f_{k-1}\right) + \frac{5h}{12}\left(f_{k-2} - 2f_{k-1} + f_k\right), \tag{5.329}$$

oder

$$y_{k+1} = y_k + \frac{h}{12}\left(23f_k - 16f_{k-1} + 5f_{k-2}\right). \tag{5.330}$$

Dies ist das *Adams-Bashforth-Verfahren dritter Ordnung*.

In analoger Weise kann man das *Adams-Bashforth-Verfahren vierter Ordnung* konstruieren (Hausaufgabe), und erhält

$$y_{k+1} = y_k + \frac{h}{24}\left(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}\right). \tag{5.331}$$

[8] Die Ableitung des Ausdrucks in [...] in (5.326) liefert $(x-x_k)(x-x_{k-1})$.

Dies kann man nun bis zu beliebiger Ordnung fortsetzen. Die Adams-Bashforth-Verfahren sind *Mehrschritt-Verfahren*, da zur Berechnung der neuen Näherung y_{k+1} die Werte von y_k aus vorangegangenen Schritten verwendet werden. Die Strategie ist in Abb. 5.3a für das Adams-Bashforth-Verfahren vierter Ordnung (5.331) dargestellt.

Anders als die Einschritt-Verfahren, wie etwa die Runge-Kutta-Verfahren, tritt bei Mehrschritt-Verfahren ganz allgemein die Problematik der Startwerte auf. Denn zu Beginn der Rechnung liegen noch nicht hinreichend viele berechnete Funktionswerte vor, um ein Mehrschritt-Verfahren anwenden zu können. In diesem Fall behilft man sich meist mit der Verwendung eines Einschritt-Verfahrens derselben Fehlerordnung, bis man hinreichend viele Punkte berechnet hat. Ähnliches gilt, wenn man innerhalb der Rechnung die Schrittweite h wechseln möchte.

Mehrschritt-Verfahren besitzen gegenüber Einschritt-Verfahren den Vorteil, daß man auf berechnete Werte zurückgreift, die man im Speicher halten kann. Es ist bei jedem Schritt die Funktion f nur einmal auszuwerten. Ist diese Auswertung rechenintensiv, werden Einschritt-Verfahren entsprechend aufwendig.

5.1.5. Prädiktor-Korrektor-Verfahren

Adams-Bashforth-Verfahren haben den Nachteil, daß sie manchmal instabil werden können. Ursache dafür ist die Vorhersage eines Punktes (x_{k+1}, y_{k+1}), der außerhalb des Interpolationsintervalls $(x \le x_k)$ liegt (Abb. 5.3a). Um dieses Defizit zu beheben, kann man die Interpolation formal bis auf den Punkt x_{k+1} ausdehnen (Abb. 5.3b). Dies führt auf die *Adams-Moulton-Verfahren*.

Im Fall der linearen Interpolation von $y' = f$ zwischen den Punkten k und $k+1$ können wir die Integration in (5.320) direkt ausführen und erhalten

Forest Ray Moulton
1872–1952

$$y_{k+1} = y_k + \underbrace{\frac{h}{2}\left(f_{k+1} + f_k\right)}_{\text{Trapez-Regel}} . \tag{5.332}$$

Dies ist das *Adams-Moulton-Verfahren zweiter Ordnung*. Es ist identisch mit dem Crank-Nicolson-Verfahren (5.300).[9] Im Fall eines kubischen Interpolationspolynoms durch die vier Punkte (x_{k-2}, f_{k-2}), (x_{k-1}, f_{k-1}), (x_k, f_k) und (x_{k+1}, f_{k+1}) erhält man das *Adams-Moulton-Verfahren vierter Ordnung*

$$y_{k+1} = y_k + \frac{h}{24}\left(9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}\right) . \tag{5.333}$$

An den Adams-Moulton-Formeln sieht man, daß auf der rechten Seite auch $f_{k+1} = f(x_{k+1}, y_{k+1})$ auszuwerten ist. Der Wert y_{k+1} liegt aber noch gar nicht vor.

[9] Das Adams-Moulton-Verfahren erster Ordnung ist identisch mit dem Rückwärts-Euler-Verfahren (5.297).

Daher muß man für die Adams-Moulton-Formeln in jedem Schritt eine (i.a. nichtlineare) Gleichung lösen, um y_{k+1} zu berechnen. Die Adams-Moulton-Verfahren sind also *implizit*, im Gegensatz zu den Adams-Bashforth-Verfahren, die *explizit* sind.

Man könnte nun versuchen, die Adams-Moulton-Formeln mittels Newton-Iteration oder einem anderen Verfahren zu lösen. Es gibt jedoch einen einfacheren Zugang. Dabei wird zunächst eine explizite Formel gleicher Fehlerordnung verwendet, um y_{k+1} zu berechnen. Dies der ein Vorhersage-Schritt (*Prädiktor*-Schritt). Danach wird der Vorhersage-Wert in die implizite Formel eingesetzt und damit korrigiert (*Korrektor*-Schritt). So erhält man zum Beispiel das *Prädiktor-Korrektor-Verfahren vierter Ordnung*

$$y_{k+1}^{(p)} = y_k + \frac{h}{24}\left(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}\right), \tag{5.334a}$$

$$f_{k+1}^{(p)} = f\left(x_{k+1}, y_{k+1}^{(p)}\right), \tag{5.334b}$$

$$y_{k+1} = y_k + \frac{h}{24}\left(9f_{k+1}^{(p)} + 19f_k - 5f_{k-1} + f_{k-2}\right). \tag{5.334c}$$

Hier ist (5.334a) der Prädiktor-Schritt, in dem das Adams-Bashforth-Verfahren (5.331) verwendet wird, und (5.334c) der Korrektor-Schritt. Das Verfahren (5.334) ist damit vollständig explizit. Wenn man das Verfahren in dieser Art implementiert, was der Regelfall ist, dann hat man zwar *nur* eine Näherung für das voll implizite Adams-Moulton-Verfahren (5.333) gefunden. Da aber auch eine genauere Lösung der impliziten Gleichung (5.333) immer noch fehlerbehaftet mit Fehlerordnung $O(h^4)$ ist, macht eine genauere Lösung wenig Sinn. Im Prinzip könnte man jedoch weiter iterieren und die Schritte (5.334b) und (5.334c) so lange wiederholen, bis Konvergenz erreicht ist. Dies würde einer iterativen Lösung von (5.333) entsprechen.

5.1.6. Systeme von Differentialgleichungen

Die meisten Probleme in Naturwissenschaft und Technik sind nicht von erster Ordnung. Man kann aber jede gewöhnliche Differentialgleichung höherer Ordnung in ein System von Differentialgleichungen erster Ordnung überführen. Manche Probleme treten auch in natürlicher Weise als großes System von Differentialgleichungen auf. Ein Beispiel ist die Modellierung von Verbrennungsprozessen. Hierbei sind Gleichungen für die zeitliche Entwicklung der Konzentrationen aller an der Reaktion beteiligten chemischen Verbindungen (Spezies) zu lösen. Selbst bei einfachen chemischen Reaktionen sind leicht 100 oder mehr chemische Komponenten beteiligt, da auch kurzlebige Zwischenprodukte zu berücksichtigen sind. Dann sind simultan sehr viele Differentialgleichungen zu lösen, da die Reaktionsraten der einzelnen Spezies von den Konzentrationen aller an der Reaktion beteiligten Spezies und dem thermodynamischen Zustand abhängen.

Die Erweiterung der bisher besprochenen Algorithmen auf Systeme von Differentialgleichungen ist relativ einfach, wenn wir y und f vektoriell auffassen. Mit

$\vec{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ und $\vec{f} = (f_1, \ldots, f_N)^{\mathrm{T}}$ lautet das zu lösende Problem dann

$$\vec{y}\,'(x) = \vec{f}\big[x, \vec{y}(x)\big]. \tag{5.335}$$

Man nennt $\vec{f}$ auch den *Fluß*. Die unabhängige Variable x spielt meistens die Rolle der Zeit. Für das einfache explizite Euler-Verfahren gilt dann

$$\vec{y}_{k+1} = \vec{y}_k + h\vec{f}\big[x_k, \vec{y}(x_k)\big], \tag{5.336}$$

oder in Komponenten ausgeschrieben

$$\begin{aligned} y_{1,k+1} &= y_{1,k} + hf_1\left(x_k, y_{1,k}, \ldots, y_{N,k}\right), \\ &\vdots \\ y_{N,k+1} &= y_{N,k} + hf_N\left(x_k, y_{1,k}, \ldots, y_{N,k}\right). \end{aligned} \tag{5.337}$$

Bei der Notation $y_{n,k}$ bezeichnet der erste Index n die Komponente des Vektors, während der zweite Index die unabhängige Variable indiziert (Zeitpunkt). Auch andere Verfahren lassen sich leicht vektoriell schreiben. Beispielsweise lautet das Runge-Kutta-Verfahren zweiter Ordnung (5.308) für Systeme

$$\vec{y}_{k+1} = \vec{y}_k + \frac{h}{2}\left\{\vec{f}(x_k, \vec{y}_k) + \vec{f}\left[x_{k+1}, \vec{y}_k + h\vec{f}(x_k, \vec{y}_k)\right]\right\}. \tag{5.338}$$

Das Adams-Bashforth-Verfahren zweiter Ordnung (5.324) läßt sich auch leicht vektoriell schreiben als

$$\vec{y}_{k+1} = \vec{y}_k + \frac{h}{2}\left(3\vec{f}_k - \vec{f}_{k-1}\right). \tag{5.339}$$

5.1.7. Symplektische Integration

Es gibt eine wichtige Klassen von Problemen, welche ganz bestimmte exakte Erhaltungsgrößen besitzen. Ein wichtiges Beispiel sind *Hamilton-Systeme*. Deren zeitliche Entwicklung hat immer die (sogenannte kanonische) Form

$$\dot{\vec{q}} = \frac{\partial H(\vec{p}, \vec{q})}{\partial \vec{p}}, \tag{5.340a}$$

$$\dot{\vec{p}} = -\frac{\partial H(\vec{p}, \vec{q})}{\partial \vec{q}}, \tag{5.340b}$$

wobei $\vec{q}$ und $\vec{p}$ verallgemeinerte Koordinaten und Impulse sind (kanonische Koordinaten). Der *Phasenraum* wird durch $\vec{q}$ und $\vec{p}$ aufgespannt. Aufgrund der kanonischen Struktur der Hamiltonschen Gleichungen (5.340) kann der Fluß im Phasenraum (rechte Seite von (5.340)) durch Ableitungen der *Hamilton-Funktion* $H(\vec{q}, \vec{p})$ dargestellt werden. Irgendein Zustand des Systems ist dann eindeutig durch die Angabe von $(\vec{q}, \vec{p})$ festgelegt, und bei Kenntnis von $H(\vec{p}, \vec{q})$ auch die weitere zeitliche Entwicklung.

Für Hamilton-Systeme gilt der *Liouvillesche Satz*. Er besagt, daß ein beliebiges markiertes Phasenraum-Volumen bei der zeitlichen Entwicklung erhalten bleibt. Dies bedeutet, daß der Fluß $\vec{f}(\vec{q},\vec{p})$ im Phasenraum inkompressibel ist, d.h.

$$\nabla\cdot\vec{f}=\nabla\cdot\begin{pmatrix}\dfrac{\partial H(\vec{p},\vec{q})}{\partial\vec{p}}\\ -\dfrac{\partial H(\vec{p},\vec{q})}{\partial\vec{q}}\end{pmatrix}=\frac{\partial}{\partial\vec{q}}\cdot\frac{\partial H(\vec{p},\vec{q})}{\partial\vec{p}}-\frac{\partial}{\partial\vec{p}}\cdot\frac{\partial H(\vec{p},\vec{q})}{\partial\vec{q}}=0. \tag{5.341}$$

Für die Integration von Hamilton-Systemen bzw. Systemen, deren Fluß inkompressibel ist (z.B. die Bewegung von markierten Fluidelementen in einer inkompressiblen Strömung), wäre es nun wichtig, wenn die exakten Erhaltungseigenschaften auch vom diskreten numerischen Schema erfüllt würden. Insbesondere sollte das Volumen eines beliebigen markierten Anfangsvolumens unter einem inkompressiblen Fluß auch in der numerischen Lösung exakt konstant bleiben, während es unter dem Einfluß des Flusses transportiert und deformiert wird.

Integrations-Schemata, welche diese Bedingung erfüllen werden *symplektisch* genannt.[10] Alle bisher behandelten Schemata sind nicht symplektisch, mit Ausnahme des Adams-Moulton-Schemas zweiter Ordnung. Wenn man die Differentialgleichungen mit einem nicht-symplektischen Verfahren integriert, gehen die exakte Erhaltungseigenschaften der kontinuierlichen Differentialgleichungen verloren.

Betrachte zum Beispiel das einfache *mathematische Pendel* (keine Dämpfung). Dazu setzen wir in (5.283) die Masse $m=1$, die Länge $l=1$, die Schwerebeschleunigung $g=1$ und keinen Antrieb $a=0$ und erhalten

$$\dot{q}=p, \tag{5.342a}$$
$$\dot{p}=-\sin q. \tag{5.342b}$$

Hierbei ist q der Winkel und p die Winkelgeschwindigkeit (Drehimpuls pro Masse). Dies ist ein Hamilton-System mit der Hamilton-Funktion

$$H(p,q)=E_{\text{kin}}+E_{\text{pot}}=\frac{p^2}{2}-\cos q \tag{5.343}$$

Durch Ableiten entsprechend (5.340) kann man die Bewegungsgleichungen (5.342) verifizieren. Der Fluß ist inkompressibel

$$\begin{pmatrix}\partial_q\\ \partial_p\end{pmatrix}\cdot\begin{pmatrix}p\\ -\sin q\end{pmatrix}=0. \tag{5.344}$$

Die Gesamtenergie (H) bleibt erhalten, denn $\dot{H}=p\dot{p}+\dot{q}\sin q\overset{(5.342)}{=}0$. Daher erfolgt die Bewegung im Phasenraum für kleine Werte von H (Anfangswerte, kleine Pendelausschläge) auch auf geschlossenen Bahnen um die Fixpunkte $(q,p)=(2\pi n,0)$.

[10] Symplektische Integrationsschemata bilden eine Unterklasse der *geometrischen Integrationsschemata*. Letztere erhalten alle geometrischen Eigenschaften eines gegebenen Systems von Differentialgleichungen.

Für niedrige Gesamtenergie $-1 < H < 1$ hat man eine gebundene Pendelbewegung. Für höhere Energien ($H > 1$) hat man offene Trajektorien (überschlagendes Pendel).

Wenn man nun die Bewegung mit einem nicht-symplektischen Verfahren integriert, wird die Energie des Systems nicht konstant bleiben. Typischerweise divergiert oder verschwindet sie für $t \to \infty$. Ersteres passiert zum Beispiel bei Integration mittels Euler-Vorwärts-Verfahren (5.294). Durch Verwendung eines symplektischen Verfahrens kann dies verhindert werden. Man kann das Euler-Verfahren (5.294) derart modifizieren, so daß die Energie exakt erhalten bleibt, indem man eine Gleichung von (5.342) explizit und die jeweils andere implizit auswertet. Dies führt auf das *symplektische Euler-Verfahren* für Hamilton-Systeme (eine der beiden Varianten)

$$q_{k+1} = q_k + h\nabla_p H(q_k, p_{k+1}), \tag{5.345a}$$
$$p_{k+1} = p_k + h\nabla_q H(q_k, p_{k+1}). \tag{5.345b}$$

Das symplektische Euler-Verfahren ist formal implizit. Wenn die Hamilton-Funktion aber in der Form $H(q,p) = E_{\text{kin}}(p) + E_{\text{pot}}(q)$ separiert (wie in unserem Beispiel), dann ist das Verfahren explizit. Für unser Beispiel erhalten wir[11]

$$\begin{pmatrix} q \\ p \end{pmatrix}_{k+1} = \begin{pmatrix} q \\ p \end{pmatrix}_k + h\vec{f}(\vec{q}_k, \vec{p}_{k+1}) = \begin{pmatrix} q \\ p \end{pmatrix}_k + h \begin{pmatrix} p_{k+1} \\ -\sin q_k \end{pmatrix}. \tag{5.346}$$

Beispielrechnungen sind in Abb. 5.4 dargestellt. Beachte, daß die Energie der Bewegung bei Verwendung des Vorwärts-Euler-Verfahrens (schwarz) kontinuierlich anwächst, so daß die gebundene Bewegung in eine ungebundene Bewegung übergeht, wobei die Winkelgeschwindigkeit (p) im Mittel immer weiter zunimmt. Beim symplektischen Euler-Verfahren (blau) bleibt die Energie hingegen erhalten.

Ähnlich wie das symplektische Euler-Verfahren für Hamilton-Systeme ist die *implizite Mittelpunktsregel* (vgl. (4.199))

$$\vec{y}_{k+1} = \vec{y}_k + h\vec{f}\left[\frac{\vec{y}_{k+1} + \vec{y}_k}{2}\right] \tag{5.347}$$

symplektisch. Wenn man im Argument von $\vec{f}$ den Vektor $\vec{y}_{k+1}$ mit Hilfe des Euler-Rückwärtsverfahrens $\vec{y}_{k+1} = \vec{y}_k + h\vec{f}(\vec{y}_{k+1})$ approximiert, erhält man die implizite Mittelpunktsregel auch in der alternativen Form

$$\vec{y}^* = \vec{y}_k + \frac{h}{2}\vec{f}(\vec{y}^*), \tag{5.348a}$$
$$\vec{y}_{k+1} = \vec{y}_k + h\vec{f}(\vec{y}^*). \tag{5.348b}$$

[11] Normalerweise muß man bei den impliziten Verfahren in jedem Zeitschritt ein (eventuell nichtlineares) Gleichungssystem lösen. Beim symplektischen Euler-Verfahren (5.346) für das mathematische Pendel kann man die impliziten Gleichungen jedoch explizit auflösen und erhält

$$q_{k+1} = q_k + hp_k - h^2 \sin(q_k),$$
$$p_{k+1} = p_k - h\sin(q_k).$$

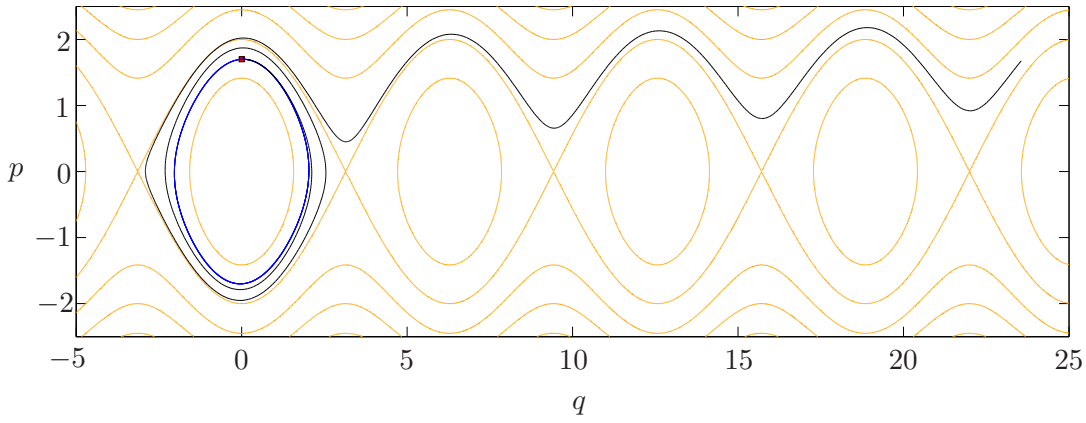


Abbildung 5.4.: Vorwärts-Euler (schwarz) und symplektisches Euler-Verfahren (5.346) (blau) für das mathematische Pendel mit Schrittweite $h = 0.05$ bis $t = 40$. Das rote Quadrat kennzeichnet den gemeinsamen Startpunkt mit Anfangswerten $(q, p) = (\varphi, \dot{\varphi}) = (0, 1.7)$. Die Isolinien von H (Gesamtenergie) sind in orange dargestellt.

Ein weiteres einfaches Verfahren ist die *implizite Trapezregel*

$$\vec{y}_{k+1} = \vec{y}_k + \frac{h}{2}\left[\vec{f}(\vec{y}_{k+1}) + \vec{f}(y_k)\right] \tag{5.349}$$

Dies ist gerade das Adams-Moulton-Verfahren zweiter Ordnung (vgl. (4.200)). Die implizite Trapezregel ist nicht symplektisch, sondern konjugiert symplektisch, besitzt aber ein ähnliches Langzeitverhalten wie eine symplektische Methode. Wenn man die Gleichung auf (5.342) anwendet und durch q und p ausdrückt, ergibt sich

$$\begin{pmatrix} q \\ p \end{pmatrix}_{k+1} = \begin{pmatrix} q \\ p \end{pmatrix}_k + \frac{h}{2}\left[\begin{pmatrix} p \\ -\sin q \end{pmatrix}_{k+1} + \begin{pmatrix} p \\ -\sin q \end{pmatrix}_k\right]. \tag{5.350}$$

Symplektische Methoden finden weite Anwendung bei molekulardynamischen Simulationen, bei der Berechnung von Planetenbewegungen und in vielen anderen Gebieten, bei denen es auf genaue Langzeit-Vorhersagen ankommt.

5.1.8. Weitere Beispielrechnungen

Für die Räuber-Beute-Gleichungen (5.285) gibt es Parameter-Kombinationen, die ein stabiles dynamisches Gleichgewicht beschreiben. Dann können beide Populationen (Räuber und Beute) innerhalb von gewissen Variationen für alle Zeiten fortbestehen. Dies ist beispielsweise dann der Fall, wenn der Räuber in natürlicher Weise dezimiert wird (positive Zerfallsrate $b_Y = 1$) und die Beute permanent Nachwuchs produziert (negative Zerfallsrate $b_X = -0.25$). Da die Beute durch die Räuber dezimiert wird, ist die *Rate* $c_X = -0.01$ sinnvollerweise negativ. Umgekehrt sollte sich die Beute positiv auf die Anzahl der Räuber auswirken ($c_Y = 0.01$). Unter diesen Bedingungen kommt es zu stabilen, phasenverschobenen Oszillationen im

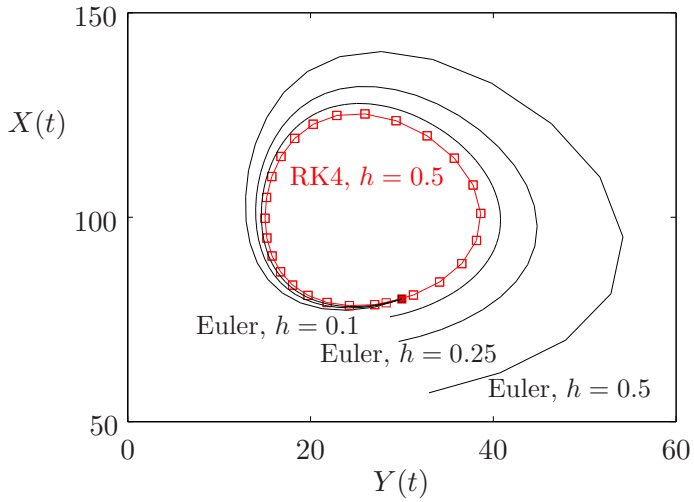


Abbildung 5.5.: Integration des Lotka-Volterra-Modells (5.285) für die Parameter $b_X = -0.25$, $b_Y = 1$, $c_X = -0.01$ und $c_Y = 0.01$. Bei allen Rechnungen wurde die Integration von $t = 0$ bis $t = T = 13$ durchgeführt. Die Schrittweite h ist als Parameter angegeben. Das gefüllte rote Quadrat kennzeichnet die Anfangsbedingung $[X, Y](t = 0) = (80, 30)$. Die rote Kurve ist sehr nahe an der exakten Lösung und wurde mit dem Runge-Kutta-Verfahren vierter Ordnung und Schrittweite $h = 0.5$ berechnet. Die schwarzen Kurven wurden mit dem Euler-Verfahren und Schrittweiten $h = 0.5$, 0.25 und 0.1 (von außen nach innen) berechnet. Alle Kurven werden im Uhrzeigersinn durchlaufen.

Tierbestand. Im einem $[X(t), Y(t)]$-Diagramm zeigt sich dieser Zyklus als eine geschlossene Kurve, die periodisch durchlaufen wird. Dies ist in Abb. 5.5 dargestellt.

Man erkennt, daß das Runge-Kutta-Verfahren vierter Ordnung trotz der großen Schrittweite (rote Linie und offene rote Symbole) sehr gute Ergebnisse liefert. Das einfache Euler-Verfahren (schwarze Kurven) hat die Tendenz, von der *exakten* Lösung wegzulaufen und spiralförmig zu divergieren. Man müßte die Schrittweite des Euler-Verfahrens schon extrem verringern (womit eine extreme Erhöhung der Rechenzeit verbunden wäre), um für einen Umlauf eine Genauigkeit zu erzielen, die mit dem Runge-Kutta-Verfahren vierter Ordnung vergleichbar wäre.

Abbildung 5.6 zeigt den Vergleich zwischen den Runge-Kutta-Verfahren 4-ter (rot) und 2-ter Ordnung (blau) für das Lotka-Volterra-Modell (5.285) mit Parametern wie in Abb. 5.5. Für die extrem große Schrittweite $h = 1$ zeigt sich, daß auch die Lösung mit dem Runge-Kutta-Verfahren 2-ter Ordnung leicht nach außen abdriftet.

In Abb. 5.7a ist schließlich ein Vergleich des Adams-Bashforth-Verfahrens zweiter Ordnung mit dem Prädiktor-Korrektor-Verfahren auf Basis des Adams-Moulton-Verfahrens analog zu (5.334) gezeigt. Man erkennt das spiralige Auslaufen des Adams-Bashforth-Verfahrens im Vergleich zum Runge-Kutta-Verfahren vierter Ordnung. Im Gegensatz dazu zeigt das Prädiktor-Korrektor-Verfahren eine Einwärts-Spirale. Dieser Verhalten ist typisch und kann auch für das konservati-

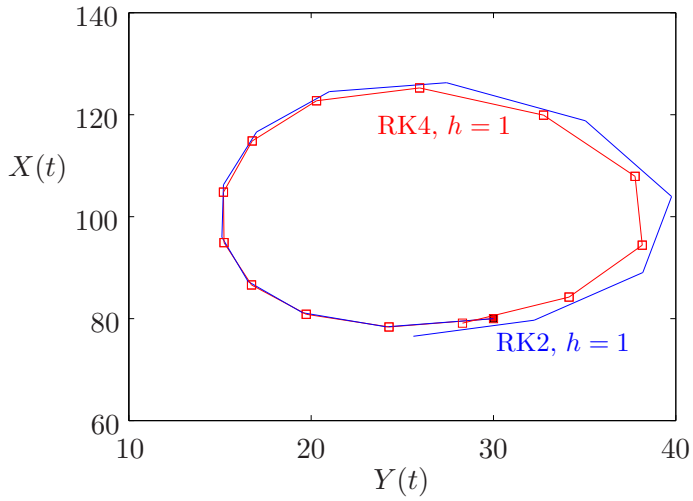


Abbildung 5.6.: Integration des Lotka-Volterra-Modells (5.285). Die Schrittweite beträgt $h = 1$. Gezeigt ist das Ergebnis des Runge-Kutta-Verfahrens vierter Ordnung (rot) im Vergleich zu demjenigen zweiter Ordnung (blau). Alle anderen Parameter sind identisch wie in Abb. 5.5.

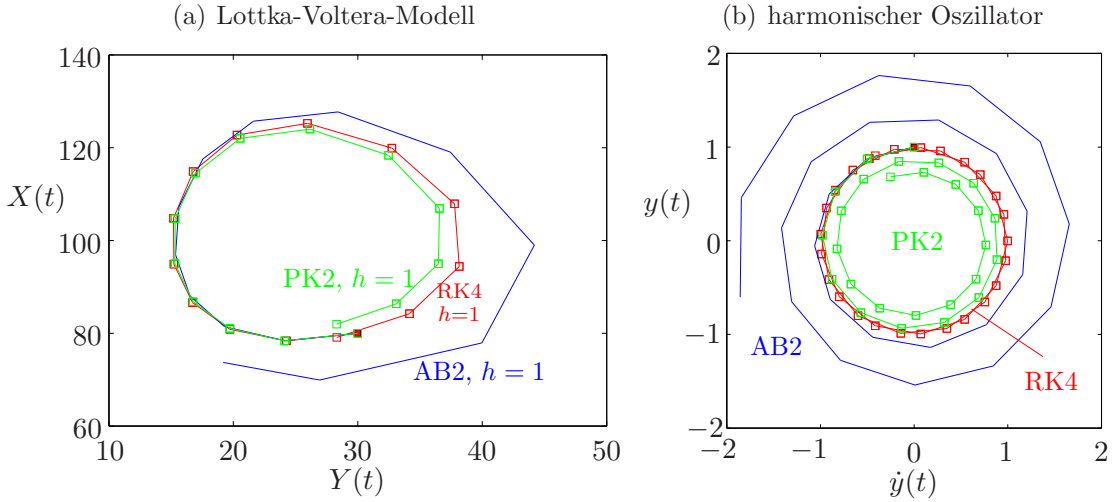


Abbildung 5.7.: Vergleich des Adams-Bashforth-Verfahrens zweiter Ordnung (AB2) und des Prädiktor-Korrektor-Verfahrens zweiter Ordnung (PK2) mit dem Runge-Kutta-Verfahren vierter Ordnung (a) für das Lotka-Volterra-Modell (5.285) mit $h = 1$ und Parametern wie in Abb. 5.5 und (b) für den ungedämpften harmonischen Oszillator $\ddot{y}+\omega^2 y = 0$ mit $h = 0.5$, $\omega = 1$ und Anfangsbedingungen $y(0) = 1$ und $\dot{y}(0) = 0$.

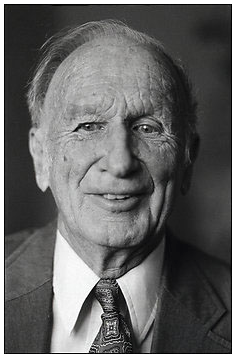
ve System eines harmonischen Oszillators (Abb. 5.7b) beobachtet werden.

Ein anderes berühmtes Modell stammt vom Meteorologen Lorenz (1963). Zur Beschreibung der Konvektion in der Atmosphäre schlug er ein Minimalmodell der thermischen Konvektion in einer ebenen Fluidschicht vor. Man kann dieses Modell aus den Navier-Stokes-Gleichungen erhalten, wenn man diese mittels *Galerkin-Methode*

auf einige wenige Fourier-Moden projiziert. Das Modell lautet

$$\dot{X} = \mathrm{Pr}\,(Y - X)\,, \tag{5.351a}$$
$$\dot{Y} = -Y + X\,(r - Z)\,, \tag{5.351b}$$
$$\dot{Z} = -bZ + XY. \tag{5.351c}$$

Die Strömung besteht aus parallelen Konvektionsrollen mit Amplitude $X(t)$. $Y(t)$ ist die zugehörige Amplitude des Temperaturfeldes und $Z(t)$ eine höhere Harmonische des Temperaturfelds, über welche die Moden X und Y miteinander gekoppelt sind. Der physikalisch relevante Parameterbereich liegt in der Nähe von $r = 1$ (reduzierte Rayleighzahl). Der Koeffizient b ist konstant.



Edward Norton Lorenz
1917–2008

Für $r < 1$ laufen alle Trajektorien in den stabilen trivialen *Fixpunkt* $X = Y = Z = 0$ ein. Dieser Zustand entspricht dem reinen Wärmeleitungszustand. Für $r > 1$ verzweigen hieraus zwei Fixpunkte, welche Konvektionsrollen mit gleicher konstanter Amplitude aber unterschiedlichem Drehsinn beschreiben. Bei weiterer Erhöhung von r werden die beiden stationären Lösungen instabil und es treten zwei stabile *Grenzzyklen* auf. Bei einer noch höheren Rayleighzahl r werden auch die beiden Grenzzyklen instabil und für die Parameter $\mathrm{Pr} = 10$, $b = 8/3$ und $r = 28$ findet man ein chaotisches Verhalten. Dabei nähern sich die Trajektorien dem *Lorenz-Attraktor*, einem *seltsamen Attraktor* in Gestalt einer *Cantor-Menge*.

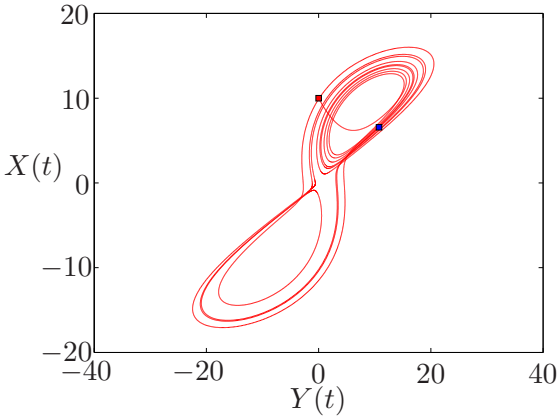
Beispielrechnungen sind in Abb. 5.8 gezeigt. Für die gegebenen Parameter und eine Rechnung bis $t = T = 10$ ist die Schrittweite $h = 0.01$ für das Runge-Kutta-Verfahrens vierter Ordnung (RK4) vollständig ausreichend, was ein Vergleich von Abb. 5.8a und b zeigt. Der Verlauf der Trajektorien und die jeweiligen Endpunkte stimmen optisch überein. Das Euler-Verfahren und auch das Prädiktor-Korrektor-Verfahren zweiter Ordnung (Abb. 5.8b,c) liefern für $h = 0.01$ jedoch völlig falsche Ergebnisse hinsichtlich der Trajektorie und der Endpunkte. Lediglich der Bereich, den die Trajektorie überstreichen, und die qualitative Struktur der Trajektorien sind ähnlich.

An dieser Stelle sei bemerkt, daß chaotische Trajektorien (wie in dem Beispiel) auch mit einem hypothetisch exakten Integrationsverfahren nur bis zu einem gewissen Zeitpunkt mit gegebener Fehlertoleranz vorhergesagt werden könnten, da dicht benachbarte Trajektorien exponentiell divergieren. Ab einem gewissen Zeitpunkt ist das Ergebnis dann unbrauchbar (völlig falsch). Man kann diese Grenze auch mit vierfach genauer Arithmetik (`real*16` anstelle von `real*8`) nur geringfügig verschieben.

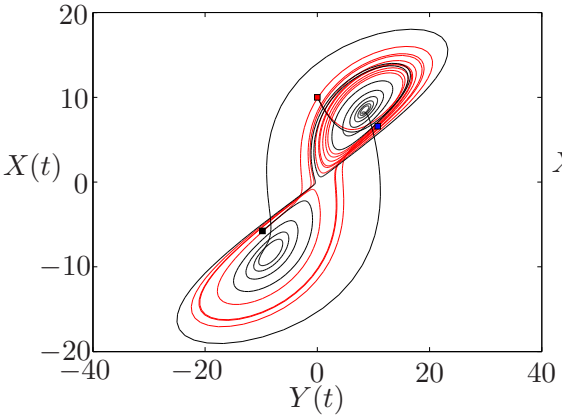
In MATLAB findet man verschiedene Löser für gewöhnliche Differentialgleichungssysteme. Ein Standard-Löser, der auf dem Runge-Kutta-Verfahren basiert, ist `[t,Y] = ode45(odefun,tspan,y0,options)` (Dormand-Prince-Methode[12]).

[12] Das Dormand-Prince-Verfahren ist adaptiv. Dabei wird ein Integrationsschritt mit einem RK4-

(a) RK4, $T = 10$, $h = 0.001$



(b) RK4 und Euler, $T = 10$, $h = 0.01$



(c) RK4 und PK2, $T = 10$, $h = 0.01$

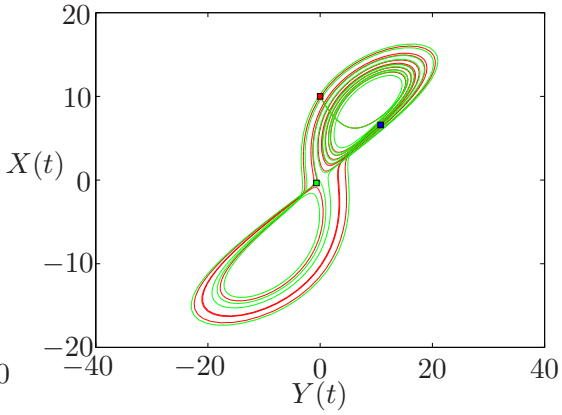


Abbildung 5.8.: Trajektorien der Lösung des Lorenzsystems (5.351) für die Parameter $b = 8/3$, $\mathrm{Pr} = 10$ und $r = 28$ mit der Anfangsbedingung $[X, Y, Z](0) = [10, 0, 20]$ (roter Punkt). Die Gleichungen wurden von $t = 0$ bis $t = T = 10$ integriert. (a) zeigt die Trajektorie des Runge-Kutta-Verfahrens vierter Ordnung (RK4, rot) für die sehr kleine Schrittweite $h = 0.001$. Sie ist optisch identisch mit den RK4-Trajektorien in (b) und (c), welche für die Schrittweite $h = 0.01$ berechnet wurden. Der blaue Punkt markiert den Endpunkt der RK4-Trajektorie. Im Vergleich zu RK4 ist in (b) die Lösung des Euler-Verfahrens gezeigt (schwarze Kurve, schwarzer Endpunkt) und (c) zeigt das Prädiktor-Korrektor-Verfahren im Vergleich zu RK4, ebenfalls für $h = 0.01$ (grüne Trajektorie, grüner Endpunkt).

Hierbei ist `t` ein Vektor der Zeitpunkte t_k, für den der Lösungsvektor ausgegeben wird, und `Y` ist ein Feld, bei dem jede Zeile einem Lösungsvektor $\vec{y}_k$ mit den Komponenten $y_{n,k}$ zum Zeitpunkt t_k repräsentiert. `odefun` ist die Funktion $\vec{f}[x, \vec{y}(x)]$, `tspan` ein mindestens zweikomponentiger Vektor mit der Zeitpunkten, an denen

und einem RK5-Verfahren berechnet. Aus dem Vergleich wird ein Fehler ermittelt. Ist der Fehler größer als die vorgegebene Fehlertoleranz, so wir die Zeitschrittweite solange verringert, bis die Fehlertoleranz eingehalten wird.

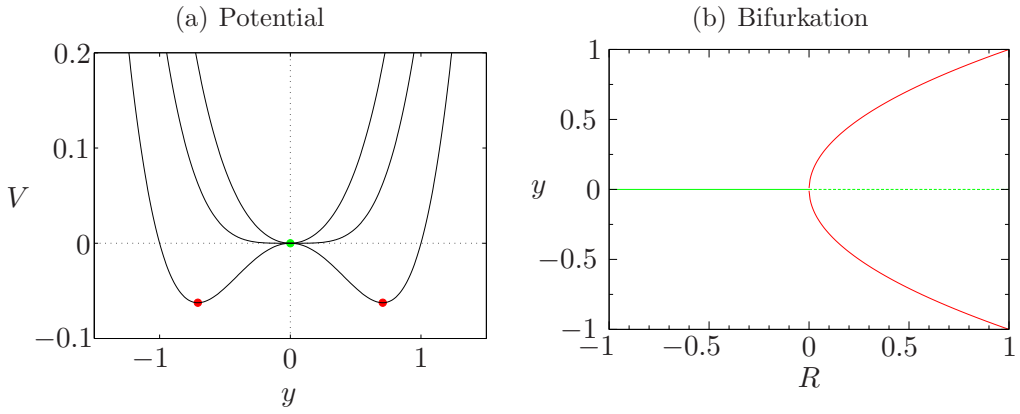


Abbildung 5.9.: (a) Potential und Gleichgewichtslagen für (5.352) für $R = -0.5$, 0 und 0.5. (b) Position der Gleichgewichtslagen als Funktion des Parameters R (Bifurkationsdiagramm).

die Lösung ausgegeben werden soll (mindestens Start- bzw. Endzeit) und `y0` der Vektor der Anfangsbedingungen. Beachte, daß die tatsächliche Zeitschrittweite variabel ist und von ode45 entsprechend der vorgegebenen Fehlertoleranzen (in `options`) adaptiv gewählt wird.

5.2. Stabilität

5.2.1. Physikalische Stabilität

In Natur und Technik treten Instabilitäten in mannigfaltiger Weise auf. Das Thema kann hier nur heuristisch gestreift werden.[13] Als einfaches Beispiel betrachten wir die überdämpfte Bewegung eines Teilchens in einem Doppelminimum-Potential (Abb. 5.9)

$$\dot{y} = -\frac{\partial V(y)}{\partial y} = -\frac{\partial}{\partial y}\left(-R\frac{y^2}{2} + \frac{y^4}{4}\right) = Ry - y^3. \tag{5.352}$$

Die Gleichgewichtslagen, für welche $\dot{y} = 0$ ist, kann man sofort angeben. Für $R < 0$ existiert nur eine einzige reelle Gleichgewichtslage, nämlich $y_0 = 0$. Sie ist stabil, da sie einem Minimum (dem einzigen) des Potentials entspricht. Bei Erhöhung des Parameters entstehen für $R > 0$ zwei neue Gleichgewichtslagen mit $y_{1,2} \neq 0$. Beide Lagen sind stabil gegenüber kleinen Störungen, da es sich um zwei lokale Minima von $V(y)$ handelt. Gleichzeitig wird die Gleichgewichtslage $y = y_0 = 0$ instabil, da $V(y = 0, R > 0)$ einem Maximum des Potentials entspricht.[14]

[13] Siehe aber die Lehrveranstaltung *Hydrodynamische Instabilitäten und Übergang zur Turbulenz*, LVA-Nr. 304.020.

[14] Aufgabe: Berechne die stationären Lösungen ($\dot{y} = 0$) von (5.352) für $R \geq 0$ mit Hilfe der Bogenlängenverfolgung beginnend mit $y^*(R = 1) = 1$.

Unabhängig von der Stabilität ist $y(t) = 0$ eine Lösung der gewöhnlichen Differentialgleichung (5.352) zum Anfangswert $y(0) = 0$. Um die Stabilität dieser Lösung zu untersuchen, betrachten wir die zeitliche Entwicklung kleiner Abweichungen $\epsilon(t)$ von der Lösung $y_0 = 0$. Wenn wir $y(t) = y_0 + \epsilon(t) = \epsilon(t)$ in (5.352) einsetzen und linearisieren[15] erhalten wir

$$\dot{y} = Ry - y^3 \qquad \overset{|y|=|\epsilon|\ll 1}{\Rightarrow} \qquad \dot{\epsilon} = R\epsilon \tag{5.353}$$

Die Lösung zum Startwert $\epsilon(t = 0) = \epsilon_0$ lautet

$$\epsilon(t) = \epsilon_0 \mathrm{e}^{Rt}. \tag{5.354}$$

Startet man also von einem kleinen Anfangswert $y(0) = \epsilon_0$, so wächst $|y(t)| \overset{y_0=0}{=} |\epsilon(t)|$ für $R > 0$ exponentiell mit der Zeit an. Man sagt: *Die Lösung y_0 ist linear instabil*. Für $R < 0$ ist die Lösung y_0 linear stabil.

Dieses Konzept kann man auf komplexere Systeme mit vielen Freiheitsgraden bzw. auf große Systeme von gewöhnlichen Differentialgleichungen übertragen. So existiert zum Beispiel im Lorenz-Modell die Gleichgewichtslage $(X, Y, Z) = (0, 0, 0)$. Für $r \leq 1$ ist sie linear stabil, für $r > 1$ ist sie instabil. Für $r > 1$ entstehen zwei neue nichttriviale stationäre Lösungen (Gleichgewichtslagen), deren Lage man exakt angeben kann (siehe z.B. Guckenheimer and Holmes, 1983). Abbildung 5.10 illustriert die Stabilität des Fixpunktes im Ursprung für $r < 1$ und dessen Instabilität für $r > 1$.

5.2.2. Stabilität numerischer Verfahren

Die *physikalische Instabilität* beschreibt ein (reales) physikalisches Phänomen. Die *numerische Instabilität* ist ein (in der Regel ungewolltes) Artefakt des numerischen Algorithmus und hat nichts mit dem physikalischen Verhalten zu tun.

Zur Motivation betrachten wir zunächst ein Beispiel für ein *numerisch* instabiles Verfahren. In Anlehnung an das Euler-Verfahren (5.294) erhält man bei einer zeitlich symmetrischen Diskretisierung der Ableitung y' aus dem Cauchy-Problem (5.288)

$$y_{n+1} = y_{n-1} + 2hf_n. \tag{5.355}$$

Dies ist ein Mehrschrittverfahren, da die Funktionswerte an den Stellen n und $n-1$ zur Berechnung von y_{n+1} benötigt werden. Das Verfahren ist zwar von zweiter Ordnung (symmetrische Bildung der Zeitableitung) und damit prinzipiell genauer als das Vorwärts-Euler-Verfahren, es ist aber instabil. Um das zu sehen, berechnen wir die Lösung $y(t)$ für das obige Beispiel (5.352) eines masselosen Teilchens im Potential mit Hilfe des symmetrischen Euler-Verfahrens (5.355). Abbildung 5.11 zeigt die Instabilität des symmetrischen Verfahrens (5.355) im Vergleich mit dem für diese Parameter ($h = 0.15$) stabilen Vorwärts-Euler-Verfahren und dem Runge-Kutta-Verfahren vierter Ordnung.[16]

[15] Für Terme ϵ^n mit $n \geq 2$ gilt für $\epsilon \to 0$: $\epsilon^n \ll \epsilon$. Sie können daher vernachlässigt werden, solange

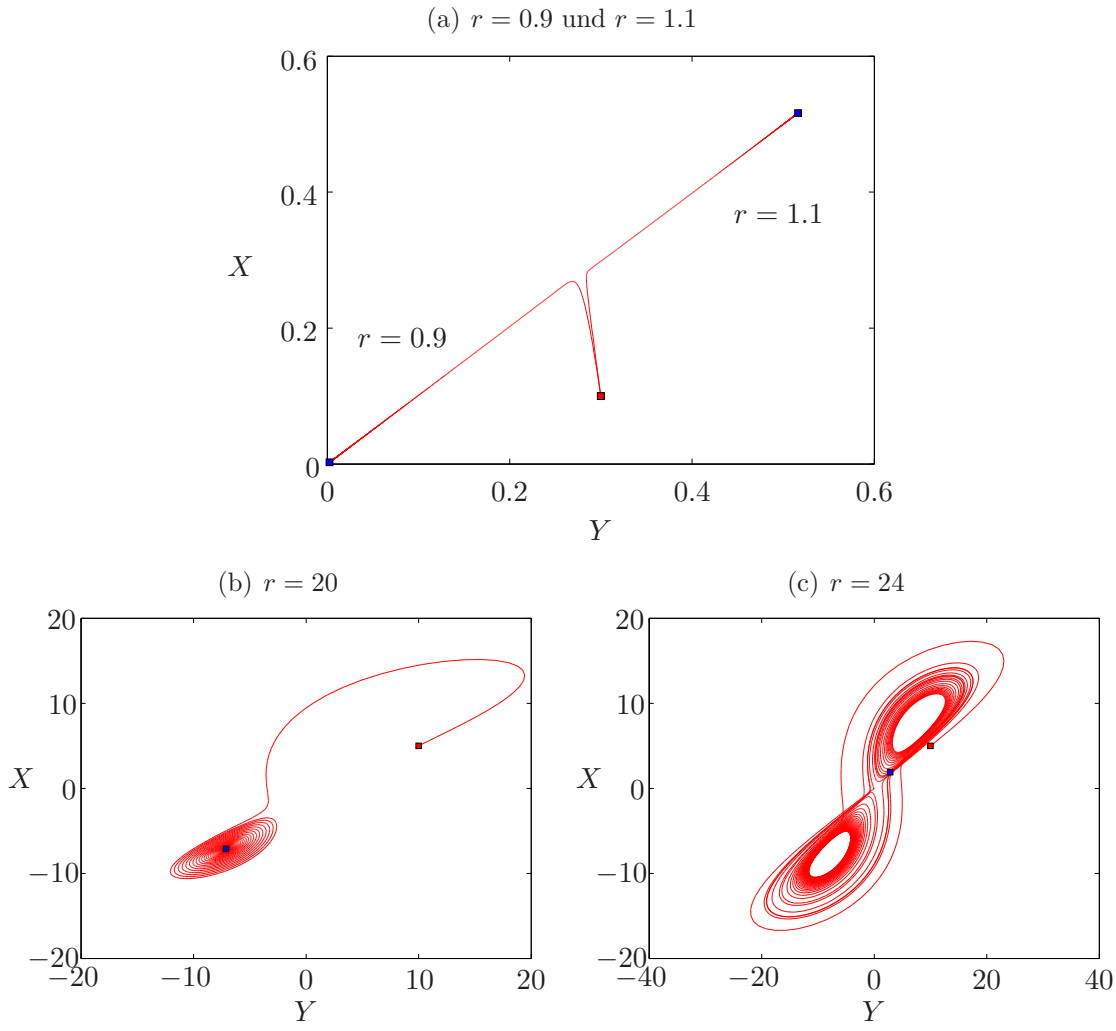


Abbildung 5.10.: (a) Trajektorien für das Lorenz-Modell mit Pr $= 10$ und $b = 8/3$ für die Anfangsbedingung $(X, Y, Z)(0) = (0.1, 0.3, 0.1)$ (rotes Quadrat). Für $r = 0.9$ besitzt das Systeme nur einen *stabilen* Fixpunkt bei $(X, Y, Z) = (0, 0, 0)$. Für $r = 1.1$ ist dieser Fixpunkt *instabil* und die Trajektorie konvergiert auf einen der beiden anderen nichttrivialen Fixpunkte. Die Berechnung erfolgte mit RK4 und $h = 0.01$ für die Zeitspanne $t \in [0, T]$. Die Lösung für $T = 50$ (blaue Quadrate) ist grapisch identisch mit der asymptotichen Lösung für $T \to \infty$. Abbildung (b) zeigt die Attraktion auf einen der beiden nichttrivialen Fixpunkte, die für $r = 20$ noch stabil sind. Für $r = 24$ (c) sind alle Fixpunkt instabil und das System oszilliert unregelmäßg und im Wechsel um jeweils einen der beiden Fixpunkte.

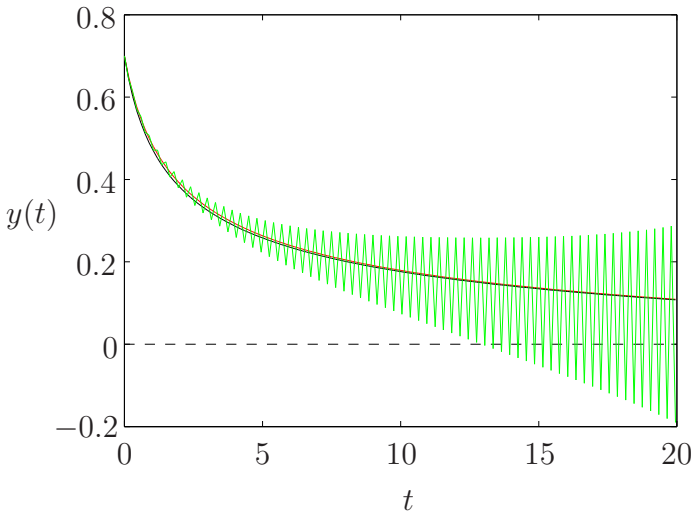
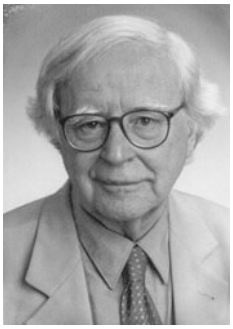


Abbildung 5.11.: Relaxation des Ortes eines masselosen Teilchens nach (5.352) zum stabilen Fixpunkt $y_0 = 0$ (gestrichelt) für $R = -0.03$. Die Anfangsbedingung ist $y(0) = 0.7$. Gezeigt sind die Lösungen für das stabile RK4-Verfahren (rot), das stabile Vorwärts-Euler-Verfahren (schwarz) und das instabile symmetrische Verfahren (5.355) (grün). Die Schrittweite für alle Rechnungen ist $h = 0.15$.



Germund Dahlquist 1925–2005

Die allgemeine Problematik der Stabilität numerischer Verfahren ist ein umfangreiches und schwieriges Gebiet. Außerdem kann man die Stabilität eines numerischen Verfahrens auf unterschiedliche Weisen definieren. Zur Charakterisierung numerischer Verfahren geht man deshalb zunächst von dem einfachen eindimensionalen skalaren Modell-Problem, der *Dahlquistschen Testgleichung*,[17] aus

$$y'(x) = \lambda y(x), \qquad x \geq 0, \tag{5.356}$$

wobei $\lambda \in \mathbb{R} < 0$ ist. Die exakte Lösung zur Anfangsbedingung $y(0) = 1$ lautet

$$y(x) = \mathrm{e}^{\lambda x}. \tag{5.357}$$

Wenn wir das Euler-Verfahren auf das Modell-Problem (5.356) anwenden, erhalten wir mit $y_0 = 1$

$$y_{k+1} = y_k + hf(x_k, y_k) = y_k + h\lambda y_k = (1 + \lambda h)\, y_k = (1 + \lambda h)^{k+1}. \tag{5.358}$$

$\epsilon(t) \ll 1$ ist.

[16] Für das einfache Doppelminimum-Potential (5.352) kann man auch die exakte Lösung $y_{\text{exakt}}(t)$ berechnen und dann den genauen Fehler $y_{\text{num}}(t) - y_{\text{exakt}}(t)$ betrachten.

[17] Die Dahlquistschen Testgleichung entspricht der überdämpften Bewegung eines Teilchens im Potential $V = -\lambda y^2/2$, $\lambda < 0$.

Für die exakte Lösung gilt $y(x \to \infty) = 0$. Damit auch für die numerische Folge $\lim_{k\to\infty} y_k = 0$ gilt, müssen wir fordern $|1 + \lambda h| < 1$ oder (mit $\lambda < 0$)

$$h < \frac{2}{|\lambda|}. \tag{5.359}$$

Ist diese Bedingung erfüllt, nennt man das Euler-Verfahren *stabil*. Ist sie nicht erfüllt, dann gilt $\lim_{k\to\infty} |y_k| = \infty$ und das Euler-Verfahren ist *instabil*. Daher definiert man:

> *Absolute Stabilität*: Ein Einschritt-Verfahren mit $\lambda h \neq 0$ wird absolut stabil genannt, wenn die Näherungslösung des eindimensionalen Modellproblems (5.356) für $\Re(\lambda) \leq 0$ beschränkt bleibt, d.h. $\sup_k |y_k| < \infty$.

Der Bereich in der komplexen Ebene $z = \lambda h \in \mathbb{C}$, für welchen ein Verfahren absolut stabil ist, nennt man den *Bereich absoluter Stabilität*.

Auskunft über die Stabilität des Modellproblems (5.356) gibt offenbar der *Verstärkungsfaktor*

$$G := \frac{y_{k+1}}{y_k}. \tag{5.360}$$

Für Stabilität muß gelten $|G| \leq 1$. Für das Euler-Verfahren ist der Verstärkungsfaktor $G(\lambda h) = 1+\lambda h$. Der Stabilitätsbereich ist dann durch $|G| = |1+\lambda h| = |1+z| \leq 1$ charakterisiert. Mit $|1 + z| = |1 + \Re(z) + \mathrm{i}\Im(z)| = \sqrt{[1 + \Re(z)]^2 + [\Im(z)]^2}$ ist die Bedingung $|1 + z| = 1$ ein Kreis mit Radius 1 um dem Punkt $z = (-1, 0)$ in der komplexen Ebene (Abb. 5.12a). Für Punkte $z = \lambda h$ innerhalb dieses Kreises ist das Vorwärts-Euler-Verfahren absolut stabil.[18]

In ähnlicher Weise kann man die Taylor- und Runge-Kutta-Formeln untersuchen. Mit $y' = f = \lambda y$ ist für das Modell-Problem (5.356) $f^{(r-1)}(x, y) = \lambda y^{(r-1)} = \lambda^r y^{(0)} = \lambda^r y$. Damit wird aus der Taylor-Formel (5.304)

$$y_{k+1} = y_k + h \sum_{r=1}^{R} \frac{h^{r-1}}{r!} f^{(r-1)}(x_k, y_k) \overset{(5.356)}{=} y_k + h \sum_{r=1}^{R} \frac{h^{r-1}}{r!} \lambda^r y_k = \left(1 + \sum_{r=1}^{R} \frac{(\lambda h)^r}{r!}\right) y_k. \tag{5.361}$$

Wir erhalten daraus den Verstärkungsfaktor

$$G = \sum_{r=0}^{R} \frac{(\lambda h)^r}{r!}. \tag{5.362}$$

Die Stabilitätsbereiche, charakterisiert durch $|G| = 1$, sind für die Ordnungen $R = 1$, 2, 3 und 4 in Abb. 5.13 dargestellt.

[18] Man betrachtet die linke komplexe Halbebene $z \in \mathbb{C}$ mit $\Re(z) \leq 0$, weil der Verstärkungsfaktor bei einigen Verfahren auch komplex sein kann.

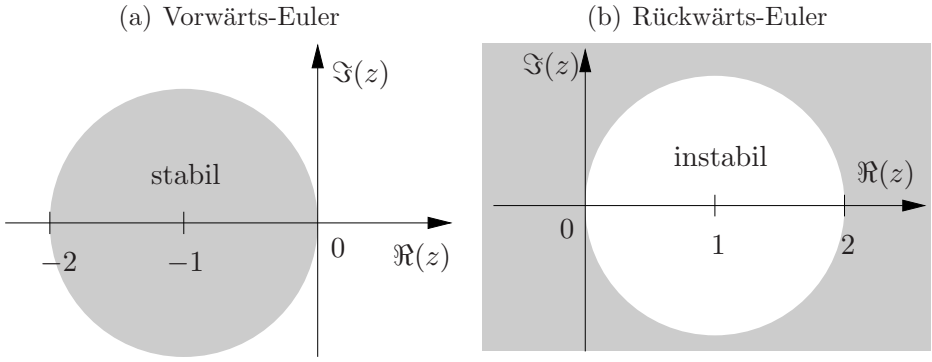


Abbildung 5.12.: Bereiche absoluter Stabilität (grau) und absoluter Instabilität (weiß) für das explizite Vorwärts-Euler-Verfahren (a) und für das implizite Rückwärts-Euler-Verfahren (b).

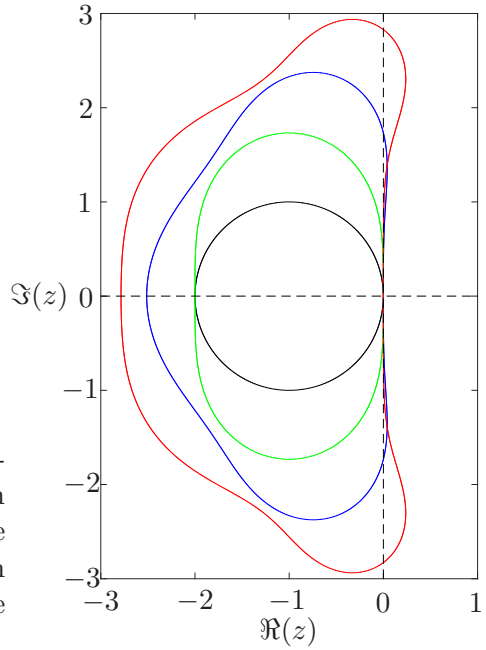


Abbildung 5.13.: Stabilitätsbereich der Taylor- und Runge-Kutta-Formeln für das Modellproblem (5.356) als Funktion von $z = \lambda h$. Der stabile Bereich liegt jeweils innerhalb der geschlossenen Kurven. Dargestellt sind die Stabilitätsbereiche von RK1 bis RK4 (von innen nach außen).

Da die Funktion $F(x,y) = \sum_{r=1}^{R} c_r K_r(x,y)$ in (5.311) des Runge-Kutta-Ansatzes bis zur Ordnung $R-1$ identisch ist mit der Taylor-Formel, sind die Taylor- und die Runge-Kutta-Formeln bis zu Ordnung R identisch und besitzen demzufolge auch dieselben Stabilitätsgrenzen.

Wenn die exakte Lösung sehr stark gedämpft ist ($\lambda \ll -1$) muß man die Schrittweite h hinreichend klein wählen, damit man den Stabilitätsbereich für $z = \lambda h$ nicht in Richtung der linken Halbebene verläßt. Daher wäre es wünschenswert, wenn sich der Stabilitätsbereich eines Verfahrens über die gesamte linke Halbebene erstrecken würde. Verfahren, die diese Eigenschaft besitzen, nennt man A-stabil (Dahlquist):

> Ein Verfahren ist *A-stabil*, wenn $\{z \in \mathbb{C} | \Re(z) < 0\} \subset \mathrm{S}$, wobei S der Stabilitätsbereich ist.

Man kann zeigen, daß explizite Methoden nie A-stabil sind. Implizite Methoden sind dagegen A-stabil. Ein Beispiel dafür ist das implizite Euler-Verfahren (5.298). Wendet man es auf das Modell-Problem (5.356) an, erhält man

$$y_{k+1} = y_k + h\lambda y_{k+1} \qquad \text{oder} \qquad y_{k+1} = \frac{y_k}{1-\lambda h} \tag{5.363}$$

Der Verstärkungsfaktor ist damit $G = (1-\lambda h)^{-1} = (1-z)^{-1}$ und für die Stabilitätsgrenze gilt $|1-z| = 1$. Dies ist ein Kreis mit Radius 1 um den Punkt $(1,0)$ in der komplexen z-Ebene (Abb. 5.12b), wobei der Bereich innerhalb des Kreises instabil ist und der gesamte Außenbereich stabil.

Ausgehend von dem Modell-Problem (5.356), kann man nun versuchen, die Ergebnisse auf allgemeinere Anfangswertaufgaben zu übertragen. Da diese Aufgabe nicht trivial ist, verzichten wir hier darauf. Die Stabilität eines Verfahrens ist jedoch sehr wichtig bei der Lösung sogenannter *steifer Probleme* (Kap. 5.2.4).

5.2.3. Wurzelbedingung

Die Stabilität linearer Mehrschrittverfahren läßt sich mit Hilfe einer Wurzelbedingung analysieren. Dazu beachten wir, daß man ein lineares Mehrschrittverfahren in der Form

$$y_{k+1} = \sum_{i=1}^{m} a_i y_{k+1-i} + h \sum_{i=0}^{m} b_i f_{k+1-i} \tag{5.364}$$

schreiben kann. Ist $b_0 = 0$ so ist das Verfahren explizit, andernfalls ist es implizit.

Zur Klassifizierung der Stabilität verwenden wir wieder das Modell-Problem (5.356). Wir gehen nun von einem beliebigen Satz von m aufeinander folgenden Werten aus und sammeln diese in einem Vektor $\vec{Y}_k = (y_k, y_{k-1}, \ldots, y_{k+1-m})^{\mathrm{T}}$ der Länge m. Mit $f_k = \lambda y_k$ (Modell-Problem) kann man das Mehrschrittverfahren (5.364) dann in der Form schreiben

$$\vec{Y}_{k+1} = \mathsf{M} \cdot \vec{Y}_k + h\vec{Z}_{k+1}(\vec{Y}_k, \vec{Y}_{k+1}), \tag{5.365}$$

wobei

$$\mathsf{M} = \begin{pmatrix} a_1 & a_2 & \dots & a_m \\ 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \vdots \\ & & 1 & 0 \end{pmatrix}. \tag{5.366}$$

Die erste untere Nebendiagonale mit den Einsen stellt nur die Identität der schon zuvor berechneten Funktionswerte sicher. Es findet nur eine Verschiebung des Indexes statt. Die erste Zeile von M generiert die neue erste Komponente y_{k+1} von $\vec{Y}_{k+1}$. Bei genügend kleiner Schrittweite $h \to 0$ bleibt der Betrag des Vektors $h\vec{Z}_{k+1}$ beschränkt.[19] Die Entwicklung der Lösung hängt dann nur von den Eigenwerten μ_i der Matrix M ab. Diese sind durch die Wurzeln des charakteristischen Polynoms

$$\begin{aligned} \rho(\mu) = \det(\mu\mathsf{I} - \mathsf{M}) &= \det \begin{pmatrix} \mu - a_1 & -a_2 & \dots & -a_m \\ -1 & \mu & & \\ & \ddots & \ddots & \\ & & -1 & \mu \end{pmatrix} \\ &= (\mu - a_1)\mu^{m-1} - a_2\mu^{m-2} \dots - a_{m-1}\mu - a_m \end{aligned} \tag{5.367}$$

gegeben. Hierbei wurde die Determinante nach der ersten Zeile der Matrix entwickelt. Damit die Folge (5.365) beschränkt bleibt, müssen alle Wurzeln μ_i des charakteristischen Polynoms (Eigenwerte von M), gegeben durch

$$\rho(\mu) = \mu^m - a_1\mu^{m-1} - a_2\mu^{m-2} - \dots - a_m \stackrel{!}{=} 0, \tag{5.368}$$

betragsmäßig keiner oder gleich eins sein. Dies ist das *Wurzel-Kriterium*

$$|\mu_i| \leq 1. \tag{5.369}$$

Für jedes Schema von mindestens erster Ordnung ist $\sum_{i=1}^{m} a_i = 1$. Damit ist $\mu = 1$ immer eine Wurzel von (5.368). Wenn die anderen $(m-1)$ Wurzeln die Bedingung $|\mu_i| < 1$ erfüllen, heißt das Verfahren *stark stabil*. Es heißt *stabil*, wenn für die anderen $(m-1)$ Wurzeln gilt $|\mu_i| \leq 1$, und wenn alle Wurzeln mit $|\mu_i| = 1$ einfach sind.

Für die Adams-Bashforth-Verfahren (explizit) der Ordnung m ist $a_1 = 1$ und $a_{i\geq 2} = 0$. Daher ist $\rho(\mu) = \mu^m - \mu^{m-1} = \mu^{m-1}(\mu - 1)$. Dann sind $(m-1)$ Wurzeln Null und eine Wurzel ist Eins. Die Adams-Bashforth-Verfahren sind also stark stabil.[20]

Beachte, daß die Stabilität nach dem Wurzelkriterium unter der Voraussetzung $h \to 0$ abgeleitet wurde. Deshalb können die Verfahren, die stabil im Sinne des Wurzel-Kriteriums sind, für hinreichend große Werte von h trotzdem instabil werden (siehe z.B. den Stabilitätsbereich der Runge-Kutta-Verfahren in Abb. 5.13, die

[19] Eine solche Abschätzung kann man auch für andere Funktionen $f(x, y)$ durchführen.

[20] Auch implizite Runge-Kutta-Verfahren höherer Ordnung können A-stabil sein. Das Stabilitätsgebiet der oben untersuchten expliziten Runge-Kutta-Verfahren ist aber immer beschränkt, genauso wie für alle linearen explizite Mehrschritt-Verfahren.

man auch mittels Wurzelkriterium auf Stabilität untersuchen kann). Für das symmetrische Verfahren (5.355) ist $m = 2$ mit $a_1 = 0$ und $a_2 = 1$ und wir erhalten aus (5.368) $\mu^2 - 1 = 0$ mit $\mu_{1,2} = \pm 1$. Das Verfahren ist also stabil, aber nicht stark stabil. Dies wirkt sich für endliche Schrittweite h katastrophal aus (Abb. 5.11). Auf der anderen Seite kann man die Schrittweite in der Praxis nicht beliebig verkleinern, da sich ansonsten Rundungsfehler negativ auswirken können und die Rechenzeit divergiert.

5.2.4. Steife Probleme

Ein Beispiel

Wenn sich die Lösung eines Anfangswertproblems auf sehr unterschiedlichen Skalen von x (entsprechend der Zeit) entwickelt, spricht man von einem *steifen System*. Dies ist zum Beispiel der Fall bei dem Problem

$$y' = -100y + 100, \qquad \text{mit} \quad y(0) = 2. \tag{5.370}$$

Die Lösung des homogenen Problems ist $y_{\text{hom}} = a\mathrm{e}^{-100x}$. Eine partikuläre Lösung ist $y_{\text{part}} = 1$. Daher lautet die Lösung ($a = 1$ wegen der Anfangsbedingung)

$$y(x) = 1 + \mathrm{e}^{-100x}. \tag{5.371}$$

Offenbar fällt y zunächst sehr stark ab ($y'(0) = -100$) und geht für $x \to \infty$ sehr schnell gegen 1. Schon bei $x = 0.1$ ist $y(0.1) = 1 + \mathrm{e}^{-10} = 1 + 4.5 \times 10^{-5}$. Danach ändert sich die Lösung kaum noch. Die entscheidende Größe ist die Änderungsrate y'. In diesem Beispiel ändert sie sich stark als Funktion von x.

Wenn wir (5.370) nun mit dem Vorwärts-Euler-Verfahren lösen wollten, erhielten wir

$$y_{k+1} = y_k + hf(y_k) = y_k + h(100 - 100y_k) = (1 - 100h)y_k + 100h. \tag{5.372}$$

Man kann leicht nachprüfen, daß die Lösung dieser Rekursionsformel zu dem Anfangswert $y_0 = 2$ gegeben ist durch

$$y_k = (1 - 100h)^k + 1. \tag{5.373}$$

Für $h > 0.01$ wird $(1 - 100h)$ negativ. Für $h = 0.02$ oszilliert die Folge der Funktionswerte schon zwischen 0 und 2. Für $h > 0.02$ wird dieser Effekt noch verstärkt. Das Verfahren ist dann instabil. Das Verhalten für $h = 0.019$ ist in Abb. 5.14 dargestellt (rote Punkte). Der Faktor $(1 - 100h)$ stellt eine Approximation von e^{-100x} dar (in jedem Schritt eine lineare Approximation von (5.371)). Sie ist allerdings nur gut für kleine Werte von h. Das Vorwärts-Euler-Verfahren versucht den Exponentialterm e^{-100x} zu approximieren, obwohl er für $x > 0.1$ praktisch nichts mehr zur Lösung beiträgt. Typischerweise werden bei steifen Gleichungen und Verwendung üblicher Methoden auch diejenigen Terme approximiert, welche das physikalische

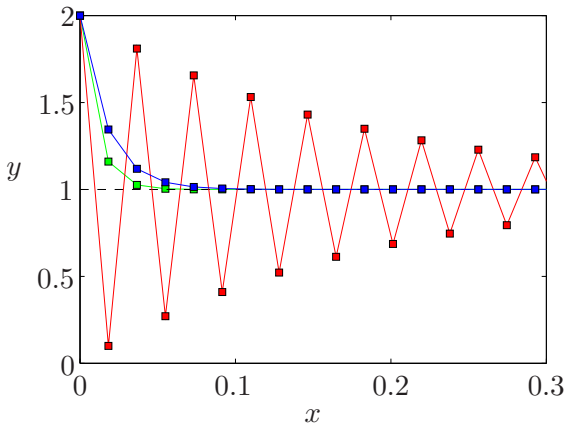


Abbildung 5.14.: Vergleich des expliziten Vorwärts-Euler- (rot) und des impliziten Rückwärts-Euler-Verfahrens (blau) mit der exakten Lösung (grün) von (5.370) für $h = 0.019$.

Verhalten stark stabilisieren, aber zur Lösung wenig beitragen. Dies führt dann zur Instabilität der Verfahren.

Um steife Anfangswertprobleme zu lösen, benötigt man besonders stabile Verfahren. Diese sollten nach Möglichkeit A-stabil sein. Da explizite Verfahren, wie das Vorwärts-Euler-Verfahren, nicht A-stabil sein können, kommen hierfür vor allem implizite Verfahren in Betracht.

Als Beispiel betrachten wir das Rückwärts-Euler-Verfahren (5.298), angewandt auf (5.370). Wir erhalten damit

$$y_{k+1} = y_k + hf(x_{k+1}, y_{k+1}) = y_k + h\left(-100y_{k+1} + 100\right). \tag{5.374}$$

Dies können wir wegen der einfachen Form von f nach y_{k+1} auflösen[21]

$$y_{k+1} = \frac{y_k + 100h}{1 + 100h}. \tag{5.375}$$

Die Lösung dieser Rekursionsformel zu der Anfangsbedingungen $y_0 = 2$ lautet

$$y_k = \frac{1}{(1 + 100h)^k} + 1. \tag{5.376}$$

Anders als die Lösung (5.373) für das Vorwärts-Euler-Verfahren zeigt die Lösung (5.376) für das Rückwärts-Euler-Verfahren keine Instabilität. Die Iterierte y_k nähert sich monoton fallend dem Grenzwert $y_k \to 1$. Man kann das unterschiedliche Verhalten auch dahingehend interpretieren, daß beim expliziten Vorwärts-Euler-Verfahren der Exponentialterm in (5.371) durch ein Polynom approximiert wird, welches für $x \to \infty$ nicht verschwinden kann. Beim impliziten Rückwärts-Euler-Verfahren wird

[21] Im allgemeinen Fall ist für jeden Integrationsschritt eine Gleichung (bzw. für Systeme ein Gleichungssystem) zu lösen, welche nichtlinear sein kann.

der Exponentialterm in (5.371) hingegen durch eine rationale Funktion approximiert, die für $x \to \infty$ verschwindet. Ein Vergleich der beiden Euler-Verfahren mit der exakten Lösung ist in Abb. 5.14 gezeigt. Das Rückwärts-Euler-Verfahren ist nur von erster Ordnung. Ein besseres implizites Verfahren wäre zum Beispiel das Adams-Moulton-Verfahren zweiter Ordnung (5.332).

Steifheit

Um ein Maß für die Steifheit zu finden, betrachten wir das lineare System

$$\vec{y}' = \mathsf{A} \cdot \vec{y} + \vec{b}(x), \tag{5.377}$$

mit einer konstanten $N \times N$ Matrix A. Man kann die allgemeine Lösung dieses Problems durch eine Superposition von Eigenvektoren $\vec{c}_n \in \mathbb{C}^N$ von A darstellen (Hauptachsentransformation)[22]

$$\vec{y}(x) = \sum_{n=1}^{N} a_n \vec{c}_n \mathrm{e}^{\lambda_n x} + \vec{g}(x), \tag{5.378}$$

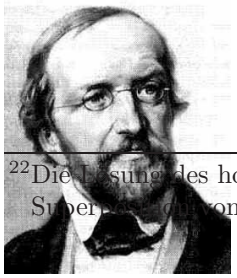
wobei $\lambda_n \in \mathbb{C}$ die (nicht-entartet angenommenen) Eigenwerte zu den Eigenvektoren $\vec{c}_n$ sind. Die Koeffizienten a_n sind Konstanten, die nur von den Anfangsbedingungen abhängen, und $\vec{g}(x)$ ist eine partikuläre Lösung. Unter der Annahme $\Re(\lambda_n) < 0$ tendiert dann jeder Term in der Summe (5.378) gegen Null: $\vec{c}_n \mathrm{e}^{\lambda_n x} \overset{x\to\infty}{\longrightarrow} 0$. Die Summe in (5.378) stellt damit den transienten Anteil der Lösung dar.

Je nach Absolutwert des Realteils $|\Re(\lambda_n)|$ gibt es transiente Terme, die schnell oder langsam zerfallen. Um die Disparität der Zerfallsraten zu quantifizieren definiert man die *Steifheit* als

$$\kappa := \frac{|\Re(\lambda)|_{\max}}{|\Re(\lambda)|_{\min}}. \tag{5.379}$$

Diese Definition kann man auf andere Systeme verallgemeinern, wenn man $\{\lambda_n\}$ als die Eigenwerte der Jacobi-Matrix $\partial f_i / \partial y_j$ auffaßt (Taylor-Entwicklung der rechten Seite von (5.335)).

5.3. Randwertprobleme



Peter Gustav Lejeune Dirichlet 1805–1859

Bisher (Kap. 5.1) hatten wir Anfangswertprobleme behandelt. Dabei werden Bedingung nur bei $x = 0$ vorgegeben. Bei Systemen von N gewöhnlichen Differentialgleichungen erster Ordnung oder bei einer gewöhnlichen Differentialgleichung der Ordnung N

[22] Die Lösung des homogenen Problems $\vec{y}' = \mathsf{A} \cdot \vec{y}$ ist $\vec{y} = \vec{c}\mathrm{e}^{\mathsf{A}x}$. Man stellt dann $\vec{c} = \sum_n a_n \vec{c}_n$ als Superposition von Eigenvektoren von A dar.

benötigt man dann auch N Anfangsbedingungen, damit die Lösung eindeutig festgelegt ist. Bei mehr als einer Bedingung können die Bedingungen im Prinzip auch bei verschiedenen Werten von x gegeben sein.

Wir betrachten die allgemeine gewöhnliche Differentialgleichung zweiter Ordnung

$$v''(x) = f\,[x, v(x), v'(x)] \tag{5.380}$$

auf dem Gebiet $x \in [a, b]$. Da die Gleichung von zweiter Ordnung ist, muß man für eine eindeutige Lösung zwei Bedingungen fordern. Es seien nun die Funktionswerte an zwei verschiedenen Stellen vorgegeben

$$v(a) = \alpha, \qquad \text{und} \qquad v(b) = \beta. \tag{5.381}$$

Diese Vorgabe von Funktionswerten bezeichnet man als *Dirichlet-Randbedingungen*. Gleichungen (5.380) und (5.381) definieren ein *Zweipunkt-Randwertproblem*. Es kann nichtlinear sein, wenn in der Funktion f beispielsweise Terme der Art vv' oder $(v')^3$ auftreten. Die einfacheren *linearen* Zweipunkt-Randwertprobleme lassen sich in der Form schreiben

$$v''(x) = B(x)v'(x) + C(x)v(x) + D(x), \qquad \text{auf} \quad a \le x \le b. \tag{5.382}$$

Hierbei sind $B(x)$, $C(x)$ und $D(x)$ fest vorgegebene Funktionen von x.[23]

5.3.1. Approximation mittels finiter Differenzen

Gleichungen ohne erste Ableitung ($B = 0$)

Eine Möglichkeit der numerischen Approximation des linearen Zweipunkt-Randwertproblems (5.382) besteht darin, die Differentialquotienten durch Differenzenquotienten zu ersetzen. Wir betrachten zunächst den einfachen Fall $B \equiv 0$[24]

$$v''(x) = C(x)v(x) + D(x), \qquad \text{auf} \quad a \le x \le b. \tag{5.383}$$

Das Gebiet $[a, b]$ wird nun in $N + 1$ kleine Intervalle aufgeteilt. Bei äquidistanter *Gitterweite* $h = (b - a)/(N + 1)$ sind dann die Gitterpunkte gegeben durch (Abb. 5.15)

$$x_n = a + nh, \qquad n = 0, \ldots, N + 1. \tag{5.384}$$

[23] Ein Beispiel, das in diese Klasse von Problemen fällt, ist die stationäre eindimensionale Wärmetransportgleichung $uT' = \kappa T''$ mit dem Wärmeleitungsterm T'', der Temperaturleitfähigkeit κ (Diffusivität) und dem konvektiven Temperaturtransport uT' (u: Strömungsgeschwindigkeit). Im stationären Gleichgewicht halten sich beide Terme die Waage. Für dieses Problem könnten Dirichlet-Randbedingungen näherungsweise mittels poröser durchströmter metallischer Wände (gute Wärmeleitung) realisiert werden.

[24] Diese Art Differentialgleichung tritt zum Beispiel auf bei der Bestimmung der Biegemomente $v(x)$ eines eingespannten Stabes unter axialer Druckkraft und homogener Querbelastung (Knickbiegung).

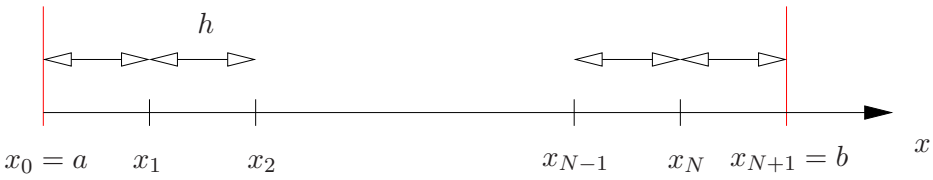


Abbildung 5.15.: Equidistantes Gitternetz für das Intervall $x \in [a, b]$ mit Gitterweite h.

Die ersten Ableitungen kann man an den Zwischengitterstellen mittels Differenzenquotienten folgendermaßen approximieren

$$v'\left(x_n - \frac{h}{2}\right) \approx \frac{v(x_n) - v(x_{n-1})}{h}, \quad \text{und} \quad v'\left(x_n + \frac{h}{2}\right) \approx \frac{v(x_{n+1}) - v(x_n)}{h}. \tag{5.385}$$

Daraus erhalten wir für die Approximation der zweiten Ableitung

$$\begin{aligned} v''(x_n) &\approx \frac{v'\left(x_n + \frac{h}{2}\right) - v'\left(x_n - \frac{h}{2}\right)}{h} \approx \frac{1}{h}\left[\frac{v(x_{n+1}) - v(x_n)}{h} - \frac{v(x_n) - v(x_{n-1})}{h}\right] \\ &= \frac{v(x_{n+1}) - 2v(x_n) + v(x_{n-1})}{h^2}. \end{aligned} \tag{5.386}$$

Wenn wir diese Approximation in (5.383) einsetzen, erhalten wir mit $v(x_n) = v_n$ für die *inneren* Gitterpunkte

$$\frac{1}{h^2}\left(v_{n+1} - 2v_n + v_{n-1}\right) = C_n v_n + D_n, \qquad \text{für} \quad n = 1, \ldots, N, \tag{5.387}$$

oder

$$-v_{n+1} + (2 + h^2 C_n)v_n - v_{n-1} = -h^2 D_n, \qquad \text{für} \quad n = 1, \ldots, N. \tag{5.388}$$

Diese Gleichungen für $n = 1, \ldots, N$ koppeln die unbekannten Funktionswerte an jeweils drei benachbarten Gitterpunkten. Die Gleichungen sind daher *implizit* und können nur gemeinsam, d.h. gekoppelt, gelöst werden. Die beiden *äußeren* Gitterpunkte sind schon durch die Randbedingungen $v_0 = \alpha$ und $v_{N+1} = \beta$ festgelegt. Wenn man die N Gleichungen für die N unbekannten Funktionswerte an den inneren Punkten in Matrixschreibweise notiert und die Randbedingungen beachtet, erhält man

$$\begin{pmatrix} 2 + h^2 C_1 & -1 & & & \\ -1 & 2 + h^2 C_2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 + h^2 C_{N-1} & -1 \\ & & & -1 & 2 + h^2 C_N \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_N \end{pmatrix} = \begin{pmatrix} \alpha - h^2 D_1 \\ -h^2 D_2 \\ \vdots \\ \vdots \\ \beta - h^2 D_N \end{pmatrix} \tag{5.389}$$

Dies ist ein *tridiagonales lineares Gleichungssystem*, das man leicht lösen kann (siehe Kap. 2.1.3). In dem Spezialfall, daß $C(x) = 0$ ist, erhält man die einfachere Matrix

$$\mathsf{A} = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \tag{5.390}$$

Bis auf den Faktor $-h^{-2}$ ist dies die Matrix-Darstellung des Operators der zweiten Ableitung, gebildet mit zentralen finiten Differenzen. Sie besitzt viele günstige Eigenschaften. Sie ist symmetrisch und auch positiv definit.[25] Für $C(x) \geq 0$ ist die Matrix A in (5.389) die Summe der positiv definiten Matrix (5.390) und einer diagonalen positiv semidefiniten Matrix $h^2 C_n \delta_{n,m}$. Man kann zeigen, daß die Summe, und damit die Matrix in (5.389), ebenfalls positiv definit und damit regulär ist. Deshalb besitzt (5.389) unter der Bedingung $C(x) \geq 0$ eine eindeutige Lösung.

Fehlerbetrachtungen

Bei der symmetrischen Diskretisierung (5.386) der zweiten Ableitung besitzt der lokale Diskretisierungsfehler die Größenordnung $O(h^2)$. Um das zu sehen, definieren wir den lokalen *Diskretisierungsfehler* der Differentialgleichung als

$$L(x_n, h) := \frac{1}{h^2}\left(\overline{v}_{n+1} - 2\overline{v}_n + \overline{v}_{n-1}\right) - C_n \overline{v}_n - D_n, \tag{5.391}$$

wobei $\overline{v}_n$ die exakte Lösung der Differentialgleichung (5.383) am Punkt x_n ist. Wegen (5.383) können wir $C_n \overline{v}_n + D_n$ durch $\overline{v}_n''$ ersetzen und erhalten

$$L(x_n, h) := \frac{1}{h^2}\left(\overline{v}_{n+1} - 2\overline{v}_n + \overline{v}_{n-1}\right) - \overline{v}_n''. \tag{5.392}$$

Dies bedeutet, daß sich der lokale Fehler nur durch die Diskretisierung der zweiten Ableitung ergibt. Wenn wir nun $\overline{v}_{n+1}$ und $\overline{v}_{n-1}$ in eine Taylorreihe um den Punkt x_n entwickeln, erhalten wir

$$\overline{v}_{n+1} = \overline{v}_n + h\overline{v}_n' + \frac{h^2}{2}\overline{v}_n'' + \frac{h^3}{6}\overline{v}_n''' + \frac{h^4}{24}\overline{v}_n'''' + O(h^5), \tag{5.393a}$$

$$\overline{v}_{n-1} = \overline{v}_n - h\overline{v}_n' + \frac{h^2}{2}\overline{v}_n'' - \frac{h^3}{6}\overline{v}_n''' + \frac{h^4}{24}\overline{v}_n'''' + O(h^5). \tag{5.393b}$$

Eingesetzt in (5.392) ergibt sich der lokale Fehler (die Terme $\sim \overline{v}_n$ und diejenigen mit ungeraden Ableitungen von $\overline{v}$ kompensieren sich)

$$L(x_n, h) = \frac{1}{h^2}\left[h^2\overline{v}_n'' + \frac{h^4}{12}\overline{v}_n'''' + O(h^6)\right] - \overline{v}_n'' = \frac{h^2}{12}\overline{v}_n'''' + O(h^4) = O(h^2). \tag{5.394}$$

[25] Man kann die Eigenwerte der Matrix explizit berechnen und zeigen, daß sie alle reell und positiv sind. Positiv definit bedeutet, $\vec{x}^{\mathrm{T}} \cdot \mathsf{A} \cdot \vec{x} > 0$ für alle $\vec{x} \neq 0$, $\vec{x} \in \mathbb{R}^N$ (siehe Anhang A.4.5).

Um nun eine Aussage über die tatsächlichen Abweichungen der *Funktionswerte* $e_n = \overline{v}_n - v_n$ zu erhalten, subtrahiert man (5.387) von (5.391) und erhält mit $\sigma_n := L(x_n, h)$ die Beziehung

$$e_{n+1} - (2 + h^2 C_n) e_n + e_{n-1} = h^2 \sigma_n, \qquad \text{für} \quad n = 1, \ldots, N. \tag{5.395}$$

Die Gleichung für den Fehler e_n ist also fast identisch mit der diskretisierten Differentialgleichung (5.388). Nur die rechten Seiten sind unterschiedlich. Um den *globalen Fehler* $e := \max_n |e_n|$ zu finden, müßte man dieses Gleichungssystem lösen. In Golub and Ortega (1996) wird gezeigt, daß sich der globale Fehler für $C(x) \geq \gamma > 0$ abschätzen läßt als

$$e \leq \frac{\sigma}{\gamma} = O(h^2), \tag{5.396}$$

wobei $\sigma := \max_n |\sigma_n|$ der maximale lokale Diskretisierungsfehler nach (5.391) ist. Jedenfalls skaliert der globale Fehler ebenfalls mit h^2 und kann durch Gitterverfeinerung systematisch reduziert werden.

Gleichungen mit erster Ableitung ($B \neq 0$)

Wenn neben der zweiten auch die erste Ableitung von v in die Differentialgleichung eingeht ($B(x) \neq 0$), müssen wir auch v' approximieren. Wenn wir *zentrale Differenzen* verwenden, ergibt sich die Näherung

$$v'(x) \approx \frac{v(x+h) - v(x-h)}{2h}. \tag{5.397}$$

Der Fehler, den man bei dieser symmetrischen Bildung der Differenz um den zentralen Punkt x herum macht, ist von der Größenordnung $O(h^2)$ und damit von der gleichen Größenordnung wie der Fehler, den wir bei der Diskretisierung von v'' mittels (5.386) machen. Wenn wir wie oben verfahren, erhalten wir für die Differentialgleichung (5.382) anstelle von (5.388) nun

$$\left(1 - \frac{hB_n}{2}\right) v_{n+1} - (2 + h^2 C_n) v_n + \left(1 + \frac{hB_n}{2}\right) v_{n-1} = h^2 D_n, \qquad \text{für} \quad n = 1, \ldots, N. \tag{5.398}$$

Mit den Abkürzungen

$$p_n = 2 + C_n h^2, \qquad q_n = -1 + \frac{B_n h}{2}, \qquad r_n = -1 - \frac{B_n h}{2} \tag{5.399}$$

läßt sich dieses Gleichungssystem auch in der Form schreiben

$$r_n v_{n-1} + p_n v_n + q_n v_{n+1} = -h^2 D_n. \tag{5.400}$$

In Matrixform erhalten wir mit $v_0 = \alpha$ und $v_{N+1} = \beta$ für die inneren Gitterpunkte

$$\begin{pmatrix} p_1 & q_1 & & & \\ r_2 & p_2 & q_2 & & \\ & \ddots & \ddots & \ddots & \\ & & r_{N-1} & p_{N-1} & q_{N-1} \\ & & & r_N & p_N \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \\ \vdots \\ \\ v_N \end{pmatrix} = -h^2 \begin{pmatrix} D_1 + r_1 \alpha / h^2 \\ D_2 \\ \vdots \\ D_{N-1} \\ D_N + q_N \beta / h^2 \end{pmatrix}. \tag{5.401}$$

Für die Lösung linearer Gleichungssysteme ist es generell sehr vorteilhaft, wenn die Matrix diagonaldominant ist. Andernfalls können unphysikalische Oszillationen der Näherungslösung auftreten. Diagonaldominanz erfordert (siehe (3.125))

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|. \tag{5.402}$$

In Falle der Matrix (5.401) bedeutet dies

$$|p_n| \geq |r_n| + |q_n|, \qquad \text{für} \quad n = 1, \ldots, N, \tag{5.403}$$

bzw.

$$\left|2 + C_n h^2\right| \geq \underbrace{\left|1 + \frac{B_n h}{2}\right| + \left|-1 + \frac{B_n h}{2}\right|}_{=2 \text{ für } |B_n h/2| \leq 1}, \qquad \text{für} \quad n = 1, \ldots, N. \tag{5.404}$$

Für

$$|B_n h| \leq 2, \qquad \text{für} \quad n = 1, \ldots, N. \tag{5.405}$$

ist die Matrix in (5.401) also diagonaldominant, wenn

$$\left|2 + C_n h^2\right| \geq 2, \qquad \text{für} \quad n = 1, \ldots, N. \tag{5.406}$$

Mit $C(x) \geq 0$ (s.o.) ist (5.406) für alle h erfüllt. Entscheidend ist jedoch offenbar die Bedingung (5.405).[26]

Die Bedingung (5.405) stellt eine Einschränkung der Gitterweite h dar. Man kann die Bedingung abschwächen, wenn man für die Diskretisierung der ersten Ableitung einseitige Differenzen verwendet. Dabei ist es wichtig, die Seite, zu welcher die Differenzen gebildet werden, davon abhängig zu machen, welches Vorzeichen $B(x_n)$ besitzt. Man diskretisiert daher

$$v'_n \approx \begin{cases} \dfrac{v_{n+1} - v_n}{h}, & \text{falls } B_n < 0, \\ \dfrac{v_n - v_{n-1}}{h}, & \text{falls } B_n > 0. \end{cases} \tag{5.407}$$

In der Strömungsmechanik wird dieses Verfahren auch *Upwind-Schema* genannt.[27]

[26] In der Strömungsmechanik wird $\max_n |B_n h| = \mathrm{Re_G} = |\vec{u}|h/\nu$ auch als Gitter-Reynoldszahl bezeichnet. Das rührt daher, daß der Koeffizient $B(x)$ in dem Term $B(x)v'$ dem Geschwindigkeitsbetrag $|\vec{u}|$ im konvektiven Term $\vec{u} \cdot \nabla\vec{u}$ der Navier-Stokes-Gleichung entspricht. Die kinematische Viskosität ν steht bei $\nabla^2\vec{u}$, während in (5.382) der entsprechende Faktor vor v'' gleich 1 ist. Damit entspricht $B_n h$ einer *Reynoldszahl* Re_G, die mit der Längenskala der Gitterweite h gebildet wird. Um Instabilitäten des numerischen Verfahrens zu vermeiden (dabei treten starke unphysikalische Oszillationen von Gitterpunkt zu Gitterpunkt auf), muß die Gitter-Reynoldszahl immer kleiner sein als 2. Siehe dazu auch das Skriptum zur Vorlesung *Numerische Methoden der Strömungsmechanik*, LVA-Nr. 302.042.

[27] In der Navier-Stokes-Gleichung entspricht B_n einer Geschwindigkeitskomponente. Beim Upwind-Schema wird die erste Ableitung dann abhängig von der Stromrichtung (Vorzeichen von B) unsymmetrisch entgegen dem Strom gebildet (in Luv-Richtung).

Bei Verwendung des Upwind-Schemas wird aus (5.399)

r_n	p_n	q_n	
-1	$2 + C_n h^2 - B_n h$	$-1 + B_n h$	falls $B_n \leq 0$,
$-(1 + B_n h)$	$2 + C_n h^2 + B_n h$	-1	falls $B_n \geq 0$.

(5.408)

Mit $C_n \geq 0$ kann man nun leicht die Diagonaldominanz der Matrix in (5.401) überprüfen. Sie ist nun unabhängig von der Gitterweite. Diese vorteilhafte Eigenschaft hat ihren Preis. Denn die einseitigen Differenzen des Upwind-Schemas sind nur von erster Ordnung $O(h)$ genau.

Neumann-Randbedingungen

John von Neumann 1903–1957

Bisher hatten wir Randbedingungen verwendet, bei denen die gesuchte Funktion am Rand vorgegeben war (Dirichlet-Randbedingungen). Oftmals ist jedoch anstelle des Funktionswertes am Rand die erste Ableitung vorgegeben

$$v'(a) = \alpha \qquad \text{und} \qquad v'(b) = \beta. \tag{5.409}$$

Diese Randbedingungen werden *Neumann-Randbedingungen* genannt. Gemische Randbedingungen

$$\eta_1 v(a) + \eta_2 v'(a) = \alpha \quad \text{und} \quad \xi_1 v(b) + \xi_2 v'(b) = \beta \tag{5.410}$$

werden manchmal auch als *Robin-Randbedingungen* bezeichnet.

Um zu sehen, wie man im Falle von Neumann-Randbedingungen (5.409) vorgeht, betrachten wir die einfache Differentialgleichung (5.383), d.h. $B(x) = 0$, die bei symmetrischer Diskretisierung der zweiten Ableitung auf das System (5.389) führte. Bei Neumann-Randbedingungen sind die Randwerte v_0 und v_{N+1} nicht a priori bekannt und müssen (im Gegensatz zu Dirichlet-Randbedingungen) als Variablen betrachtet werden. Deshalb muß auch die Differentialgleichung an den Punkten x_0 und x_{N+1} aufgestellt werden. Wenn man die zweite Ableitung von v am linken Rand x_0 symmetrisch bildet, erhält man

$$v_0'' \approx \frac{v_{-1} - 2v_0 + v_1}{h^2}. \tag{5.411}$$

Der hierbei auftauchende Wert v_{-1} am fiktiven Punkt x_{-1} kann nun mittels der diskretisierten Neumann-Randbedingung eliminiert werden. Dazu diskretisieren wir die Neumann-Randbedingung mittels zentraler Differenzen als

$$\alpha = v'(a) \approx \frac{v_1 - v_{-1}}{2h}. \tag{5.412}$$

Es folgt $v_{-1} = v_1 - 2\alpha h$. Damit können wir den unbekannten Wert v_{-1} aus (5.411) eliminieren und erhalten so

$$v_0'' \approx \frac{2v_1 - 2v_0 - 2\alpha h}{h^2}. \tag{5.413}$$

Dies ist der gesuchte Ausdruck für die zweite Ableitung im Punkt x_0. Falls man am rechten Rand Dirichlet-Randbedingung hat, ist man fertig. Man hat dann ein $(N+1)$-dimensionales Gleichungssystem zu lösen. Sind am rechten Rand auch Neumann-Randbedingungen vorgegeben, verfährt man analog wie am linken Rand und fügt einen externen Gitterpunkt bei x_{N+2} ein. Dann resultiert ein Gleichungssystem der Ordnung $N + 2$.

Wenn wir nun (5.413) für $n = 0$ und die entsprechende Gleichung für $n = N + 1$ verwenden, erhalten wir die diskretisierte Differentialgleichung analog zu (5.387) an den beiden Randpunkten in der Form

$$\frac{2v_1 - 2v_0 - 2\alpha h}{h^2} = C_0 v_0 + D_0, \tag{5.414a}$$

$$\frac{2\beta h - 2v_{N+1} + 2v_N}{h^2} = C_{N+1} v_{N+1} + D_{N+1}, \tag{5.414b}$$

oder

$$\left(2 + h^2 C_0\right) v_0 - 2v_1 = -2\alpha h - h^2 D_0, \tag{5.415a}$$

$$-2v_N + \left(2 + h^2 C_{N+1}\right) v_{N+1} = 2\beta h - h^2 D_{N+1}. \tag{5.415b}$$

damit erhält man das Gleichungssystem entsprechend zu (5.389)

$$\begin{pmatrix} 2 + h^2 C_0 & -2 & & & \\ -1 & 2 + h^2 C_1 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 + h^2 C_N & -1 \\ & & & -2 & 2 + h^2 C_{N+1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ \\ \vdots \\ \\ v_{N+1} \end{pmatrix} = \begin{pmatrix} -2\alpha h - h^2 D_0 \\ -h^2 D_1 \\ \vdots \\ -h^2 D_N \\ 2\beta h - h^2 D_{N+1} \end{pmatrix}. \tag{5.416}$$

Die bei Neumann-Randbedingungen auftretende Matrix ist nicht mehr symmetrisch wie diejenige in (5.389); man kann (5.416) aber auf eine symmetrische Form transformieren. Für $C_n > 0$ hat man Diagonaldominanz und die Matrix ist regulär und besitzt eine eindeutige Lösung. Im Falle $C \equiv 0$ ist die Matrix singulär, denn mit $\vec{e} = (1, \ldots, 1)^{\mathrm{T}}$ ist $\mathsf{A} \cdot \vec{e} = 0$. Damit ist die Differentialgleichung $v'' = D(x)$ mit Neumann-Randbedingungen nicht eindeutig lösbar. Hat man eine Lösung $V(x)$ gefunden, so erhält man beliebig viele andere Lösungen $V(x) + c$ durch Addition einer Konstanten c.

Periodische Randbedingungen

Periodische Randbedingungen

$$v(a) = v(b), \qquad \text{und} \qquad v'(a) = v'(b) \tag{5.417}$$

stellen einen weiteren Typus möglicher Randbedingungen dar. Sie treten bei Problemen mit Rotationssymmetrie oder bei zeitlich periodischen Vorgängen auf.

Bei periodischen Randbedingungen (5.417) ist $v_0 = v_{N+1}$ und auch alle Ableitungen von v an der Stelle $x = a$ sind identisch mit denjenigen bei $x = b$. Deshalb braucht der Punkt x_{N+1} nicht betrachtet zu werden. Wegen der periodischen Randbedingungen (5.417) kann die Funktion periodisch über das Intervall $[a, b]$ hinaus fortgesetzt werden. Dann ist $v_n = v_{n+k(N+1)}$ mit $k \in \mathbb{Z}$. Insbesondere ist $v_{-1} = v_N$. Damit können wir die zweite Ableitung am (Rand-)Punkt $n = 0$ approximieren als

$$v_0'' \approx \frac{v_N - 2v_0 + v_1}{h^2}. \tag{5.418}$$

Für den Punkt $n = N$ erhalten wir entsprechend

$$v_N'' \approx \frac{v_{N-1} - 2v_N + v_{N+1}}{h^2} = \frac{v_{N-1} - 2v_N + v_0}{h^2}. \tag{5.419}$$

Damit haben wir ein System von $N+1$ Unbekannten $v_0, \ldots, v_N$. Das (5.383) entsprechende System lautet damit

$$\begin{pmatrix} 2+h^2C_0 & -1 & & & -1 \\ -1 & 2+h^2C_1 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2+h^2C_{N-1} & -1 \\ -1 & & & -1 & 2+h^2C_N \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ \vdots \\ v_N \end{pmatrix} = \begin{pmatrix} -h^2D_0 \\ -h^2D_1 \\ \vdots \\ \vdots \\ -h^2D_N \end{pmatrix} \tag{5.420}$$

Die periodischen Randbedingungen zerstören die Tridiagonalität. Trotzdem ist die Matrix in (5.420) für $C_n > 0$ diagonaldominant, so daß sie regulär ist und eine eindeutige Lösung existiert. Ähnlich wie bei den Neumann-Randbedingungen ist die Matrix aber für $C(x) \equiv 0$ singulär und man kann weitere Lösungen ganz ähnlich durch Addition einer Konstanten zu einer bekannten Lösung konstruieren.

Nichtlinearitäten

Um die Behandlung von Nichtlinearitäten zu demonstrieren, nehmen zur Vereinfachung eine gewöhnliche nichtlineare Differentialgleichung zweiter Ordnung an

$$v''(x) = f\left[x, v(x)\right], \qquad x \in [a, b], \tag{5.421}$$

wobei die Funktion f unabhängig von der ersten Ableitung v' ist.[28] Die Verwendung zentraler Differenzen zweiter Ordnung führt auf die diskretisierte Gleichung

$$-v_{n-1} + 2v_n - v_{n+1} + h^2 f(x_n, v_n) = 0, \qquad n = 1, \ldots, N. \tag{5.422}$$

Am Rand mögen Dirichlet-Randbedingungen (5.381) vorgegeben sein, so daß $v_0 = \alpha$ und $v_{N+1} = \beta$. Wenn man die diskretisierte zweite Ableitung als Matrix-Operator A wie in (5.390) schreibt und den Rest der Gleichung in einem Vektor zusammenfaßt, erhält man das Problem in Matrix-Darstellung

$$\mathsf{A} \cdot \vec{v} + \vec{g}(\vec{v}) = 0. \tag{5.423}$$

[28] Ein Beispiel für ein einfaches nichtlineares Zweipunkt-Randwertproblem ist die Deformation eines dünnen elastischen Stabes (eingespannter Knickstab), der axial belastet wird. Die Differentialgleichung für den Winkel $\theta(s)$, den der Stab bezüglich der Achse (unbelastete gerade Ruhelage) bildet, lautet

$$\frac{\partial^2\theta}{\partial s^2} = -P\sin(\theta), \quad s \in [0, 1],$$

wobei s die normierte Bogenlänge ist und $P > 0$ die Stärke der axialen Last mißt. Bei eingespannten Enden lauten die Randbedingungen $\theta(0) = \theta(1) = 0$.

Hierbei ist A durch (5.390) gegeben und der Vektor $\vec{g}$ lautet

$$\vec{g}(\vec{v}) = h^2 \begin{pmatrix} f(x_1, v_1) \\ \vdots \\ f(x_N, v_N) \end{pmatrix} - \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix}. \tag{5.424}$$

Um das nichtlineare Gleichungssystem (5.423) für v_n zu lösen, muß man ein iteratives Verfahren verwenden. Dazu definieren wir die nichtlineare Vektorfunktion

$$\vec{F}(\vec{v}) := \mathsf{A} \cdot \vec{v} + \vec{g}(\vec{v}) \tag{5.425}$$

und suchen die Lösung des Nullstellenproblems $\vec{F}(\vec{v}) = 0$, welche man zum Beispiel mittels Newton-Iteration finden kann (Kap. 3.5.6). Zur Implementierung des Newton-Verfahrens benötigen wir die Jacobi-Matrix (3.174) von $\vec{F}(\vec{v})$. Hier lautet sie

$$\mathsf{J} = \frac{\partial F_i}{\partial v_j} = \frac{\partial}{\partial v_j}\left[\sum_k A_{ik} v_k + g_i(\vec{v})\right] = \sum_k A_{ik}\delta_{k,j} + \frac{\partial g_i(\vec{v})}{\partial v_j} = A_{ij} + \frac{\partial g_i(\vec{v})}{\partial v_j}. \tag{5.426}$$

In vorliegenden Fall hängt die n-te Komponente von $\vec{g}$ nur von v_n ab. Daher gilt

$$\frac{\partial g_i}{\partial v_j} = \frac{\partial g_i}{\partial v_i}\delta_{i,j} = h^2 \frac{\partial f(x_i, v_i)}{\partial v_i}\delta_{i,j}. \tag{5.427}$$

Die zugehörige Matrix ist also diagonal. Damit erhalten wir die Jacobi-Matrix

$$\mathsf{J} = A_{ij} + h^2 \frac{\partial f(x_i, v_i)}{\partial v_i}\delta_{i,j} = \tag{5.428}$$

$$\begin{pmatrix} 2 + h^2 \frac{\partial f(x_1,v_1)}{\partial v_1} & -1 & & & \\ -1 & 2 + h^2 \frac{\partial f(x_2,v_2)}{\partial v_2} & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 + h^2 \frac{\partial f(x_{N-1},v_{N-1})}{\partial v_{N-1}} & -1 \\ & & & -1 & 2 + h^2 \frac{\partial f(x_N,v_N)}{\partial v_N} \end{pmatrix}$$

Ausgehend von einer Anfangsschätzung $\vec{v}^{(0)}$ lautet das Newton-Verfahren dann wie auf S. 64

- Löse $\mathsf{J}^{(k)} \cdot \vec{\delta}^{(k)} = -\vec{F}^{(k)}$,
- Setze $\vec{v}^{(k+1)} = \vec{v}^{(k)} + \vec{\delta}^{(k)}$

In jeden Iterationsschritt ist also ein tridiagonales Gleichungssystem zu lösen. Beachte, daß für nichtlineare Probleme mehrere nicht-triviale Lösungen existieren können.[29]

[29] Aufgabe: Es sei $v''(x) = 3v(x) + x^2 + 10v^3(x)$. Diskretisiere das Problem und stelle die Jacobi-Matrix J auf. Ist J diagonaldominant?

5.3.2. Schießverfahren

Im vorangegangenen Kapitel wurde die Lösung von Randwertproblemen mittels Diskretisierung in finiten Differenzen und simultaner Lösung der resultierenden gekoppelten Gleichungen für die Unbekannten an den Gitterpunkten behandelt. Dies führt in der Regel auf große Gleichungssysteme. Eine alternative Möglichkeit der Lösung eines Randwertproblems besteht darin, von einem Randpunkt $x = a$ auszugehen und die Differentialgleichung bis zum Punkt $x = b$ wie bei einem Anfangswertproblem zu integrieren.

Um das Vorgehen zu demonstrieren, betrachten wir wieder das (i.a. nichtlineare) Randwertproblem zweiter Ordnung (5.380) mit Dirichlet-Randbedingungen (5.381). Wenn wir die Gleichungen zweiter Ordnung durch Integration in x-Richtung lösen wollen, benötigen wir neben der Anfangsbedingung $v(a) = \alpha$ jedoch noch eine weitere Anfangsbedingung. Dazu bietet sich die erste Ableitung an, die zunächst auf einen Schätzwert s gesetzt wird. Wir lösen also

$$v''(x) = f\left[x, v(x), v'(x)\right] \tag{5.429}$$

auf $x \in [a, b]$ zu den *Anfangsbedingungen*

$$v(a) = \alpha \qquad \text{und} \qquad v'(a) = s. \tag{5.430}$$

Hierbei ist s ein neuer freier Parameter, den wir beliebig variieren können. In der Regel wird die Lösung des Anfangswertproblems, welche wir mit einer ersten Schätzung $s = s_0$ erzielen, nicht die Randbedingung $v(b) = \beta$ erfüllen. Vielmehr ist $v(b; s_0) \neq \beta$. Die Idee ist nun, aus der Kenntnis der Abweichung $v(b; s_0)$ von β eine bessere Schätzung $v'(a) = s_1$ für die erste Ableitung im Punkt $x = a$ zu finden. Dieses Verfahren wird solange wiederholt, bis die Randbedingung bei $x = b$ hinreichend genau erfüllt ist: $v(b; s^*) = \beta$. Diese Vorgehensweise nennt man *Schießverfahren*.[30]

Das Problem besteht offenbar darin, die Nullstelle $g(s^*) = 0$ der Fehlerfunktion

$$g(s) := v(b; s) - \beta \tag{5.431}$$

zu finden. Zur Approximation der Nullstelle s^* von $g(s)$ können wir im Prinzip alle in Kap. 3.5 behandelten Verfahren zur Berechnung von Nullstellen verwenden. Zur Berechnung eines jeden Funktionswerts von $g(s)$ muß man allerdings jeweils ein Anfangswertproblem lösen. Da die Berechnung der Funktion $g(s)$ numerisch kostspielig sein kann, sollte man ein Verfahren verwenden, welches schnell konvergiert. In dieser Hinsicht ist das Sekanten-Verfahren (Kap. 3.5.2) dem Bisektionsverfahren (Kap. 3.5.3) überlegen. Zur Verwendung des Sekanten-Verfahrens benötigt man (ähnlich wie beim Newton-Verfahren) eine gute Schätzung von s^* (siehe Kap. 3.5.1), die zu Beginn der Rechnung meist nicht vorliegt. Daher könnte man mit einem robusten Verfahren (z.B. Bisektion) beginnen und nach einigen Iterationen zu einem Newton-Verfahren übergehen.

[30] Der Begriff *Schießverfahren* deutet auf den Zusammenhang mit der Ballistik hin.

5.3.3. Schießverfahren für lineare Gleichungen

Wenn die gewöhnliche Differentialgleichung des Randwertproblems linear ist, kann man die Lösung direkt und ohne Iteration erhalten. Dazu betrachten wir den allgemeinen Fall eines Systems von N linearen Differentialgleichungen erster Ordnung. Man kann diese in der Form

$$\vec{v}' = \mathsf{T}(x) \cdot \vec{v} + \vec{c}(x) \tag{5.432}$$

schreiben, mit den N linearen Randbedingungen

$$\mathsf{A} \cdot \vec{v}(a) + \mathsf{B} \cdot \vec{v}(b) + \vec{d} = 0. \tag{5.433}$$

Hierbei ist T eine x-abhängige $N \times N$-Matrix, die ebenso wie der Vektor $\vec{c}(x)$ stetig sei. A und B sind konstante $N \times N$-Matrizen und $\vec{d}$ ist ein konstanter Vektor. Sei nun $\vec{v}(x;\vec{s})$ die Lösung des Anfangswertproblems (5.432) zu der Anfangsbedingung

$$\vec{v}(a;\vec{s}) = \vec{s}, \tag{5.434}$$

wobei $\vec{s}$ zunächst unbestimmt ist.

Die Lösung des linearen Problems erster Ordnung (5.432) kann man explizit angeben. Sie lautet

$$\vec{v}(x;\vec{s}) = \vec{u}(x;\vec{s}) := \vec{v}(x;0) + \mathsf{Y}(x) \cdot \vec{s}, \tag{5.435a}$$

wobei die $N \times N$-Matrix $\mathsf{Y}(x)$ die Lösung des Anfangswertproblems

$$\mathsf{Y}'(x) = \mathsf{T}(x) \cdot \mathsf{Y}(x), \qquad \text{mit} \quad \mathsf{Y}(a) = \mathsf{I}. \tag{5.435b}$$

ist. Hiermit wird die Lösung von (5.432) auf die Berechnung von $\mathsf{Y}(x)$ verlagert.

Daß (5.435a) die Lösung von (5.432) zum (nicht spezifizierten) Anfangswert (5.434) ist, kann man leicht zeigen:

(a) Zunächst einmal wird der Anfangswert richtig reproduziert

$$\vec{u}(a;\vec{s}) \overset{(5.435a)}{=} \underbrace{\vec{v}(a;0)}_{(5.434)} + \underbrace{\mathsf{Y}(a) \cdot \vec{s}}_{(5.435b)} = 0 + \mathsf{I} \cdot \vec{s} = \vec{s}. \tag{5.436}$$

(b) Desweiteren wird das Differentialgleichungssystem (5.432) erfüllt, denn

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}\vec{u}(x;\vec{s}) &\overset{(5.435a)}{=} \underbrace{\vec{v}'(x;0)}_{(5.432)} + \underbrace{\mathsf{Y}'(x) \cdot \vec{s}}_{(5.435b)} = \overbrace{\mathsf{T}(x) \cdot \vec{v}(x;0) + \vec{c}(x)} + \mathsf{T} \cdot \mathsf{Y} \cdot \vec{s} \qquad (5.437) \\ &= \mathsf{T}(x) \cdot \underbrace{[\vec{v}(x;0) + \mathsf{Y} \cdot \vec{s}]}_{\vec{u}(x;s)} + \vec{c}(x) = \mathsf{T}(x) \cdot \vec{u}(x;s) + \vec{c}(x). \qquad \square \end{aligned}$$

Es ist die Frage, ob man $\vec{s}$ so wählen kann, daß auch die geforderten Randbedingungen (5.433) erfüllt werden. Dazu setzen wir die allgemeine Lösung (5.435a) in die Randbedingung (5.433) ein und erhalten

$$\begin{aligned} 0 \stackrel{!}{=} & \mathsf{A} \cdot \left[\underbrace{\vec{v}(a;0)}_{=0} + \underbrace{\mathsf{Y}(a) \cdot \vec{s}}_{=\vec{s}} \right] + \mathsf{B} \cdot \left[\vec{v}(b;0) + \mathsf{Y}(b) \cdot \vec{s} \right] + \vec{d} \\ = & \mathsf{A} \cdot \vec{s} + \mathsf{B} \cdot \left[\vec{v}(b;0) + \mathsf{Y}(b) \cdot \vec{s} \right] + \vec{d} \\ = & \left[\mathsf{A} + \mathsf{B} \cdot \mathsf{Y}(b) \right] \cdot \vec{s} + \mathsf{B} \cdot \vec{v}(b;0) + \vec{d}. \end{aligned} \tag{5.438}$$

Offenbar können diese N Randbedingungen erfüllt werden, wenn man $\vec{s} = \vec{s}^*$ wählt mit

$$\vec{s}^* = - \left[\mathsf{A} + \mathsf{B} \cdot \mathsf{Y}(b) \right]^{-1} \cdot \left[\mathsf{B} \cdot \vec{v}(b;0) + \vec{d} \right]. \tag{5.439}$$

Voraussetzung ist natürlich, daß $[\mathsf{A} + \mathsf{B} \cdot \mathsf{Y}(b)]$ invertierbar ist. Damit sind alle erforderlichen Bedingungen erfüllt.

Um s^* zu bestimmen, berechnet man zunächst die Matrix $\mathsf{Y}(b)$. Man erhält sie durch Lösen von (5.435b). Hierbei sind nur die Endwerte bei $x = b$ von Interesse. Gleichung (5.435b) entspricht N Vektorgleichungen der Form

$$\left[\vec{Y}_1'(x), \ldots, \vec{Y}_N'(x) \right] = \mathsf{T}(x) \cdot \left[\vec{Y}_1(x), \ldots, \vec{Y}_N(x) \right], \tag{5.440}$$

mit den Anfangswerten

$$\left[\vec{Y}_1(a), \ldots, \vec{Y}_N(a) \right] = \left[\vec{e}_1, \ldots, \vec{e}_N \right]. \tag{5.441}$$

Die Lösung von (5.440) liefert dann

$$\mathsf{Y}(b) = \left[\vec{Y}_1(b), \ldots, \vec{Y}_N(b) \right]. \tag{5.442}$$

Da die Anfangswerte $\vec{Y}_i(a)$ als orthogonale Einheitsvektoren eingehen, wird das Verhalten des Systems vollständig erfaßt. Den Vektor $\vec{v}(b;0)$ erhält man durch einmaliges Lösen des Anfangswertproblems (5.432) mit $\vec{s} = 0$. Insgesamt muß man also $N+1$ Anfangswertaufgaben lösen, um $\mathsf{Y}(b)$ und $\vec{v}(b;0)$ zu erhalten. Daraus ergibt sich dann s^* gemäß (5.439). Schließlich wird ein finaler (goldener) Schuß durch einmaliges Lösen von (5.432) mit $\vec{s} = \vec{s}^*$ gemacht.

6. Eigenwertprobleme

In Natur und Technik treten Eigenwertprobleme häufig dann auf, wenn kleine Abweichungen von einem Referenzzustand betrachtet werden. Der Referenzzustand kann ein mechanisches Gleichgewicht sein oder ein stationärer Strömungszustand. Als Beispiel betrachten wir die eindimensionale Bewegung zweier über Federn gekoppelter Punktmassen mit gleichen Massen $m = m_1 = m_2$ wie in Abb. 6.1.

Ausgehend von den beiden Gleichgewichtslagen y_1 und y_2 lauten die Bewegungsgleichungen für die beiden Punktmassen

$$m\ddot{x}_1 = K\,(x_2 - x_1) - Kx_1, \tag{6.443a}$$
$$m\ddot{x}_2 = K(x_1 - x_2), \tag{6.443b}$$

wobei x_1 und x_2 sehr kleine (infinitesimale) Auslenkungen aus den Gleichgewichtslagen sind und die beiden Federkonstanten $K = K_1 = K_2$ identisch angenommen wurden. In Matrixform lautet das Schwingungsproblem

$$m\ddot{\vec{x}} = K \begin{pmatrix} -2 & 1 \\ 1 & -1 \end{pmatrix} \cdot \vec{x}, \tag{6.444}$$

mit $\vec{x} = (x_1, x_2)^{\mathrm{T}}$. Für dieses lineare System können wir den Ansatz

$$\vec{x} = \vec{z}\,\mathrm{e}^{\mathrm{i}\omega t} + \text{c.c.} \tag{6.445}$$

machen. Er beschreibt harmonische Schwingungen mit Frequenz ω und (komplexen) Amplituden $\vec{z} = (z_1, z_2)^{\mathrm{T}}$. Eingesetzt erhalten wir

$$\underbrace{\frac{m\omega^2}{K}}_{:=\lambda}\vec{z} = \underbrace{\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}}_{=\mathsf{A}} \cdot \vec{z} \tag{6.446}$$

Dies ist ein Eigenwertproblem der Form $\mathsf{A} \cdot \vec{z} = \lambda\vec{z}$ mit $\lambda = m\omega^2/K$.

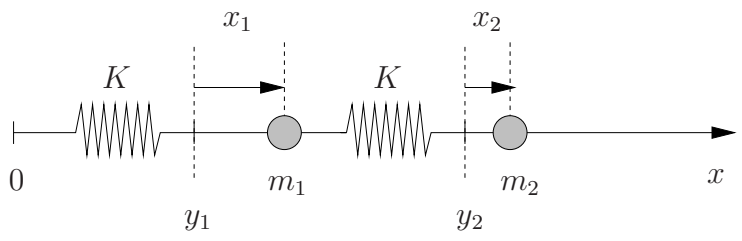


Abbildung 6.1.: Zwei gekoppelte Punktmassen.

Das resultierende 2×2-System besitzt eine nichttriviale Lösung genau dann, wenn (Lösbarkeitsbedingung)

$$\det(\lambda\mathsf{I}-\mathsf{A}) = \det\begin{vmatrix} \lambda-2 & 1 \\ 1 & \lambda-1 \end{vmatrix} = (\lambda-2)(\lambda-1)-1 = 0. \tag{6.447}$$

Diese Bedingung führt auf die *Eigenwerte* λ als Lösung der quadratischen Gleichung

$$\lambda^2-3\lambda+1=0 \quad \Rightarrow \quad \lambda^2-2\left(\frac{3}{2}\right)\lambda+\left(\frac{3}{2}\right)^2=\left(\frac{3}{2}\right)^2-1. \tag{6.448}$$

Aus den Eigenwerten

$$\lambda_{1,2}=\frac{3}{2}\pm\sqrt{\left(\frac{3}{2}\right)^2-1}=\begin{cases}0.382,\\ 2.618.\end{cases} \tag{6.449}$$

ergeben sich die Eigenfrequenzen $\omega_{1,2}$. Die Lösungen des Eigenwertproblems (Eigenvektoren) $\vec{z}^{(1)}$ und $\vec{z}^{(2)}$ sind nur bis auf einen Faktor bestimmt. Setzt man $z_1^{(i)}=1$, so ergibt sich $z_2^{(i)}$ aus (6.446). Die allgemeine Lösung (6.445) ist damit eine Superposition von *Eigenmoden* $\vec{z}^{(i)}\,\mathrm{e}^{\mathrm{i}\omega_i t}$

$$\vec{x}=a\vec{z}^{(1)}\,\mathrm{e}^{\mathrm{i}\omega_1 t}+b\vec{z}^{(2)}\,\mathrm{e}^{\mathrm{i}\omega_2 t}+\text{c.c.} \tag{6.450}$$

Die beiden komplexen Zahlen a und b müssen aus den vier Anfangsbedingungen (Positionen und Geschwindigkeiten der beiden Massen) bestimmt werden. Im Falle einer komplexen Eigenfrequenz würde die entsprechende *Mode* exponentiell anwachsen oder abklingen, was hier aber nicht der Fall ist, da (6.443) keinen Dämpfungsterm enthält.

6.1. Allgemeine Grundlagen

Die Gleichung

$$\mathsf{A}\cdot\vec{x}=\lambda\vec{x}. \tag{6.451}$$

bezeichnet man als *einfaches Eigenwertproblem*.[1] Elementare Definitionen und Eigenschaften derartiger Eigenwertprobleme sind in Kap. A.5 zu finden. Die Eigenvektoren $\vec{x}$ sind nur bis auf einen konstanten Faktor bestimmt. Die Eigenwerte $\lambda\in\mathbb{C}$ sind im allgemeinen komplex, auch wenn die Matrix $\mathsf{A}\in\mathbb{R}^{N\times N}$ reell ist.

[1] Das *verallgemeinerte Eigenwertproblem* lautet

$$\mathsf{A}\cdot\vec{x}=\lambda\mathsf{B}\vec{x},$$

wobei sowohl A wie auch B $N\times N$-Matrizen sind.

Die zu einer Matrix A *adjungierte Matrix* $\mathsf{B} = \mathsf{A}^\dagger$ ergibt sich durch Transponieren und Bilden des konjugiert Komplexen. Eine Matrix heißt *selbstadjungiert* (hermitesch konjugiert, siehe (A.592)), wenn gilt

$$\mathsf{A}^\dagger := (\mathsf{A}^*)^{\mathrm{T}} \stackrel{!}{=} \mathsf{A} \qquad \text{oder} \qquad A^\dagger_{ij} = A^*_{ji} \stackrel{!}{=} A_{ij}. \tag{6.452}$$

Bei einer selbstadjungierten Matrix sind je zwei bezüglich der Diagonalen symmetrisch gelegene Matrixelemente (Vertauschung der Indizes) konjugiert komplex zueinander.

Die Eigenwerte selbstadjungierter Matrizen sind reell. Um dies zu sehen, multiplizieren wir (6.451) von links skalar mit $\vec{x}^*$ (Zeilenvektor[2]) und erhalten

$$\vec{x}^* \cdot \mathsf{A} \cdot \vec{x} = \lambda\, \vec{x}^* \cdot \vec{x}. \tag{6.453}$$

Dann definiert man die skalare komplexe Funktion für *beliebige* Vektoren $\vec{x}$

$$R(\vec{x}) := \frac{\vec{x}^* \cdot \mathsf{A} \cdot \vec{x}}{\vec{x}^* \cdot \vec{x}}. \tag{6.454}$$

Sie wird *Rayleigh-Quotient* genannt. Wenn $\vec{x} = \vec{x}_j$ ein Eigenvektor von A ist, dann gilt $R(\vec{x}_j) = \lambda_j$, wobei λ_j der Eigenwert zu $\vec{x}_j$ ist. Wenn A nun selbstadjungiert ist, so gilt

$$\lambda = \frac{\vec{x}^* \cdot \mathsf{A} \cdot \vec{x}}{\vec{x}^* \cdot \vec{x}} = \frac{\vec{x} \cdot \mathsf{A}^{\mathrm{T}} \cdot \vec{x}^*}{\vec{x}^* \cdot \vec{x}} \overset{\text{selbstadj.}}{=} \frac{\vec{x} \cdot \mathsf{A}^* \cdot \vec{x}^*}{\vec{x}^* \cdot \vec{x}} = \left(\frac{\vec{x}^* \cdot \mathsf{A} \cdot \vec{x}}{\vec{x}^* \cdot \vec{x}} \right)^* = \lambda^*. \tag{6.455}$$

Der Rayleigh-Quotient (und damit auch jeder Eigenwert) einer selbstadjungierten Matrix ist reell.

6.1.1. Rayleigh-Quotient

John William Strutt
Lord Rayleigh
1842–1919

Wenn $\vec{x}$ eine Schätzung eines Eigenvektors $\vec{x}_j$ ist, dann ist der Rayleigh-Quotient $R(\vec{x})$ eines Schätzung des zugehörigen Eigenwerts λ_j. Hierin liegt die Bedeutung des Rayleigh-Quotienten.

Um zu untersuchen, wie die Abweichung $R(\vec{x}) - \lambda_j$ vom Eigenwert λ_j von der Abweichung $\vec{x} - \vec{x}_j$ vom Eigenvektor $\vec{x}_j$ abhängt, kann man $R(\vec{x})$ in eine Taylor-Reihe um $\vec{x}_j$ entwickeln

$$R(\vec{x}) = \underbrace{R(\vec{x}_j)}_{\lambda_j} + (\vec{x} - \vec{x}_j) \cdot \nabla R(\vec{x}_j) + O\left(\|\vec{x} - \vec{x}_j\|^2\right). \tag{6.456}$$

[2] Der hochgestellt Index T wird weggelassen. Man sieht an der Position von $\vec{x}$ bzgl. A, ob es sich um einen Zeilen- oder einen Spaltenvektor handelt. Es bedeuten: $\vec{a} \cdot \vec{b} = \vec{a}^{\mathrm{T}} \vec{b}$ und $\vec{c} \cdot \mathsf{A} = \vec{c}^{\mathrm{T}} \mathsf{A}$.

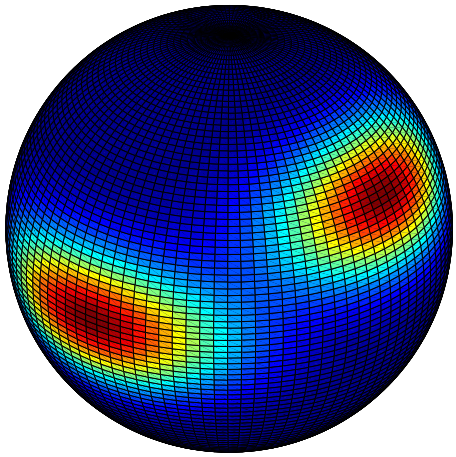


Abbildung 6.2.: Prinzip-Skizze für $N = 3$ der Abweichung des Rayleigh-Quotienten vom Eigenwert $R(\vec{x}) - \lambda_j$. Die drei auf eins normierten Eigenvektoren zeigen auf drei Punkte der Einheitskugel. Die Abweichung des Rayleigh-Quotienten für einen Einheitsvektor, der auf einen beliebigen Punkt der Kugel zeigt, ist in Farbe dargestellt. Dort, wo die Abweichung Null ist ($\vec{x} = \vec{x}_j$), verschwindet die Abweichung (dunkelrot).

Wenn A selbstadjungiert ist (reelle Eigenwerte), kann man leicht zeigen, daß $\nabla R(\vec{x}_j) = 0$ ist.[3] Dann gilt

$$R(\vec{x}) - \lambda_j = O\left(\|\vec{x} - \vec{x}_j\|^2\right). \tag{6.457}$$

Das heißt, der Rayleigh-Quotient besitzt für alle Eigenvektoren ein quadratisches Minimum mit dem Funktionswert λ_j. Die Abweichung des Rayleigh-Quotienten vom Eigenwert $\lambda_j = R(\vec{x}_j)$ ist in der Umgebung der Eigenvektoren quadratisch klein. Damit ist der Rayleigh-Quotient eine quadratisch genaue Schätzung des Eigenwerts.

Da die Eigenvektoren nur bis auf einen Faktor bestimmt sind, kann man die Betrachtung auf Vektoren $\|\vec{x}\| = 1$ der Norm 1 beschränken. Für reelle symmetrische Matrizen[4] ist $R(\vec{x})$ damit auf der Oberfläche der N-dimensionalen Einheitskugel im $\mathbb{R}^N$ definiert. $R(\vec{x})$ besitzt auf dieser Oberfläche quadratische Extrema bei $\vec{x}_j$. Dies ist in Abb. 6.2 schematisch dargestellt.

6.1.2. Diagonalisierung

Im folgenden geben wir die Beschränkung auf reelle symmetrische Matrizen wieder auf. Die *Eigenwert-Zerlegung* einer $N \times N$-Matrix A hat die Form

$$\mathsf{A} = \mathsf{X} \cdot \Lambda \cdot \mathsf{X}^{-1}, \tag{6.458}$$

[3] Um den Gradienten von $R(\vec{x})$ zu berechnen, beschränken wir uns auf reelle und symmetrische Matrizen ($\mathsf{A} = \mathsf{A}^\dagger$). Dann sind auch die Eigenwerte reell und wir erhalten mit $\vec{x} \in \mathbb{R}^N$ (beachte: $\nabla\vec{x} = \mathsf{I}$)

$$\nabla R(\vec{x}) = \frac{\nabla(\vec{x}\cdot\mathsf{A}\cdot\vec{x})}{\vec{x}\cdot\vec{x}} - \frac{(\vec{x}\cdot\mathsf{A}\cdot\vec{x})\,\nabla(\vec{x}\cdot\vec{x})}{(\vec{x}\cdot\vec{x})^2} = \frac{\mathsf{A}\cdot\vec{x} + \vec{x}\cdot\mathsf{A}}{\vec{x}\cdot\vec{x}} - \frac{(\vec{x}\cdot\mathsf{A}\cdot\vec{x})\,2\vec{x}}{(\vec{x}\cdot\vec{x})^2}$$
$$= \frac{2}{\vec{x}\cdot\vec{x}}\left[\frac{1}{2}(\mathsf{A}\cdot\vec{x} + \vec{x}\cdot\mathsf{A}) - R(\vec{x})\vec{x}\right] \overset{\mathsf{A}\text{ symm.}}{=} \frac{2}{\vec{x}\cdot\vec{x}}\left[\mathsf{A}\cdot\vec{x} - R(\vec{x})\vec{x}\right] \overset{\vec{x}=\vec{x}_j}{=} 0.$$

[4] Dann sind die Eigenwerte und Eigenvektoren sowie $R(\vec{x})$ reell; siehe Kap. A.5.

wobei Λ eine Diagonalmatrix ist, deren Diagonale mit allen Eigenwerten von A besetzt ist. Eine derartige Zerlegung existiert nicht immer (siehe Kap. 6.1.3). Wenn sie existiert, kann man die Zerlegung auch schreiben als

$$\mathsf{A} \cdot \mathsf{X} = \mathsf{X} \cdot \Lambda, \tag{6.459}$$

oder

$$\mathsf{A} \cdot \left[\begin{array}{c|c|c} & & \\ \vec{x}_1 & \dots & \vec{x}_N \\ & & \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ \vec{x}_1 & \dots & \vec{x}_N \\ & & \end{array}\right] \cdot \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix}, \tag{6.460}$$

wobei $\vec{x}_k$ hier die Spaltenvektoren der Matrix X sind. In Komponentenschreibweise lautet die Zerlegung (Summenkonvention)

$$A_{ij} X_{jk} = X_{jk} \delta_{ki} \lambda_i. \tag{6.461}$$

Wenn wir die k-te Spalte von (6.460) betrachten, erhalten wir

$$\mathsf{A} \cdot \vec{x}_k = \lambda_k \vec{x}_k. \tag{6.462}$$

Der k-te Spaltenvektor von X ist also genau der Eigenvektor $\vec{x}_k$ von A zum Eigenwert λ_k.

Die Eigenwertzerlegung (6.458) ist äquivalent zu einer Transformation auf *Eigenvektor-Koordinaten*. Denn wenn wir zum Beispiel von dem linearen Problem

$$\mathsf{A} \cdot \vec{x} = \vec{b} \tag{6.463}$$

ausgehen und für A die Zerlegung (6.458) verwenden, erhalten wir

$$\mathsf{X} \cdot \Lambda \cdot \mathsf{X}^{-1} \cdot \vec{x} = \vec{b} \qquad \Rightarrow \qquad \Lambda \cdot \underbrace{\mathsf{X}^{-1} \cdot \vec{x}}_{\tilde{\vec{x}}} = \underbrace{\mathsf{X}^{-1} \cdot \vec{b}}_{\tilde{\vec{b}}} \tag{6.464}$$

Mit der Definition von $\tilde{\vec{x}} = \mathsf{X}^{-1} \cdot \vec{x}$ und $\tilde{\vec{b}} = \mathsf{X}^{-1} \cdot \vec{b}$ erhalten wir das transformierte System

$$\Lambda \cdot \tilde{\vec{x}} = \tilde{\vec{b}}. \tag{6.465}$$

Wenn wir ein lineares Problem in der Basis der Eigenvektoren darstellen, dann ist die transformierte Matrix Λ diagonal. Dies wird durch die Transformation $\vec{x} \to \tilde{\vec{x}} = \mathsf{X}^{-1} \cdot \vec{x}$ bewirkt; man nennt sie auch *Hauptachsentransformation*. Da die Spaltenvektoren von X aus den Eigenvektoren bestehen, transformiert $\vec{x} = \mathsf{X} \cdot \tilde{\vec{x}}$ die Einheitsvektoren in Raum $\tilde{\vec{x}}$ gerade auf die Eigenvektoren: $\vec{x}_k = \mathsf{X} \cdot \tilde{\vec{e}}_k$.

6.1.3. Entartung von Eigenwerten und Eigenvektoren

Die Menge aller Eigenwerte $\{\lambda_j\}$ einer Matrix A bezeichnet man als *Spektrum* von A. Die Eigenwerte λ können *entartet* sein. Dies bedeutet, daß sie mehrfach vorkommen.

Aus der Eigenwertgleichung (6.451) folgt

$$(\lambda\mathsf{I} - \mathsf{A}) \cdot \vec{x} = 0. \tag{6.466}$$

Damit dieses homogene Gleichungssystem eine nichttriviale Lösungen $\vec{x}$ (Eigenvektoren) besitzt, muß die Determinante der Koeffizientenmatrix verschwinden. Diese Bedingung kann man im Prinzip zur Bestimmung der Eigenwerte verwenden. Sie ist gleichbedeutend damit, daß das *charakteristische Polynom* der $N \times N$-Matrix A (komplex)

$$p_A(z) := \det(z\mathsf{I} - \mathsf{A}), \tag{6.467}$$

Nullstellen bei $z = \lambda_i \in \mathbb{C}$ hat. Das charakteristische Polynom der Ordnung N können wir schreiben als

$$p_A(z) = (z - \lambda_1)(z - \lambda_2)\ldots(z - \lambda_N), \qquad \lambda_j \in \mathbb{C}. \tag{6.468}$$

Selbst wenn die Matrix A reell ist, können die Wurzeln des charakteristischen Polynoms, also die Eigenwerte, komplex sein. Sie treten dann aber immer als konjugiert komplexe Paare auf.

In (6.468) kann ein Faktor $(z - \lambda_i)$ auch mehrfach vorkommen. Dann ist der Eigenwert λ_i entartet. Die Anzahl identischer Faktoren $(z - \lambda_i)$ bezeichnet man als *algebraische Multiplizität* von λ_i. Wenn ein Faktor nur einfach auftritt, bezeichnet man den zugehörigen Eigenwert als *einfachen Eigenwert* (die algebraische Multiplizität ist dann 1). Jede $N \times N$-Matrix besitzt mindestens einen Eigenwert.

Nun kann es sein, daß alle Eigenvektoren eines entarteten Eigenwerts verschieden (linear unabhängig) sind, oder daß einige von ihnen gleich (linear abhängig) sind. Die Anzahl der verschiedenen Eigenvektoren zu einem entarteten Eigenwert nennt man *geometrische Multiplizität*. Es ist dann klar, daß die geometrische Multiplizität immer kleiner oder gleich der algebraischen Multiplizität ist.

Die Eigenvektoren zu ein und demselben (multiplen) Eigenwert λ bilden einen sogenannten *invarianten Unterraum* E_λ des Raumes, der von allen Eigenvektoren aufgespannt wird. Denn die Operation $\mathsf{A} \cdot \vec{x}$ führt nicht aus dem Unterraum E_λ heraus, der zu λ gehört: $\mathsf{A} \cdot \vec{x} \subseteq E_\lambda$. Die Dimension des Unterraums E_λ ist gleich der Anzahl der linear unabhängigen Eigenvektoren zu einem festen Eigenwert, also gleich der geometrischen Multiplizität.

Meistens ist die geometrische und auch die algebraische Multiplizität gleich 1. Es gibt aber Ausnahmen. Als Beispiele betrachte

$$\mathsf{A} = \begin{pmatrix} 2 & & \\ & 2 & \\ & & 2 \end{pmatrix} \qquad \text{und} \qquad \mathsf{B} = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{pmatrix}. \tag{6.469}$$

Beide Matrizen besitzen das charakteristische Polynom $p_A = p_B = (z-2)^3$. Der einzige Eigenwert $\lambda = 2$ kommt also in beiden Fällen dreimal vor: Die algebraische Multiplizität (des EW 2) ist in beiden Fällen gleich drei.

- Matrix A: Als Eigenvektoren kann man die die Basisvektoren $\vec{e}_1$, $\vec{e}_2$ und $\vec{e}_3$ identifizieren. Sie sind linear unabhängig voneinander. (Im Prinzip kann man aber jeden Vektor als Eigenvektor identifizieren, genauso wie bei I.) Die geometrische Multiplizität ist daher ebenfalls gleich drei.

- Matrix B: Man kann nur einen einzigen Eigenvektor finden. Er ist ein Vielfaches des Einheitsvektors $\vec{e}_1$.[5] Die geometrische Multiplizität beträgt also nur eins.

Ein Eigenwert, für den die geometrische Multiplizität kleiner ist als die algebraische, wird *defekter Eigenwert* genannt. Dann gibt es zu einem entarteten Eigenwert mindestens zwei identische (linear abhängige) Eigenvektoren. Eine Matrix mit mindestens einem defekten Eigenwert wird *defekte Matrix* genannt (z.B. Matrix B im obigen Beispiel). Man kann folgenden Satz beweisen (hier nicht gezeigt):

Eine $N \times N$-Matrix A ist nicht defekt
$\Leftrightarrow$
die Matrix A besitzt eine Eigenwertzerlegung $\mathsf{A} = \mathsf{X} \cdot \mathsf{\Lambda} \cdot \mathsf{X}^{-1}$.

6.1.4. Vorbetrachtung zur Berechnung von Eigenwerten

Für die Berechnung von Eigenwerten gibt es unterschiedliche Verfahren, von denen wir einige noch diskutieren werden. Man könnte nun auf die Idee kommen, die Eigenwerte einer Matrix dadurch zu berechnen, daß man die Nullstellen des charakteristischen Polynoms (6.468) bestimmt. Dies ist aber keine gute Idee, da die Berechnung der Nullstellen eines Polynoms ein schlecht konditioniertes Problem ist, selbst dann, wenn die Matrix A gut konditioniert ist.

Zunächst soll jedoch gezeigt werden, daß man zu einem gegebenen Polynom m-ten Grades

$$p(z) = z^m + a_{m-1}z^{m-1} + \ldots + a_1 z + a_0 \tag{6.470}$$

diejenige $m \times m$ Matrix finden kann, deren Eigenwerte die Nullstellen des vorgege-

[5] Um dies zu sehen, suche Lösungen der Eigenwertgleichung $2\vec{x} = \mathsf{B} \cdot \vec{x}$ mit $\vec{x} = (x, y, z)^{\mathrm{T}}$ zum Eigenwert 2 und setze $x = a$. Dann folgt $y = z = 0$.

benen Polynoms sind. Bis auf den Faktor $(-1)^m$ lautet diese Matrix

$$p(z) = (-1)^m \det \begin{pmatrix} -z & & & & & -a_0 \\ 1 & -z & & & & -a_1 \\ & 1 & -z & & & -a_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \ddots & -z & -a_{m-2} \\ & & & & 1 & -z - a_{m-1} \end{pmatrix}, \tag{6.471}$$

Dies kann man leicht verifizieren, indem man die Determinante der Matrix durch Entwicklung nach der letzten Spalte berechnet.

Die Matrix (6.471) können wir auch schreiben als

$$\begin{pmatrix} -z & & & & & -a_0 \\ 1 & -z & & & & -a_1 \\ & 1 & -z & & & -a_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \ddots & -z & -a_{m-2} \\ & & & & 1 & -z - a_{m-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & & & & & -a_0 \\ 1 & 0 & & & & -a_1 \\ & 1 & 0 & & & -a_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \ddots & 0 & -a_{m-2} \\ & & & & 1 & -a_{m-1} \end{pmatrix}}_{\mathsf{A}} - z\mathsf{I}. \tag{6.472}$$

Für $z = \lambda$ ist laut Voraussetzung $p(\lambda) = 0$ und wir können das zugehörige Eigenwertproblem identifizieren als

$$(\mathsf{A} - \lambda\mathsf{I}) \cdot \vec{x} = 0. \tag{6.473}$$

Die Matrix A wird auch *Begleitmatrix* (*companion matrix*) von $p(z)$ genannt. Sie ist also die Matrix, welche ein vorgegebenes charakteristisches Polynom produziert.

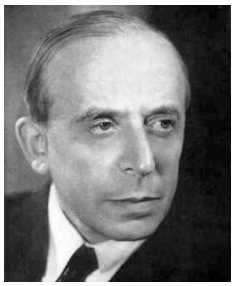
Abel hat 1824 gezeigt, daß man die Nullstellen eines Polynoms mit Ordnung $m > 4$ nicht geschlossen (d.h. durch rationale Zahlen, die vier Grundrechenarten und m-te Wurzeln) als Formeln darstellen kann. Dies bedeutet, daß wir die exakten Nullstellen eines Polynoms mit $m \geq 5$ auch dann nicht in einem Computer darstellen können, selbst wenn wir eine exakte Arithmetik hätten und unendlich lange rechen würden. Übertragen auf die Eigenwerte bedeutet dies, daß auch sie nicht exakt berechnet werden können. Dieser Befund steht im Gegensatz zu Algorithmen, wie z.B. der Gauß-Elimination, mit der man die exakte Lösung eines linearen Gleichungssystems nach einer endlichen Anzahl von Rechenschritten erhielte, falls die Arithmetik exakt wäre.

Niels Henrik Abel
1802–1829

Wir müssen schließen, daß die Eigenwerte von Matrizen mit $m \geq 5$ auf jeden Fall *nur iterativ* berechnet werden können. Tatsächlich gibt es iterative Methoden, die sehr schnell konvergieren, wobei in jedem Schritt zwei oder drei Dezimalstellen in der Genauigkeit gewonnen werden. Die Lösung eines Eigenwertproblems ist damit nicht sehr viel aufwendiger als die Lösung eines linearen Gleichungssystems.

6.2. Numerische Berechnung bestimmter Eigenwerte

6.2.1. Berechnung des größten Eigenwerts



Richard von Mises*
1883–1953

Wenn man nicht an allen Eigenwerten interessiert ist, sondern nur an dem betragsmäßig größten oder dem betragsmäßig kleinsten Eigenwert, dann kann man ein Potenzverfahren anwenden, das auf von Mises zurückgeht. Dazu ordnen wir die Eigenwerte entsprechend ihres Betrags an

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_N|. \tag{6.474}$$

Weiter nehmen wir an, daß der betragsmäßig größte Eigenwert λ_1 nicht entartet ist, also nur einmal vorkommt.[6] Man kann nun den Eigenwert λ_1 und den zugehörigen Eigenvektor $\vec{x}_1$ durch ein *Potenzverfahren* berechnen, wobei ein beliebiger Startvektor $\vec{x}^{(0)}$ wiederholt mit der Matrix A multipliziert wird. Mit der Normierung $\vec{y}^{(0)} = \vec{x}^{(0)} / \left\|\vec{x}^{(0)}\right\|$ lautet die Iteration

$$\vec{x}^{(k)} = \mathsf{A} \cdot \vec{y}^{(k-1)}, \tag{6.475a}$$

$$\vec{y}^{(k)} = \frac{\vec{x}^{(k)}}{\|\vec{x}^{(k)}\|}, \tag{6.475b}$$

$$\lambda^{(k)} = \vec{y}^{(k)*} \cdot \mathsf{A} \cdot \vec{y}^{(k)}. \tag{6.475c}$$

In (6.475c) erkennen wir den Rayleigh-Quotienten (6.454) wieder. Den iterierten Vektor $\vec{y}^{(k)}$ kann man explizit durch den normierten Startvektor $\vec{y}^{(0)}$ ausdrücken

$$\vec{y}^{(k)} = \frac{\mathsf{A}^k \cdot \vec{y}^{(0)}}{\prod_{i=1}^{k} \|\vec{x}^{(i)}\|}. \tag{6.476}$$

Im Limes $k \to \infty$ erhält man den betragsmäßig größten Eigenwert und den zugehörigen Eigenvektor

$$\lim_{k\to\infty} \lambda^{(k)} = \lambda_1 \qquad \text{und} \qquad \lim_{k\to\infty} \vec{y}^{(k)} = \vec{y}_1, \tag{6.477}$$

wobei $\vec{y}_1$ der auf eins normierte Eigenvektor zum Eigenwert λ_1 ist.

Der Beweis nutzt die Darstellung des Anfangsvektors als Überlagerung von Eigenvektoren, ähnlich wie bei der Berechnung der Konvergenzgeschwindigkeit von iterativen Gleichungslösern in Kap. 2.2.2. Wir nehmen an, daß die Eigenvektoren

*Richard Edler von Mises war ein österreichischer Mathematiker, der bedeutende Beiträge zur numerischen Mathematik, Strömungsmechanik, Aerodynamik, sowie Statistik und Wahrscheinlichkeitstheorie lieferte. Seit 1989 vergibt die Gesellschaft für Angewandte Mathematik und Mechanik (GAMM) alljährlich den *Richard-von-Mises-Preis*.

[6] Es darf also nicht ein bei reellen unsymmetrischen Matrizen häufig vorkommender konjugiert komplexer Eigenwert sein.

$\vec{x}_i$ von A linear unabhängig sind und den Raum $\mathbb{C}^N$ aufspannen. Dann können wir einen beliebigen Startvektor als Linearkombination der Eigenvektoren darstellen

$$\vec{x}^{(0)} = \sum_{i=1}^{N} a_i \vec{x}_i. \tag{6.478}$$

Bei jeden Iterationsschritt (6.475) wird aus der Matrixmultiplikation mit A im i-ten Summanden ein Faktor λ_i. Außerdem wird das Ergebnis noch normiert, wodurch in jedem Schritt ein Vorfaktor $\left\|\vec{x}^{(k)}\right\|^{-1}$ entsteht. Im k-ten Schritt haben wir dann

$$\vec{y}^{(k)} = \frac{1}{\|\vec{x}^{(0)}\| \ldots \|\vec{x}^{(k)}\|} \sum_{i=1}^{N} a_i \lambda_i^k \vec{x}_i. \tag{6.479}$$

Dieses Ergebnis kann man auch in der Form

$$\vec{y}^{(k)} = \frac{\lambda_1^k}{\|\vec{x}^{(0)}\| \ldots \|\vec{x}^{(k)}\|} \left[a_1 \vec{x}_1 + \underbrace{\sum_{i=2}^{N} a_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \vec{x}_i}_{\to 0} \right], \tag{6.480}$$

schreiben. Mit $|\lambda_i/\lambda_1| < 1$ für alle $i \geq 2$ erkennt man, daß der zweite Summand in der eckigen Klammer für $k \to \infty$ verschwindet. Es bleibt dann nur der Beitrag von $\vec{x}_1$ übrig. Selbst wenn a_1 zu Beginn der Iteration zufällig Null ist, so wird doch durch Rundungsfehler immer eine kleine Komponente $\sim \vec{x}_1$ generiert, so daß die Iteration zum Erfolg führt. In diesem Fall wirkt sich der Rundungsfehler ausnahmsweise einmal positiv aus!

Für den Fehler ist der Term mit $i = 2$ relevant (zweitgrößter Eigenwert). Im k-ten Schritt erhalten wir für die Abweichung vom exakten Eigenvektor[7][8]

$$\left\|\vec{y}^{(k)} - \vec{x}_1\right\| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right). \tag{6.481}$$

Für die Abweichung vom Eigenwert gilt

$$\left|\lambda^{(k)} - \lambda_1\right| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right). \tag{6.482}$$

Hierbei taucht der zweifache Exponent auf, da der Rayleigh-Quotient quadratisch genau ist (siehe (6.457) und auch Trefethen and Bau, III, 1997).

Die Potenzmethode ist sehr einfach anzuwenden.[9] Sie ist aber eingeschränkt, da nur der betragsmäßig größte Eigenwert berechnet werden kann. Darüber hinaus

[7] Die Eigenvektoren $\vec{x}_i$ seien auch auf 1 normiert.

[8] Wenn $\lambda_1 < 0$ ist, muß man auf das Vorzeichen von $\vec{x}_1$ achten, denn das Vorzeichen von $\vec{y}^{(k)}$ kann sich dann in jedem Iterationsschritt ändern.

[9] Google verwendet die Potenzmethode zur Berechnung des *Page Rank*.

wird der Fehler des Eigenvektors in jedem Schritt nur um den festen Faktor $|\lambda_2/\lambda_1|$ verringert (lineare Konvergenz). Dies ist insbesondere dann problematisch, wenn sich der größte Eigenwert λ_1 und der zweitgrößte λ_2 betragsmäßig nur wenig unterscheiden.

Beachte außerdem, daß $\vec{y}^{(k)}$ bei einem komplexen Eigenwert $\lambda_1 = a\mathrm{e}^{\mathrm{i}\varphi}$ nicht konvergiert, sondern rotiert: Die Matrix-Multiplikation mit A liefert dann für hinreichend große k

$$\vec{y}^{(k+1)} = \frac{\vec{x}^{(k+1)}}{\|\vec{x}^{(k+1)}\|} = \frac{\mathsf{A}\cdot\vec{y}^{(k)}}{\|\vec{x}^{(k+1)}\|} \cong \frac{a\mathrm{e}^{\mathrm{i}\varphi}\vec{y}^{(k)}}{\|\vec{x}^{(k+1)}\|} = \mathrm{e}^{\mathrm{i}\varphi}\vec{y}^{(k)}, \tag{6.483}$$

und man erhält im Limes $\vec{y}^{(k)} = \mathrm{e}^{\mathrm{i}k\varphi}\vec{y}_1$, wobei $\vec{y}_1$ der auf 1 normierte Eigenvektor ist. Ein Phasenfaktor $\mathrm{e}^{\mathrm{i}k\varphi}$ hat keinen Einfluß auf den Rayleigh-Quotienten, also den Eigenwert. Ein normierter Eigenvektor ist nur bis auf einen Phasenfaktor bestimmt.

Als Beispiel ist in Abb. 6.3a das Ergebnis der Potenz-Iteration für die 10×10-Tridiagonal-Matrix

$$\mathsf{A} = \begin{pmatrix} 1 & 1+\mathrm{i} & & & & & & & & \\ -5 & 2 & 1 & & & & & & & \\ & -5 & 3 & 1 & & & & & & \\ & & -5 & 4 & 1 & & & & & \\ & & & -5 & 5 & 1 & & & & \\ & & & & -5 & 6 & 1 & & & \\ & & & & & -5 & 7 & 1 & & \\ & & & & & & -5 & 8 & 1 & \\ & & & & & & & -5 & 9 & 1 \\ & & & & & & & & -5+30\mathrm{i} & 10 \end{pmatrix} \tag{6.484}$$

gezeigt (blaue Linie). Man sieht, daß die Potenz-Iteration auf den betragsmäßig größten Eigenwert führt.

6.2.2. Berechnung des kleinsten Eigenwerts

Wenn die Matrix A nicht singulär ist, kann man sie im Prinzip invertieren. Dann können wir ausnutzen, daß die Eigenwerte der inversen Matrix A^{-1} gerade die Kehrwerte der Eigenwerte λ_i von A sind (siehe (A.639)). Die Eigenvektoren ändern sich nicht! Wenn man dann die Potenz-Methode mit der Matrix A^{-1} durchführt, konvergiert die Iteration auf den inversen Eigenwert λ^{-1} mit dem größten Betrag, also auf den Eigenwert λ mit dem kleinsten Betrag.

Damit lautet die *Methode der inversen Potenzen* (auch *inverse Iteration* genannt)

$$\vec{x}^{(k)} = \mathsf{A}^{-1}\cdot\vec{y}^{(k-1)}, \tag{6.485a}$$

$$\vec{y}^{(k)} = \frac{\vec{x}^{(k)}}{\|\vec{x}^{(k)}\|}, \tag{6.485b}$$

$$\mu^{(k)} = \vec{y}^{(k)*}\cdot\mathsf{A}\cdot\vec{y}^{(k)}. \tag{6.485c}$$

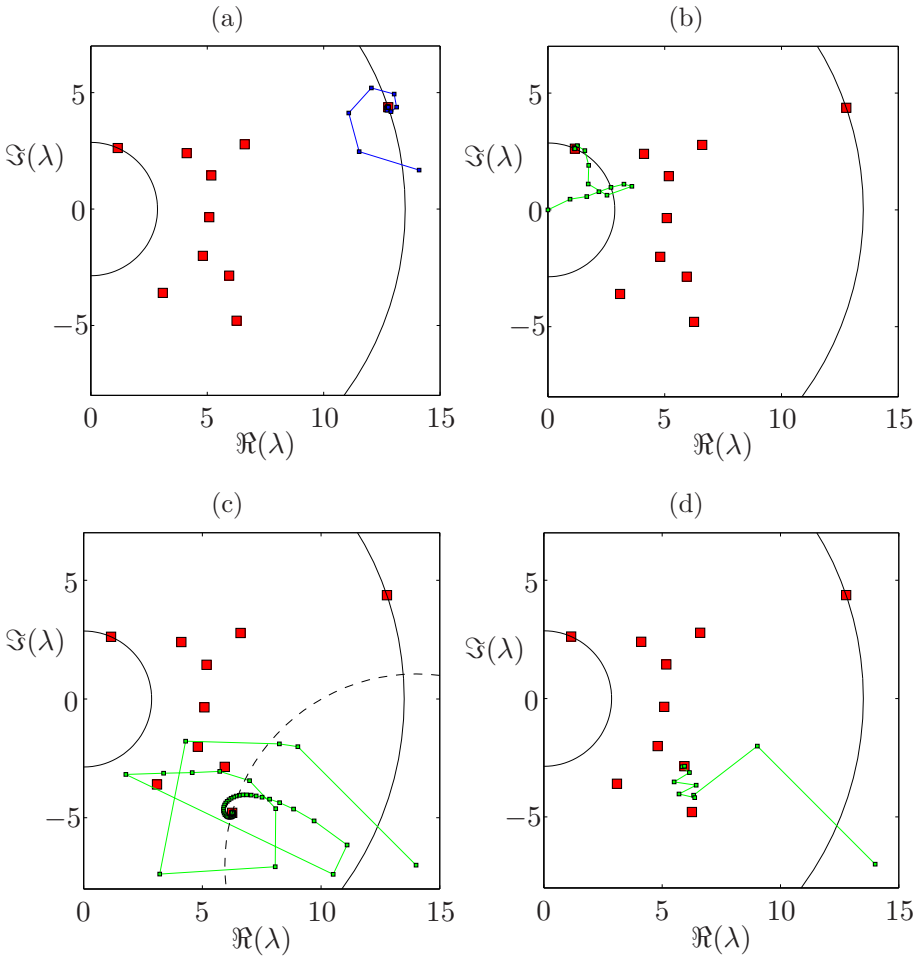


Abbildung 6.3.: Iteration von bestimmten Eigenwerten der Matrix (6.484) (rote Quadrate). Die Kreise zeigen den betragsmäßig kleinsten und den größten EW an. In allen Fällen wurde der Anfangsvektor $\vec{x}^{(0)} = (1, \ldots, 1)^{\mathrm{T}}$ gewählt. Sei K die Anzahl der Iterationen, die erforderlich war, um die Abbruchbedingung $|\lambda^{(k)} - \lambda^{(k-1)}| < 10^{-6}$ zu erfüllen. (a) Potenz-Iteration blauen Quadrate, $K = 32$. (b) Inverse Iterationen (mit Shift $\mu = 0$) grüne Quadrate, $K = 32$. (c) Inverse Iterationen mit Shift $\mu = 14 - 7\mathrm{i}$, $K = 132$. Die gestrichelte Linie zeigt den zu μ nächstgelegenen Eigenwert an. (d) Rayleigh-Quotient-Iteration mit demselben anfänglichen Shift μ wie in (c), $K = 13$.

Die Iteration konvergiert dann zu dem Eigenvektor mit dem kleinsten Eigenwert von A. Damit ist $\mu^{(k)}$ eine Schätzung des kleinsten Eigenwertes von A (des größten von A^{-1}). Wenn A genau N linear unabhängige Eigenvektoren besitzt, dann liefert dieses Verfahren im Limes

$$\mu_N = \lim_{k\to\infty} \mu^{(k)} = \lambda_N, \tag{6.486}$$

wobei λ_N der betragsmäßig kleinste Eigenwert von A ist. Anstelle der Matrix-Multiplikation beim Potenzverfahren (6.475) müssen wir hier bei der inversen Ite-

ration im k-ten Schritt ein lineares Gleichungssystem

$$\mathsf{A} \cdot \vec{x}^{(k)} = \vec{y}^{(k-1)} \tag{6.487}$$

lösen. Aus diesem Grund ist es sinnvoll, zu Beginn der Rechnung eine LU-Zerlegung der Matrix vorzunehmen mit $\mathsf{A} = \mathsf{L} \cdot \mathsf{U}$. Dann braucht man in jedem Schritt nur Dreiecksmatrizen zu lösen. Für symmetrische positiv definite Matrizen wäre eine Cholesky-Zerlegung $\mathsf{A} = \mathsf{L} \cdot \mathsf{L}^{\mathrm{T}}$ sinnvoll.

6.2.3. Inverse Iteration: Berechnung eines bestimmten Eigenwerts

Man kann das einfache Potenzverfahren verallgemeinern, um denjenigen Eigenwert λ_μ von A zu berechnen, der einer gegebenen komplexen Zahl μ am nächsten liegt. Um das zu sehen, beachten wir zunächst, daß die Eigenwerte von

$$\mathsf{A}^{\text{shift}} = \mathsf{A} - \mu\mathsf{I} \tag{6.488}$$

durch $\lambda^{\text{shift}} = \lambda - \mu$ gegeben sind, denn jeder Eigenvektor von A ist auch Eigenvektor von I und damit auch Eigenvektor von $\mathsf{A}^{\text{shift}}$. Wenn wir nun das inverse Potenzverfahren (6.485) auf $\mathsf{A}^{\text{shift}}$ anwenden, konvergiert die Iteration zu dem Eigenvektor, der zu dem kleinsten Eigenwert von $\mathsf{A}^{\text{shift}}$ gehört, also zu

$$\lambda^{\text{shift}}_{\min} = \lambda_\mu - \mu, \qquad \text{wobei} \quad |\lambda_\mu - \mu| = \min_i |\lambda_i - \mu| \,. \tag{6.489}$$

Hierbei ist λ_μ derjenige Eigenwert von A, der dem Wert μ am nächsten liegt. Es ist also

$$\lambda_\mu = \lambda^{\text{shift}}_{\min} + \mu. \tag{6.490}$$

Dieses Art von Verfahren heißt *Potenzverfahren mit Shift*, *inverse Iteration* oder *gebrochene Iteration von Wielandt*. Die Verschiebung (*shift*) μ kann man beliebig wählen.

Der Algorithmus stellt sich dann folgendermaßen dar:

Inverse Iteration (Wielandt):
$\vec{y}^{(0)} =$ `ein Vektor mit` $\|\vec{y}^{(0)}\| = 1$
`for` $k = 1, 2, \ldots$
  `Löse` $(\mathsf{A} - \mu\mathsf{I}) \cdot \vec{x}^{(k)} = \vec{y}^{(k-1)}$ (Anwendung von $(\mathsf{A} - \mu\mathsf{I})^{-1}$)
  $\vec{y}^{(k)} = \vec{x}^{(k)} / \left\|\vec{x}^{(k)}\right\|$ (Normierung)
  $\lambda^{(k)} = \vec{y}^{(k)*} \cdot \mathsf{A} \cdot \vec{y}^{(k)}$ (Rayleigh-Quotient)
`end`

In der letzten Zeile haben wir A verwendet, denn ein Eigenvektor von $\mathsf{A}^{\text{shift}} = \mathsf{A} - \mu\mathsf{I}$ ist auch Eigenvektor von A. Im Limes geht $\vec{x}^{(k)} \to \vec{x}_\mu$ und $\lambda^{(k)} \to \lambda_\mu$.

Es kann nun sein, daß für gegebenes μ die Matrix $\mathsf{A} - \mu\mathsf{I}$ singulär oder nahezu singulär ist. Dies stellt aber kein Problem für die inverse Iteration dar. Denn auch wenn die numerisch (aber rückwärts stabil) berechnete und fehlerbehaftete Lösung $\tilde{\vec{x}}^{(k)}$ stark von der exakten Lösung $\vec{x}^{(k)}$ abweicht, ist die berechnete *normierte* Lösung $\tilde{\vec{y}}^{(k)} = \tilde{\vec{x}}^{(k)}/\|\tilde{\vec{x}}^{(k)}\|$ nur wenig von $\vec{y}^{(k)}$ verschieden (siehe z.B. Trefethen and Bau, III, 1997).

Helmut Wielandt
1910–2001

Wie das normale Potenzverfahren konvergiert die inverse Iteration auch nur linear; der Fehler wird bei jedem Schritt um einen konstanten Faktor reduziert. Dieser Faktor hängt aber davon ab, wie dicht μ bei einem Eigenwert von A liegt. Je dichter μ an einem Eigenwert liegt, desto schneller ist die Konvergenz, denn der größte Eigenwert von $(\mathsf{A} - \mu\mathsf{I})^{-1}$ ist dann sehr viel größer als alle anderen Eigenwerte von $(\mathsf{A} - \mu\mathsf{I})^{-1}$. Die immer schlechter werdende Kondition von $\mathsf{A} - \mu\mathsf{I}$ macht in diesem Fall also kein Problem.

In Abb. 6.3 ist das Ergebnis der inversen Iteration für die obige Matrix (6.484) und für $\mu = 0$ (b) und $\mu = 14 - 7\mathrm{i}$ (c) gezeigt (grüne Quadrate). Man erkennt, daß die inverse Iteration auf denjenigen Eigenwert führt, der dem gewählten Wert μ am nächsten liegt.

Die inverse Iteration ist ein sehr wichtiges Werkzeug der numerischen linearen Algebra. Wenn die Eigenwerte schon bekannt sind, kann man die Eigenvektoren nach dem obigen Algorithmus sofort berechnen, wobei die Berechnung des Rayleigh-Quotienten entfallen kann. Oft kennt man das Spektrum von A aber nicht. Wenn man dann zum Beispiel den Eigenwert mit dem größten Realteil sucht, könnte man den in Frage kommenden Bereich in der komplexen Ebene mit Schätzwerten μ_i möglichst dicht abdecken und untersuchen, zu welchen Eigenwerten die inverse Iteration führt. Oft kann man so den gesuchten Eigenwert mit großer Wahrscheinlichkeit identifizieren. Es gibt aber bessere Verfahren (siehe z.B. Meerbergen et al., 1994).

Eine Beschleunigung der inversen Iteration erhält man, wenn man in jedem Iterationsschritt μ durch die jeweils aktuelle Näherung des Eigenwertes $\lambda^{(k-1)}$ ersetzt. Dann erhält man die *Rayleigh-Quotient-Iteration*

Rayleigh-Quotient-Iteration:
$\vec{y}^{(0)} =$ `ein Vektor mit` $\|\vec{y}^{(0)}\| = 1$
$\lambda^{(0)} = \vec{y}^{(0)*} \cdot \mathsf{A} \cdot \vec{y}^{(0)}$ (Rayleigh-Quotient)
`for` $k = 1, 2, \ldots$
  `Löse` $\left(\mathsf{A} - \lambda^{(k-1)}\mathsf{I}\right) \cdot \vec{x}^{(k)} = \vec{y}^{(k-1)}$ (Anwendung von $(\mathsf{A} - \mu\mathsf{I})^{-1}$)
  $\vec{y}^{(k)} = \vec{x}^{(k)} / \left\|\vec{x}^{(k)}\right\|$ (Normierung)
  $\lambda^{(k)} = \vec{y}^{(k)*} \cdot \mathsf{A} \cdot \vec{y}^{(k)}$ (Rayleigh-Quotient)
`end`

Anstelle von $\lambda^{(0)}$ kann man auch irgendeinen anderen Anfangswert nehmen.[10]

Man kann zeigen (Trefethen and Bau, III, 1997), daß diese Methode kubisch konvergiert; sowohl für den Eigenvektor als auch für den Eigenwert. In der Numerik existiert kaum eine iterative Methode die schneller ist. In jedem Schritt gewinnt man ca. 3 Dezimalen. Wenn man das Verfahren so implementiert, ist es jedoch aufwendig, weil sich die Matrix in jedem Iterationsschritt ändert. Es ist daher ratsam, die Schätzung des Eigenwerts $\lambda^{(k-1)}$ in $\left(\mathsf{A} - \lambda^{(k-1)}\mathsf{I}\right) \cdot \vec{x}^{(k)} = \vec{y}^{(k-1)}$ nicht in jedem Iterationsschritt zu korrigieren, sondern für einige Iterationen den Wert λ bzw. μ konstant zu halten. Denn dann muß man die LU-Zerlegung von $\left(\mathsf{A} - \lambda^{(k-1)}\mathsf{I}\right)$ nicht für jeden Schritt neu berechnen, sondern kann für einige Iterationen auf ein und dieselbe LU-Zerlegung zurückgreifen, wodurch die linearen Systeme wesentlich schneller gelöst werden können.

6.3. Numerische Berechnung aller Eigenwerte

Zur Berechnung *aller* Eigenwerte wird normalerweise versucht, die Matrix näherungsweise in eine Diagonal- oder eine Dreiecksmatrix zu überführen. In beiden Fällen kann man die Eigenwerte dann auf der Diagonalen ablesen. Die Transformation auf die gewünschte Gestalt kann man mit Hilfe einer Sequenz von Ähnlichkeitstransformationen (Kap. A.5.3) erreichen, da Ähnlichkeitstransformationen die Eigenwerte unverändert lassen.

Um Verfahren zur Berechnung aller Eigenwerte zu diskutieren, muß man sich zuerst mit den Möglichkeiten der Transformation der Matrix beschäftigen. Dies sprengt aber den Rahmen dieser Vorlesung. Daher werden wir hier nur einige bekannte Methoden vorstellen, die aber nicht immer die effizientesten sind.

6.3.1. Jacobi-Algorithmus

Wir haben gesehen, daß wir zur Berechnung der Eigenwerte von Matrizen mit Dimension $N \geq 5$ iterieren müssen. Eigenwerte von kleineren Matrizen können direkt berechnet werden. Jacobi hatte nun 1845 die Idee, sukzessive kleine Untermatrizen einer großen Matrix A zu diagonalisieren bis schließlich die gesamte Matrix diagonalisiert ist.

Standardmäßig werden dabei 2×2-Untermatrizen diagonalisiert. Wir beschränken uns hier auf reelle symmetrische Matrizen. Die Ähnlichkeitstransformation zur Diagonalisierung lautet dann

$$\mathsf{J}^{\mathrm{T}} \cdot \begin{pmatrix} a & d \\ d & b \end{pmatrix} \cdot \mathsf{J} = \begin{pmatrix} * & 0 \\ 0 & * \end{pmatrix}, \tag{6.491}$$

[10] Wenn man einen bestimmten Eigenwert sucht, wird man wohl erst eine inverse Iteration machen und erst wenn die Eigenwertschätzung hinreichend dicht am gesuchten Eigenwert liegt, auf die Rayleigh-Iteration umschalten. Dies ist wahrscheinlich auch der Grund, warum in Abb. 6.3d die Rayleigh-Iteration zwar schnell konvergiert, aber nicht auf den zum anfänglichen Shift nächstgelegenen Eigenwert.

wobei J orthogonal ist, d.h. $\mathsf{J}^{-1} = \mathsf{J}^{\mathrm{T}}$, dann ist $\mathsf{J}^{\mathrm{T}} \cdot \mathsf{J} = \mathsf{I}$ (siehe Kap. A.4). Es gibt verschiedene Möglichkeiten J zu wählen. Meist wird die Drehmatrix (*Jacobi-Rotation*)

$$\mathsf{J} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \tag{6.492}$$

verwendet, wobei $c = \cos\theta$ und $s = \sin\theta$ ist. Offensichtlich ist $\det \mathsf{J} = s^2 + c^2 = 1$. Die Bedingung für ein Verschwinden der Außerdiagonalelemente von (6.491) lautet $(b-a)sc = d(c^2-s^2)$. Dies führt auf den für eine Diagonalisierung von A notwendigen Winkel θ[11]

$$\tan(2\theta) = \frac{2d}{b-a}. \tag{6.493}$$

Wenn wir nun eine symmetrische $N \times N$-Matrix haben, wird die Jacobi-Rotation iterativ angewandt, wobei in jedem Schritt zwei Außerdiagonalelemente zu Null gemacht werden. Dazu iteriert man mit der leicht modifizierten Einheitsmatrix, wobei eine 2×2-Untermatrix aus einer Jacobi-Rotation besteht,

$$\underbrace{\begin{pmatrix} 1 & & & & & \\ & c & & -s & & \\ & & 1 & & & \\ & s & & c & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}}_{\mathsf{G}^{\mathrm{T}}} \cdot \underbrace{\begin{pmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}}_{\mathsf{A}} \cdot \underbrace{\begin{pmatrix} 1 & & & & & \\ & c & & s & & \\ & & 1 & & & \\ & -s & & c & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}}_{\mathsf{G}}$$

$$= \begin{pmatrix} * & * & * & * & * & * \\ * & * & * & \mathbf{0} & * & * \\ * & * & * & * & * & * \\ * & \mathbf{0} & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}. \tag{6.494}$$

Die Matrix G wird auch *reelle Givens-Rotation* genannt.[12] Durch die Multiplikation mit G von rechts werden die blauen Elemente von A verändert, durch die Multi-

[11] Beachte: $2(\cot\theta - \tan\theta)^{-1} = \tan(2\theta)$.

[12] Allgemein ist eine Givens-Rotation definiert durch

$$\mathsf{G} = \begin{pmatrix} 1 & & & & & \\ & c^* & & s^* & & \\ & & 1 & & & \\ & -s & & c & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix},$$

wobei

$$c = \mathrm{e}^{\mathrm{i}\alpha}\cos\theta \qquad \text{und} \qquad s = \mathrm{e}^{\mathrm{i}\beta}\sin\theta$$

mit $\theta \in (-\pi, \pi]$ und $\alpha, \beta \in [0, 2\pi)$. Givens-Rotationen sind unitäre Matrizen ($\mathsf{G}^{-1} = \mathsf{G}^{\dagger}$). Im vorliegenden und wichtigen reellen Fall ($\alpha = \beta = 0$) bewirkt die Matrix-Multiplikation mit G

plikation mit G^{T} von links die roten. Durch diesen Iterationsschritt werden zwei Elemente von A zu Null gemacht (grün). Andererseits werden existierende Nullen zerstört, die schon vorher in den blauen Spalten oder roten Zeilen vorhanden waren. Dies ist aber nicht so schlimm.

Diese Transformation wird nun iteriert, wobei alle Außerdiagonalelemente bearbeitet werden. Es wäre naheliegend, in jedem Schritt das betragsmäßig größte Außerdiagonalelement zu eliminieren. Für diese Strategie kann zeigen, daß die Summe der Quadrate der Außerdiagonalelemente in jedem Schritt mindestens um den Faktor $1 - 2/(N^2 - N)$ abnimmt. Daher muß man $O(N^2)$ Iterationen durchführen, um A zu diagonalisieren, wobei jede Iteration $O(N)$ Operationen erfordert. Maschinengenauigkeit erreicht man dann nach $O\left(N^3 |\log(\epsilon_{\text{machine}})|\right)$ Operationen.

Wallace Givens
1910–1993

Auf dem Computer wird man die Außerdiagonalpositionen aber in einer systematischen Abfolge bearbeiten. Damit kann man sich die Suche nach dem größten Element, die $O(N^2)$ Operationen kostet, ersparen. Dann muß man in einem Durchgang (*sweep*) $N(N-1)/2$ Paare von Elementen bearbeiten. Für Matrizen mit $N < 10^3$ kommt man normalerweise mit weniger als 10 Sweeps aus. Die dann erreichte Genauigkeit ist typischerweise besser als die, die man mit dem QR-Algorithmus (unten) erreicht.

Die Jacobi-Iteration ist nicht so populär, weil sie nicht so schnell ist wie Verfahren, die auf der Transformation auf Tridiagonalgestalt (symmetrische reelle Matrizen) und anschließendem QR-Algorithmus beruhen. Die Jacobi-Iteration ist typischerweise nur um einen Faktor 10 langsamer.

6.4. QR-Methode

In Kap. A.5.3 haben wir gesehen, daß eine Ähnlichkeitstransformation (A.645)

$$\mathsf{B} = \mathsf{Q}^{-1} \cdot \mathsf{A} \cdot \mathsf{Q}, \tag{6.495}$$

die Eigenwerte unverändert läßt. Die Eigenvektoren ändern sich aber. Zur Berechnung aller Eigenwerte einer Matrix kann man deshalb versuchen, die Matrix A durch eine Ähnlichkeitstransformation mittels der Transformationsmatrix Q auf eine Diagonalmatrix oder Dreiecksmatrix B zu bringen, denn dann stehen die Eigenwerte auf der Diagonalen von B. Wenn man dies in einem einzigen Schritt versucht, ist dies numerisch sehr aufwendig. Effizienter ist es, die Diagonalisierung iterativ zu erreichen.

Das bekanntest dieser Verfahren ist das *QR-Verfahren*.[13] Es wurde im Jahre 1961–1962 unabhängig voneinander von Francis und Kublanovskaya eingeführt. Das

eine Drehung um den Winkel θ in der Ebene, die von denjenigen Einheitsvektoren aufgespannt wird, die durch die Indexpositionen von c und c^* gegeben sind.

[13] Das QR-Verfahren ist in MatLab in der Funktion `D = eig(A)` implementiert, wobei der Vektor `D` alle Eigenwerte von `A` enthält.

Verfahren ist sehr aufwendig und deshalb für sehr große Matrizen mit $N > O\,(10^5)$ nicht mehr praktikabel.[14]

Bei dem QR-Verfahren wird die vollständige Matrix A wiederholt einer Ähnlichkeitstransformation mittels unitärer Transformationsmatrizen (komplex mit orthonormalen Spalten, $\mathsf{Q}^{-1} = \mathsf{Q}^\dagger$) unterworfen. Die Iteration lautet

$$\mathsf{A}_k = \mathsf{Q}_k^{-1} \cdot \mathsf{A}_{k-1} \cdot \mathsf{Q}_k. \tag{6.496}$$

Dies kann man schreiben als

$$\mathsf{Q}_k \cdot \underbrace{\mathsf{A}_k \cdot \mathsf{Q}_k^{-1}}_{:=\mathsf{R}_k} = \mathsf{A}_{k-1}, \tag{6.497}$$

was auf die algorithmische Form (jetzt mit hochgestellten Indizes) führt:

QR-Algorithmus:

$\mathsf{A}^{(0)} = \mathsf{A}$
`for` $k = 1, 2, \ldots$
  $\mathsf{Q}^{(k)} \cdot \mathsf{R}^{(k)} = \mathsf{A}^{(k-1)}$ (QR-Faktorisierung von $\mathsf{A}^{(k-1)}$)
  $\mathsf{A}^{(k)} = \mathsf{R}^{(k)} \cdot \mathsf{Q}^{(k)}$ (Rekombination in umgekehrter Reihenfolge)
`end`

Im Limes nimmt A die Gestalt einer oberen Dreiecksmatrix an, bei der die (reellen) Eigenwert auf der Diagonalen stehen.[15] Bei dem Algorithmus wird die Matrix in jedem Schritt in ein Produkt aus einer vollen Matrix Q und einer oberen Dreiecksmatrix R zerlegt und anschließend in umgekehrter Reihenfolge wieder zurückgebildet.

Die QR-Iteration konvergiert kubisch (wie die Rayleigh-Quotient-Iteration). Um diese Konvergenzrate zu erreichen, muß man die QR-Iteration aber in einer anderen Form (mit *shift*) implementieren. Dies würde hier zu weit führen. Daher wird nur das Prinzip der QR-Zerlegung behandelt. Man kann diese mit der Gram-Schmidt-Iteration erreichen.

6.4.1. Klassische Gram-Schmidt-Orthogonalisierung

Bei dem QR-Verfahren wird die Matrix $\mathsf{A} = \mathsf{Q} \cdot \mathsf{R}$ zerlegt in ein Produkt aus einer unitären Matrix Q mit orthonormalen Spaltenvektoren und einer oberen Dreiecksmatrix R. Diese Zerlegung ist eine der wichtigsten Operationen der numerischen

[14] Es existieren aber Varianten dieses Verfahrens, deren numerischer Aufwand nur mit N^3 skaliert (Multishift-Verfahren von Bai und Demmel 1989). Das numerisch stabilere Verfahren von Braman, Byers und Mathias (2002) ist auch in LAPACK implementiert.

[15] Im Falle komplexer Eigenwerte erhält man eine obere Blockdreiecksmatrix, bei der auf der Diagonalen auch antisymmetrische 2×2 Blöcke stehen können, wobei die Diagonalelemente der Blöcke den Realteil angeben und die Außerdiagonalelemente den Imaginärteil.

linearen Algebra. Wir können diese Zerlegung etwas genauer aufschreiben, wenn wir die Spaltenvektoren $\vec{a}_j$ und $\vec{q}_j$ von A und Q verwenden

$$\underbrace{\left[\begin{array}{c|c|c|c} & & & \\ \vec{a}_1 & \vec{a}_2 & \dots & \vec{a}_n \\ & & & \end{array}\right]}_{\mathsf{A}} = \underbrace{\left[\begin{array}{c|c|c|c} & & & \\ \vec{q}_1 & \vec{q}_2 & \dots & \vec{q}_n \\ & & & \end{array}\right]}_{\mathsf{Q}} \cdot \underbrace{\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix}}_{\mathsf{R}}. \tag{6.498}$$

Wenn jetzt A eine $m \times n$ Matrix ($m > n$) ist mit vollem Rang, d.h. $\operatorname{rang}(\mathsf{A}) = n$, ist, dann besteht die Aufgabe darin, die Dreiecksmatrix R so zu suchen, daß die Matrix Q aus orthonormalen Vektoren besteht.

Hierbei ist es sinnvoll, sich daran zu erinnern, daß man diese Relation so lesen kann: *Der j-te Spaltenvektor von A ist eine Superposition aller Spaltenvektoren von Q mit Gewichten, die in der j-ten Spalte von R stehen.*[16]

Dann wird klar, daß

$$\operatorname{span}(\vec{a}_1, \dots, \vec{a}_j) = \operatorname{span}(\vec{q}_1, \dots, \vec{q}_j). \tag{6.499}$$

Wenn wir die Spaltenvektoren von A gemäß (6.498) einzeln notieren, erhalten wir

$$\begin{aligned} \vec{a}_1 &= r_{11}\vec{q}_1, \\ \vec{a}_2 &= r_{12}\vec{q}_1 + r_{22}\vec{q}_2, \\ &\vdots \\ \vec{a}_n &= r_{1n}\vec{q}_1 + r_{2n}\vec{q}_2 + \dots r_{nn}\vec{q}_n. \end{aligned} \tag{6.500}$$

Dieses Gleichungssystem können wir aber sukzessive von oben nach unten nach den Vektoren $\vec{q}_j$ auflösen. Wir erhalten

$$\begin{aligned} \vec{q}_1 &= \frac{\vec{a}_1}{r_{11}}, \\ \vec{q}_2 &= \frac{\vec{a}_2 - r_{12}\vec{q}_1}{r_{22}}, \\ \vec{q}_3 &= \frac{\vec{a}_3 - r_{13}\vec{q}_1 - r_{23}\vec{q}_2}{r_{33}}, \\ &\vdots \\ \vec{q}_n &= \frac{\vec{a}_n - \sum_{i=1}^{n-1} r_{in}\vec{q}_i}{r_{nn}}, \end{aligned} \tag{6.501}$$

[16] In Indexschreibweise:

$$A_{ij} = Q_{ik}R_{kj}.$$

Zur Konstruktion der orthonormalen Vektoren $\vec{q}_j$ wird also in jedem Schritt ein weiterer Spaltenvektor $\vec{a}_j$ von A verwendet.

Damit hätten wir die orthonormalen Vektoren $\vec{q}_j$ gewonnen, wenn wir die Koeffizienten r_{nm} kennen würden. Diese erhalten wir durch die klassische *Gram-Schmidt-Orthogonalisierung* eines Satzes linear unabhängiger reeller Vektoren. Sei also $\{\vec{a}_j\}$ ein Satz linear unabhängiger Vektoren. Dann konstruiert man sich daraus einen anderen Satz von Vektoren $\{\vec{p}_n\}$, der orthonormal ist. Dazu nimmt man in jedem Schritt eine neue Raumrichtung hinzu, zieht aber davon alle vektoriellen Anteile ab, deren Richtungen schon von den vorherigen Vektoren abgedeckt werden. Dies liefert die folgende Gram-Schmidt-Orthogonalisierung

$$\begin{aligned}
\vec{v}_0 &= \vec{a}_0 & \vec{p}_0 &= \vec{v}_0/\|\vec{v}_0\|,\\
\vec{v}_1 &= \vec{a}_1 - (\vec{a}_1\cdot\vec{p}_0)\,\vec{p}_0 & \vec{p}_1 &= \vec{v}_1/\|\vec{v}_1\|,\\
\vec{v}_2 &= \vec{a}_2 - (\vec{a}_2\cdot\vec{p}_0)\,\vec{p}_0 - (\vec{a}_2\cdot\vec{p}_1)\,\vec{p}_1 & \vec{p}_2 &= \vec{v}_2/\|\vec{v}_2\|,\\
\vec{v}_3 &= \dots
\end{aligned} \tag{6.502}$$

Allgemein lautet also das Bildungsgesetz (der Index beginnt hier mit 1)

$$\vec{v}_j = \vec{a}_j - (\vec{a}_j\cdot\vec{p}_1)\,\vec{p}_1 - (\vec{a}_j\cdot\vec{p}_2)\,\vec{p}_2 - \dots - (\vec{a}_j\cdot\vec{p}_{j-1})\,\vec{p}_{j-1} \tag{6.503}$$

Wenn $\vec{a}_j$ komplexe Vektoren sind, muß man

$$\vec{v}_j = \vec{a}_j - (\vec{a}_j\cdot\vec{p}_1^*)\,\vec{p}_1 - (\vec{a}_j\cdot\vec{p}_2^*)\,\vec{p}_2 - \dots - \left(\vec{a}_j\cdot\vec{p}_{j-1}^*\right)\vec{p}_{j-1} \tag{6.504}$$

verwenden. Wenn man nun diese Gleichung mit (6.501) vergleicht, sieht man die Analogie. Durch Koeffizientenvergleich können wir nun die Elemente von R bestimmen zu

$$r_{ij} = \vec{a}_j\cdot\vec{q}_i^*; \qquad (i<j). \tag{6.505}$$

Die Nenner werden so gewählt, daß $\vec{q}_j$ auf 1 normiert ist

$$|r_{jj}| = \left\|\vec{a}_j - \sum_{i=1}^{j-1} r_{ij}\vec{q}_i\right\|_2. \tag{6.506}$$

Das Vorzeichen von r_{jj} (und damit dasjenige von $\vec{q}_j$) ist nicht festgelegt. Wir wählen $r_{jj}>0$. Dann hat R positive Diagonalelemente.

Damit haben wir den *klassischen Gram-Schmidt-Algorithmus*, der aus einem gegebenen Satz von n linear unabhängigen Vektoren $\{\vec{a}_j\}$ einen Satz $\{\vec{q}_j\}$ von orthonormalen Vektoren erzeugt, die denselben Vektorraum aufspannen:

Gram-Schmidt-Algorithmus (klassisch)
```
for j = 1,...,n
   v_j = a_j
   for i = 1,...,j-1
```

```
        r_ij = q_i^* · a_j
        v_j = v_j − r_ij q_i
    end
    r_jj = ‖v_j‖_2
    q_j = v_j / r_jj
end
```

Der Bezug zur QR-Zerlegung (6.498) besteht darin, daß der Satz der Vektoren $\{\vec{a}_j\}$ durch die Spaltenvektoren von A vorgegeben ist. Dann bildet der konstruierte Satz von Vektoren $\{\vec{q}_j\}$ die Spaltenvektoren von Q, und die Dreicksmatrix R wird durch die Faktoren r_{ij} aufgebaut, die im Rahmen der Gram-Schmidt-Orthogonalisierung anfallen.

Man kann allgemein zeigen, daß alle Matrizen eine QR-Zerlegung besitzen. Wenn die Matrix $\mathsf{A} \in \mathbb{C}^{m\times n}$ mit $m \geq n$ den vollen Rang hat, dann besitzt sie eine eindeutige QR-Zerlegung mit $r_{ii} > 0$.

Leider ist der obige Algorithmus instabil. Rundungsfehler der endlich genauen Arithmetik führen zu großen Fehlern.

6.4.2. Modifizierte Gram-Schmidt-Orthogonalisierung

Man kann das Problem der Instabilität umgehen, wenn man die Orthogonalisierung anders durchführt. Dazu betrachten wir den *orthogonalen Projektionsoperator*[17]

$$\mathsf{P}_{\perp\vec{a}} := \mathsf{I} - \frac{\vec{a}\,\vec{a}^*}{\vec{a}^*\cdot\vec{a}}. \tag{6.507}$$

Beachte, daß $\vec{a}\vec{a}^*$ eine Matrix ist, die aus dem dyadischen Produkt von $\vec{a}$ und $\vec{a}^*$ besteht. Der Nenner $\vec{a}^*\cdot\vec{a}$ ist nur ein Faktor, so daß der Vektor $\vec{a}$ auf 1 normiert wird.[18]

Wenn man den Projektionsoperator auf eine Vektor $\vec{b}$ anwendet, wird von diesem offenbar derjenige Anteil abgezogen, der in Richtung von $\vec{a}$ zeigt. Der resultierende Vektor besitzt dann keine Komponente in $\vec{a}$-Richtung mehr, da

$$\vec{a}^*\cdot\left(\mathsf{P}_{\perp\vec{a}}\cdot\vec{b}\right) = \vec{a}^*\cdot\left(\mathsf{I} - \frac{\vec{a}\,\vec{a}^*}{\vec{a}^*\cdot\vec{a}}\right)\cdot\vec{b} = \vec{a}^*\cdot\vec{b} - \vec{a}^*\cdot\frac{\vec{a}\,\vec{a}^*}{\vec{a}^*\cdot\vec{a}}\cdot\vec{b} = 0. \tag{6.508}$$

[17] Der parallele Projektionsoperator ist

$$\mathsf{P}_{\|\vec{a}} := \frac{\vec{a}\,\vec{a}^*}{\vec{a}^*\cdot\vec{a}}.$$

Offenbar gilt $\mathsf{P}_{\|\vec{a}} + \mathsf{P}_{\perp\vec{a}} = \mathsf{I}$.

[18] Beachte die Form des Skalarprodukts für komplexe Vektoren

$$\vec{a}^*\cdot\vec{b} = \sum_{i=1}^{N} a_i^* b_i.$$

Hierbei kann man * auch als das Adjungierte auffassen, wodurch aus dem Spaltenvektor ein Zeilenvektor wird.

Komponenten des projizierten Vektors $\vec{b}$, die senkrecht zu $\vec{a}$ sind, werden durch die Projektion nicht verändert.

Damit kann man die Matrix A beziehungsweise ihre linear unabhängigen Spaltenvektoren $\vec{a}_j$ durch eine Serie von orthogonalen Projektionen folgendermaßen orthogonalisieren ($\vec{q}_j = \vec{v}_j / \|\vec{v}_j\|$)

$$\begin{aligned}
\vec{v}_j^{(1)} &= \vec{a}_j, && && (j = 1, \ldots, n),\\
\vec{v}_j^{(2)} &= \mathsf{P}_{\perp\vec{q}_1} \cdot \vec{v}_j^{(1)} && = (\mathsf{I} - \vec{q}_1\, \vec{q}_1^{\,*}) \cdot \vec{v}_j^{(1)}, && (j = 2, \ldots, n),\\
\vec{v}_j^{(3)} &= \mathsf{P}_{\perp\vec{q}_2} \cdot \vec{v}_j^{(2)} && = (\mathsf{I} - \vec{q}_2\, \vec{q}_2^{\,*}) \cdot \vec{v}_j^{(2)}, && (j = 3, \ldots, n),\\
&\vdots && \vdots && \\
\vec{v}_j^{(j)} &= \mathsf{P}_{\perp\vec{q}_{j-1}} \cdot \vec{v}_j^{(j-1)} && = \left(\mathsf{I} - \vec{q}_{j-1}\, \vec{q}_{j-1}^{\,*}\right) \cdot \vec{v}_j^{(j-1)}, && (j = n).
\end{aligned} \tag{6.509}$$

Der hochgestellte Index in Klammern zeigt den jeweiligen Iterationsschritt an. Im ersten Schritt werden alle n betrachten Spaltenvektoren von Q (in nicht-normierter Version sind dies die Vektoren $\vec{v}_j$) identisch mit den entsprechenden Spaltenvektoren von A gesetzt. Im zweiten Durchgang wird $\vec{q}_1$ beibehalten und aus allen anderen Vektoren wird der Anteil in Richtung von $\vec{q}_1$ entfernt. Im dritten Schritt werden $\vec{q}_1$ und $\vec{q}_2$ beibehalten und aus allen anderen Vektoren der Anteil in Richtung $\vec{q}_2$ entfernt. Wenn man den Schritt (j) vollendet hat, sind die Vektoren $\vec{q}_1, \ldots, \vec{q}_j$ orthonormal. Man muß nur dabei aufpassen, denn in jedem Schritt (i) ändern sich die Vektoren $\vec{v}_j$ mit $j \geq i$. Deshalb muß man diese in jedem Schritt erneut normieren, um die entsprechenden Einheitsvektoren $\vec{q}_j$ zu erhalten. Dieses Verfahren führt auf den *modifizierten Gram-Schmidt-Algorithmus*

Gram-Schmidt-Algorithmus (modifiziert)

```
for i = 1, ..., n
    v_i = a_i
end
for i = 1, ..., n
    r_ii = ||v_i||
    q_i = v_i / r_ii
    for j = i + 1, ..., n
        r_ij = q_i^* · v_j
        v_j = v_j − r_ij q_i
    end
end
```

Der modifizierte Gram-Schmidt-Algorithmus ist stabil. In der Praxis wird $\vec{a}_i$ mit $\vec{v}_i$ und $\vec{v}_i$ mit $\vec{q}_i$ überschrieben, um Speicherplatz zu sparen. Man kann sich überlegen (Trefethen and Bau, III, 1997), daß der Algorithmus einen numerischen Aufwand von $O(2mn^2)$ Operationen erfordert.

In vielen Anwendungen mit quadratischen Matrizen $\mathsf{A} \in \mathbb{C}^{m\times m}$, insbesondere wenn m sehr sehr groß ist, wird man die Orthogonalisierung nicht bis $n = m$ durchführen, sondern nur bis zu einem moderate Wert von $n \ll m$. Der numerische Aufwand wäre sonst zu hoch $O(m^3)$. Wenn n jedoch klein ist, skaliert der Aufwand bzgl. m nur wie $\sim m^1$.

Weiteres zum QR-Algorithmus kann man in Trefethen and Bau, III (1997) ab Seite 211 finden. Zur Berechnung einzelner Eigenwerte gibt es effizientere Methoden. Sie basieren auf der Krylov-Unterraum-Iteration (siehe Kap. 7.1.3). Zu nennen sind hier die Arnoldi-Iteration, CG, GMRES und deren Derivate. Die Diskussion würde hier den Rahmen sprengen; für die *Arnoldi-Iteration* siehe aber Trefethen and Bau, III (1997) ab Seite 250. Trotzdem soll die Arnoldi-Iteration kurz skizziert werden.

6.4.3. Arnoldi-Iteration

Hintergrund ist die folgende Idee, die auch schon zur Potenzmethode in Kap. 6.2.1 geführt hat. Gegeben ein beliebiger Startvektor $\vec{v}$. Wir stellen uns vor, $\vec{v}$ sei durch eine Superposition von Eigenvektoren der gegebenen Matrix A dargestellt. Dann betrachte die wiederholte Multiplikation von $\vec{v}$ mit A. In resultierenden Vektoren für hinreichend großes m werden dann im wesentlichen durch die Eigenvektoren bestimmt, welche den größten Betrag haben (siehe (6.480)). Daher werden die Eigenvektoren zu den betragsmäßig größten Eigenwerten in sehr guter Näherung in dem Krylov-Unterraum

$$\mathcal{K}_m(\mathsf{A}, \vec{v}) = \text{span}\{\vec{v}, \mathsf{A}\cdot\vec{v}, \mathsf{A}^2\cdot\vec{v}, \dots\ \mathsf{A}^{m-1}\cdot\vec{v}\} \tag{6.510}$$

darstellbar sein. Um eine orthogonale Basis des Krylov-Raums zu erzeugen, muß man den Satz der Vektoren in (6.510) orthogonalisieren. Das kann man machen, ohne die Potenzen $\mathsf{A}^j\cdot\vec{v}$ explizit zu berechnen. Damit wird auch vermieden, daß diese Ausdrücke divergieren. Die Orthogonalisierung wird durch den folgenden Algorithmus bewerkstelligt (MGS: Modifiziertes Gram-Schmidt-Verfahren).

Arnoldi-Algorithmus (MGS):

Sei $\vec{q}_1$ ein Startvektor mit $\|\vec{q}_1\| = 0$
`for` $k = 1, \dots, m,$
  $\vec{v}_k = \mathsf{A}\cdot\vec{q}_k$ (nächster Vektor)
  `for` $j = 1, \dots, k$
    $h_{jk} = \vec{q}_j^* \cdot \vec{v}_k$ (Projektion schon vorhandenen Komponenten)
    $\vec{v}_k = \vec{v}_k - h_{jk}\vec{q}_j$ (Subtraktion vorhandener Komponenten)
  `end`
  $h_{k+1,k} = \|\vec{v}_k\|$
  `If` $h_{k+1,k} = 0$ `Stop`
  $\vec{q}_{k+1} = \vec{v}_k / h_{k+1,k}$
`end`

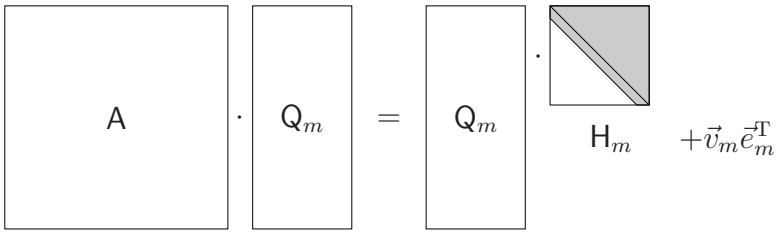


Abbildung 6.4.: Symbolische Darstellung von Gleichung (6.513) nach Saad (2003). Die Multiplikation der Matrix A mit Q_m ergibt Q_m skalar H_m plus eine Matrix vom Rang eins.

Man erkennt die Ähnlichkeit mit der Gram-Schmidt-Orthogonalisierung. Nach m Schritten hat das Arnoldi-Verfahren im wesentlichen eine Orthonormalbasis in der $n \times m$ Matrix $\mathsf{Q}_m = (\vec{q}_1, \ldots, \vec{q}_m)$ bestimmt und eine $(m+1) \times m$ Hessenbergmatrix

$$\tilde{\mathsf{H}}_m = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1m} \\ h_{21} & h_{22} & \ldots & h_{2m} \\ & \ddots & \ddots & \vdots \\ & & h_{m,m-1} & h_{mm} \\ & & & h_{m+1,m} \end{pmatrix}. \tag{6.511}$$

Sei $\tilde{\mathsf{H}}_m$ ohne die letzte Zeile die $m \times m$ Hessenbergmatrix

$$\mathsf{H}_m = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1m} \\ h_{21} & h_{22} & \ldots & h_{2m} \\ & \ddots & \ddots & \vdots \\ & & h_{m,m-1} & h_{mm} \end{pmatrix}. \tag{6.512}$$

Dann gilt (für den Beweis siehe Saad, 2003)

$$\mathsf{A} \cdot \mathsf{Q}_m = \mathsf{Q}_m \cdot \mathsf{H}_m + \vec{v}_m \vec{e}_m^{\mathrm{T}} = \mathsf{Q}_{m+1} \cdot \tilde{\mathsf{H}}_m. \tag{6.513}$$

Der zweite Summand auf der rechen Seite strebt gegen Null und wir haben näherungsweise eine QR-Zerlegung (QH-Zerlegung) erreicht (vgl. (6.495)). Dann sind die Eigenwerte der Matrix H_m eine gute Näherung der betragsmäßig größten Eigenwerte (am Rand des Spektrums) von A. Die Beziehung (6.513) ist grafisch in Abb. 6.4 dargestellt.

Wenn man (6.513) von links mit $\mathsf{Q}_m^{\mathrm{T}}$ multipliziert, erhält man[19]

$$\mathsf{Q}_m^{\mathrm{T}} \cdot \mathsf{A} \cdot \mathsf{Q}_m = \mathsf{H}_m. \tag{6.514}$$

Der Term $\mathsf{Q}_m^{\mathrm{T}} \cdot \vec{v}_m \vec{e}_m^{\mathrm{T}}$ verschwindet wegen der Orthogonalität der Spaltenvektoren $\vec{q}_m$ von Q_m, denn mit $\vec{v}_m = h_{m+1,m}\vec{q}_{m+1}$ (siehe Algorithmus) ist $\vec{v}_m \| \vec{q}_{m+1}$ und damit $\vec{v}_m \perp \vec{q}_j$, $j = 1, \ldots, m$.

[19] Für reelle orthogonale Matrizen ist $\mathsf{A}^{-1} = \mathsf{A}^{\mathrm{T}}$. Für komplexe orthogonale Matrizen gilt $\mathsf{A}^{-1} = \mathsf{A}^{\dagger}$.

Der Vorteil der Arnoldi-Iteration besteht darin, daß die Eigenwerte von A durch diejenigen von H_m approximiert werden. Die Dimension der $m \times m$ Matrix H_m ($m = 500$ reicht meist) ist nämlich sehr viel kleiner ist als diejenige der $n \times n$ Matrix A (oft mit $n > 10^5$).

7. Weiteres zu großen linearen Systemen

Der Gauß-Algorithmus ist ein direktes Verfahren zur Lösung linearer Gleichungssysteme mit voll-besetzter Matrix. Die Lösung erfordert $O(N^x)$ Operationen mit $x = 3$. Dieser Aufwand für direkte Löser dicht-besetzte Matrizen konnte in den letzten Jahrzehnten verbessert werden (Abb. 7.1). Obwohl der Exponent x verringert wurde, sind die Vorfaktoren oft sehr groß, so daß die effizienteren direkten Löser erst bei extrem großen Systemen schneller werden als die Gauß-Elimination. Für eine effiziente Lösung sind daher iterative Methoden erforderlich (siehe auch Kap. 2.2).

7.1. Minimierungsverfahren

Eine Klasse iterativer Verfahren zur Lösung linearer Gleichungssysteme basiert auf Ideen zur Lösung nichtlinearer Probleme. Diese kann man entweder mit dem *Newton-Verfahren* (siehe Kap. 3.5.6) oder dessen Varianten lösen, oder mit einem *Abstiegsverfahren* (*descent method*). Verfahren nach dem Newton-Typ konvergieren sehr schnell, haben aber einen recht kleinen Konvergenzradius, d.h. man benötigt schon eine recht gute Anfangsschätzung, damit das Newton-Verfahren konvergiert. Abstiegsverfahren konvergieren dagegen langsam, dafür aber garantiert!

Um ein Abstiegsverfahren zur Lösung eines linearen Problems

$$\mathsf{A} \cdot \vec{x} = \vec{b} \tag{7.515}$$

verwenden zu können, muß dieses zunächst in ein Minimierungsproblem überführt

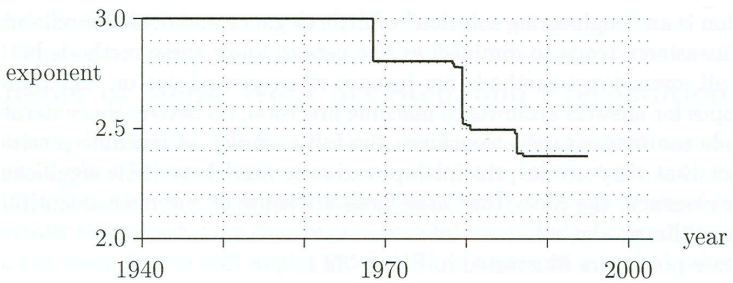


Abbildung 7.1.: Historische Entwicklung des besten bekannten Exponenten x der Skalierung $O(N^x)$ für die zur *direkten* Lösung des linearen Problems $\mathsf{A} \cdot \vec{x} = \vec{b}$ erforderlichen Operationen, wenn A dicht besetzt ist (nach Trefethen and Bau, III, 1997).

werden. Falls A *symmetrisch* und *positiv definit* ist,[1] dann ist das Minimum der skalaren Funktion

$$F(\vec{x}) = \frac{1}{2}\vec{x}\cdot\mathsf{A}\cdot\vec{x} - \vec{x}\cdot\vec{b} \tag{7.516}$$

identisch mit der Lösung des linearen Systems. Dies folgt aus der Bedingung, daß am Minimum von F alle partiellen Ableitungen nach den N Unbekannten verschwinden müssen. Diese Bedingung liefert[2]

$$0 \stackrel{!}{=} \nabla F(\vec{x}) = \nabla\left(\frac{1}{2}\vec{x}\cdot\mathsf{A}\cdot\vec{x} - \vec{x}\cdot\vec{b}\right) = \frac{1}{2}\left(\vec{x}\cdot\mathsf{A} + \mathsf{A}\cdot\vec{x}\right) - \vec{b} \stackrel{\mathsf{A}\ \text{symm.}}{=} \mathsf{A}\cdot\vec{x} - \vec{b}. \tag{7.517}$$

Angenommen, wir haben eine Approximation $\vec{x}^{(n)}$ der Lösung. Dann ist $\vec{\rho}^{(n)} = \vec{b} - \mathsf{A}\cdot\vec{x}^{(n)}$ das zugehörige Residuum. Wir möchten nun die Approximation durch eine Korrektur verbessern, wobei wir um eine gewisse Distanz $\alpha^{(n)}$ in eine bestimmte Richtung gehen, die wir mit $\vec{p}^{(n)}$ bezeichnen. Für die verbesserte Näherung setzen wir deshalb an

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} + \alpha^{(n)}\vec{p}^{(n)}. \tag{7.518}$$

Ziel ist es, $\alpha^{(n)}$ und $\vec{p}^{(n)}$ optimal zu wählen.

7.1.1. Bestimmung der Schrittweite

Sei zunächst die Richtung $\vec{p}^{(n)}$ vorgegeben. Dann ist es offenbar sinnvoll, die skalare Schrittweite $\alpha^{(n)}$ so zu wählen, daß F entlang der gewählten Richtung $\vec{p}^{(n)}$ minimal wird. Wir fordern also

$$F\left(\vec{x}^{(n)} + \alpha^{(n)}\vec{p}^{(n)}\right) = \min_{\alpha}\left[F\left(\vec{x}^{(n)} + \alpha\vec{p}^{(n)}\right)\right]. \tag{7.519}$$

Die Minimierung können wir leicht durchführen. Zunächst einmal ist die zu minimierende Funktion (A ist symmetrisch)

$$\begin{aligned} F\left(\vec{x}^{(n)} + \alpha\vec{p}^{(n)}\right) = {} & \frac{1}{2}\vec{x}^{(n)}\cdot\mathsf{A}\cdot\vec{x}^{(n)} + \alpha\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{x}^{(n)} + \frac{\alpha^2}{2}\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)} \\ & - \left(\vec{x}^{(n)} + \alpha\vec{p}^{(n)}\right)\cdot\vec{b}. \end{aligned} \tag{7.520}$$

Die Bedingung $\partial F/\partial\alpha\big|_{\alpha_{\min}} = 0$ liefert dann die lineare Gleichung in α

$$\left[-\vec{p}^{(n)}\cdot\vec{b} + \alpha\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)} + \vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{x}^{(n)}\right]_{\alpha_{\min}} = 0. \tag{7.521}$$

Nach $\alpha^{(n)} = \alpha_{\min}$ aufgelöst ergibt sich

$$\alpha^{(n)} = \frac{\vec{p}^{(n)}\cdot\vec{b} - \vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{x}^{(n)}}{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)}} = \frac{\vec{p}^{(n)}\cdot\vec{\rho}^{(n)}}{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)}}. \tag{7.522}$$

[1] Eine Matrix A ist positiv definit, wenn für alle $\vec{x} \neq 0$ mit $\vec{x} \in \mathbb{R}^N$ gilt $\vec{x}\cdot\mathsf{A}\cdot\vec{x} > 0$. Reelle symmetrische Matrizen haben reelle Eigenwerte. Sie sind positiv definit, wenn all Eigenwerte positiv sind.

[2] ∇ ist hier der N-dimensionale ∇-Operator $(\partial_1, \partial_2, ..., \partial_N)^{\mathrm{T}}$. Beachte $\nabla\vec{x} = \partial_i x_j = \delta_{i,j} = \mathsf{I}$.

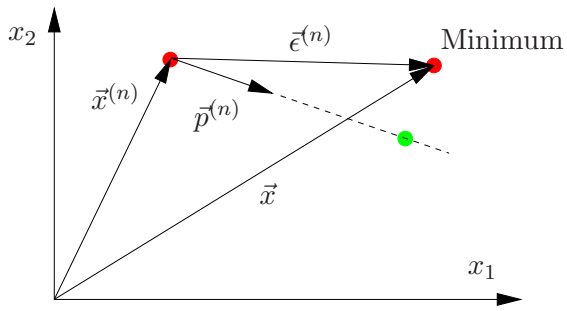


Abbildung 7.2.: Beispiel für eine mögliche Orientierung der Richtung $\vec{p}^{(n)}$. Der grüne Punkt (●)soll die Stelle andeuten, an der $F(\vec{x})$ entlang der Richtung $\vec{p}^{(n)}$ ein Minimum hat. $\vec{\epsilon}^{(n)} = \vec{x} - \vec{x}^{(n)}$ ist der Fehler.

Damit erhalten wir die Iteration

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} + \alpha^{(n)}\vec{p}^{(n)}, \tag{7.523}$$

mit $\alpha^{(n)}$ nach (7.522). Die Vektoren sind in in Abb. 7.2 illustriert. Die Richtung $\vec{p}^{(n)}$ haben wir bisher allerdings noch nicht festgelegt.

Da die Lösung des linearen Problems $\mathsf{A} \cdot \vec{x} = \vec{b}$ eindeutig ist, besitzt F auch nur ein Extremum. Dies ist klar, da F nur quadratisch von $\vec{x}$ abhängt.[3]

7.1.2. Methode des steilsten Abstiegs

Eine einfache Möglich, $\vec{p}^{(n)}$ zu wählen, besteht darin, die Richtung des aktuellen Residuums zu wählen $\vec{p}^{(n)} = \vec{\rho}^{(n)}$. Dann zeigt $\vec{p}^{(n)}$ in die Richtung des negativen Gradienten $-\nabla F\left(\vec{x}^{(n)}\right)$ (vgl. (7.517) und (2.65)). Zusammen mit $\alpha^{(n)}$ nach (7.522) erhalten wir so das *steepest-descent*-Verfahren, das auch als *Richardson-Verfahren* bekannt ist.

Dieses Verfahren ist eng mit dem Jacobi-Verfahren verwandt und konvergiert ähnlich langsam. Insbesondere wenn die Gradienten in verschiedenen Richtungen eine sehr unterschiedliche Größenordnungen besitzen (in zwei Dimensionen würde $F(\vec{x})$ dann ein sehr enges Tal beschreiben), kann es sein, daß die Iteration immer in Richtung der stärksten Änderung hin und her pendelt und nicht hinreichend schnell in den anderen Richtungen voran kommt.

Eine andere Möglichkeit besteht darin, die Richtungen $\vec{p}$ alle N orthogonalen Richtungen $\vec{e}_i$ periodisch durchlaufen zu lassen, welche den Raum der Unbekannten aufspannen, wobei α wieder nach (7.522) gewählt wird. Man kann zeigen, daß dann N Schritte äquivalent sind zu einem Gauß-Seidel-Iterationsschritt. Außerdem kann man $\alpha^{(n)}$ mit einem Relaxationsfaktor ω multiplizieren. Aufgrund der quadratischen Symmetrie des Minimums von F muß allerdings $0 < \omega < 2$ sein, damit sich F in dem Iterationsschritt auch verringert.

[3] Bei der Minimierung anderer nichtlinearer Funktionen können natürlich mehrere Extrema existieren. Es hängt dann von den Anfangswerten ab, zu welchem Minimum die Iteration führt.

7.1.3. Konjugierte Gradienten

A-Orthogonalität

Eine wichtige Variante besteht darin, die N Richtungen $\vec{p}^{(n)}$ so zu wählen, daß für sie gilt

$$\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(m)} = 0, \qquad \text{für } n \neq m. \tag{7.524}$$

Die N Vektoren $\vec{p}^{(n)}$ bezeichnet man dann als *A-orthogonal*. Man sagt auch: Die Richtungen $p^{(n)}$ und $p^{(m)}$, für welche (7.524) gilt, sind *konjugiert* zu einander.

Für die so definierten N konjugierten Richtungsvektoren $p^{(n)}$ kann man den folgenden Satz beweisen:

> **Satz:** Wenn A symmetrisch und positiv definit ist und wenn $\vec{p}^{(n)} \neq 0$ für alle $n = 0, ..., N-1$ und wenn man die Schrittweiten nach (7.522) wählt, dann konvergiert die Iteration (7.523) für jeden Startvektor $\vec{x}_0$ in höchsten $N-1$ Iterationen gegen die gesuchte Lösung des linearen Problems.

Man hat also nach spätestens $N-1$ Iterationen die Lösung bis auf Rundungsfehler gefunden. Damit ist das Verfahren der konjugierten Richtungen also eher ein direktes Verfahren.

Diesen Satz kann man allein mit Hilfe von (7.522) und (7.523) beweisen. Dazu untersuchen wir zunächst, wie das Residuum im Schritt $n+1$ von dem vorangegangen Residuum im Schritt n abhängt:

$$\begin{aligned}
\vec{\rho}^{(n+1)} \cdot \vec{p}^{(j)} &= \left(\vec{b} - \mathsf{A} \cdot \vec{x}^{(n+1)}\right) \cdot \vec{p}^{(j)} = \left(\underbrace{\vec{b} - \mathsf{A} \cdot \vec{x}^{(n)}}_{\vec{\rho}^{(n)}} - \alpha^{(n)} \mathsf{A} \cdot \vec{p}^{(n)}\right) \cdot \vec{p}^{(j)} \\
&\overset{(7.522)}{=} \vec{\rho}^{(n)} \cdot \vec{p}^{(j)} - \frac{\vec{p}^{(n)} \cdot \vec{\rho}^{(n)}}{\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(n)}} \vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(j)} = \begin{cases} \vec{\rho}^{(n)} \cdot \vec{p}^{(j)}, & n \neq j, \\ 0, & n = j. \end{cases}
\end{aligned} \tag{7.525}$$

Dies bedeutet, daß sich die Komponente des Residuums in jeder festen A-konjugierten Richtung $\vec{p}^{(j)}$ i.a. nicht ändert ($n \neq j$). Sie ändert sich nur in einem Fall, nämlich dann, wenn $n = j$ ist. Und in diesem Fall wird die betreffende Komponente des Residuums zu Null. Damit gilt für alle $j = 0, ..., N-1$ konjugierten Richtungen

$$\vec{\rho}^{(N)} \cdot \vec{p}^{(j)} = \vec{\rho}^{(N-1)} \cdot \vec{p}^{(j)} = ... = \vec{\rho}^{(j+1)} \cdot \vec{p}^{(j)} = 0, \tag{7.526a}$$

$$\vec{\rho}^{(j)} \cdot \vec{p}^{(j)} = \vec{\rho}^{(j-1)} \cdot \vec{p}^{(j)} = ... = \vec{\rho}^{(0)} \cdot \vec{p}^{(j)} \overset{\text{i.a.}}{\neq} 0. \tag{7.526b}$$

Das heißt, daß sich die Komponente des Residuums $\vec{\rho}^{(n)}$ in Richtung $\vec{p}^{(j)}$ im Laufe der Iteration solange nicht ändert, bis $n = j$ ist. In diesem Schritt wird die Komponente des Residuums $\vec{\rho}^{(n)}$ in Richtung $\vec{p}^{(j)}$ zu Null gemacht. Danach bleibt

diese Komponente Null. Da die Richtungen $\vec{p}^{(j)}$ linear unabhängig sind, kann man folgern, daß $\vec{\rho}^{(N)} = 0$ ist. Eventuell wird dies aber schon für ein $n < N$ erreicht.

Wenn man das Residuum $\vec{\rho}^{(n)} = \sum_{j=0}^{N-1} R_j \vec{p}^{(j)}$ durch die konjugierten Richtungen darstellt, ergibt sich für einen (j-ten) Iterationsschritt das folgende Bild für die Koeffizienten R_j

$$(0, \ldots, 0, R_j, R_{j+1}, \ldots) \longrightarrow (0, \ldots, 0, 0, R_{j+1}, \ldots)\,. \tag{7.527}$$

Hierbei bleiben alle blauen Entwicklungskoeffizienten R_j bis auf einen (rot) unverändert. Dies gilt natürlich nicht für die kartesischen Komponenten von $\vec{\rho}^{(n)}$. Vielmehr gilt für die Residuen (siehe auch (C.682))

$$\vec{\rho}^{(j)} \cdot \vec{\rho}^{(n)} = 0, \quad \text{für } j < n. \tag{7.528}$$

Es erhebt sich die Frage, wie man die konjugierten Richtungsvektoren $\vec{p}^{(n)}$ bestimmen kann. Eine offensichtliche Möglichkeit besteht darin, die N orthogonalen Eigenvektoren $\vec{\psi}^{(n)}$ von A verwenden. Sie sind offenbar A-orthogonal, denn es ist[4]

$$\vec{\psi}^{(m)} \cdot \mathsf{A} \cdot \vec{\psi}^{(n)} = \lambda_n \vec{\psi}^{(m)} \cdot \vec{\psi}^{(n)} = 0 \qquad \text{für } n \neq m. \tag{7.529}$$

Die Bestimmung der Eigenvektoren oder die A-Orthogonalisierung eines Satzes von linear unabhängigen Vektoren ist jedoch zu aufwendig.

Gram-Schmidt-Orthogonalisierung

Eine andere, effizientere Methode ist die *Gram-Schmidt-Orthogonalisierung* (siehe Kap. 6.4.1 und 6.4.2). Dabei nimmt man in jedem Schritt eine neue Raumrichtung hinzu, zieht aber davon alle vektoriellen Anteile ab, deren Richtungen schon von den vorherigen Vektoren abgedeckt worden sind. Es sei also eine Sequenz von linear unabhängigen Vektoren $\vec{a}_i$ gegeben. Dann generiert man eine Sequenz von orthogonalen Richtungsvektoren $\vec{p}_i$ (hier normieren wir sie) mittels der folgenden Gram-Schmidt-Orthogonalisierung

$$\begin{aligned}
\vec{v}_0 &= \vec{a}_0 & \vec{p}_0 &= \vec{v}_0/\|\vec{v}_0\|,\\
\vec{v}_1 &= \vec{a}_1 - (\vec{a}_1 \cdot \vec{p}_0)\,\vec{p}_0 & \vec{p}_1 &= \vec{v}_1/\|\vec{v}_1\|,\\
\vec{v}_2 &= \vec{a}_2 - (\vec{a}_2 \cdot \vec{p}_0)\,\vec{p}_0 - (\vec{a}_2 \cdot \vec{p}_1)\,\vec{p}_1 & \vec{p}_2 &= \vec{v}_2/\|\vec{v}_2\|,\\
\vec{v}_3 &= \ldots
\end{aligned} \tag{7.530}$$

[4] Symmetrische positiv definite Matrizen haben orthogonale Eigenvektoren. — Nur wenn die Matrizen nicht *normal* sind, sind die linear unabhängigen Eigenvektoren nicht orthogonal. Wann ist nun ein Matrix A normal? Eine Matrix ist genau dann normal, wenn sie mit der adjungierten Matrix $\mathsf{A}^\dagger$vertauscht. Mathematisch bedeutet das

$$\mathsf{A} \cdot \mathsf{A}^\dagger = \mathsf{A}^\dagger \cdot \mathsf{A}.$$

Die adjungierte Matrix ist definiert als das Transponierte und Konjugiert-komplexe von A. Offenbar sind symmetrische reelle Matrizen immer selbstadjungiert: $\mathsf{A}^\dagger = \mathsf{A}$ und damit auch normal.

Diese Strategie wollen wir jetzt verwenden, wobei wir aber nicht orthogonalisieren, sonder A-orthogonalisieren. Damit ist es möglich, die konjugierten Richtungen, die man nicht *a priori* kennt, im Laufe der Iteration zu generieren.

Als anfängliche Richtung $\vec{p}^{(0)}$ wählt man das anfängliche Residuum: $\vec{p}^{(0)} = \vec{\rho}^{(0)}$. Dann berechnet man $\vec{x}^{(1)}$ gemäß (7.523) wie beim *steepest descent*. Um die neue Richtung $\vec{p}^{(1)}$ zu gewinnen, die A-orthogonal zu $\vec{p}^{(0)}$ ist, oder allgemeiner, um die neue Richtung im Schritt $n+1$ zu bestimmen, geht man vom Gradienten $-\nabla F\left(\vec{x}^{(n+1)}\right) = \vec{\rho}^{(n+1)}$ (siehe (7.517)) aus und setzt an

$$\vec{p}^{(n+1)} = -\nabla F\left(\vec{x}^{(n+1)}\right) + \sum_{j=0}^{n} \beta_{n+1,j}\, \vec{p}^{(j)} = \vec{\rho}^{(n+1)} + \sum_{j=0}^{n} \beta_{n+1,j}\, \vec{p}^{(j)}. \tag{7.531}$$

Zur Bestimmung der Koeffizienten $\beta_{n+1,j}$ verwendet man nun die Forderung, daß die neue Richtung $\vec{p}^{(n+1)}$ bezüglich aller alten Richtungen $\vec{p}^{(l)}$, $l \le n$, A-orthogonal ist. Dazu wird (7.531) mit $\vec{p}^{(l)} \cdot \mathsf{A} \cdot$ von links multipliziert. Wir erhalten so

$$\begin{aligned} 0 \overset{l\le n}{=} \vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{p}^{(n+1)} &= \vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{\rho}^{(n+1)} + \sum_{j=0}^{n} \beta_{n+1,j} \underbrace{\vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{p}^{(j)}}_{\sim \delta_{l,j}} \\ &= \vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{\rho}^{(n+1)} + \beta_{n+1,l}\, \vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{p}^{(l)}. \end{aligned} \tag{7.532}$$

Die Summe reduziert sich hier auf nur einen Summanden, da wir im Sinne einer Induktion annehmen können, daß die $\vec{p}^{(j)}$ für $j = 0, ..., n$ schon A-orthogonal sind. Damit folgt

$$\beta_{n+1,l} = -\frac{\vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{\rho}^{(n+1)}}{\vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{p}^{(l)}}. \tag{7.533}$$

Unter Verwendung von (7.522) und (7.523) kann man nach einiger Rechnung zeigen (Anhang C), daß nur der Koeffizient $\beta_{n+1,n}$ in (7.531) eingeht, also nur der Summand mit $l = n$. Damit ist

$$\vec{p}^{(n+1)} = \vec{\rho}^{(n+1)} + \beta_{n+1,n}\, \vec{p}^{(n)} = \vec{\rho}^{(n+1)} - \frac{\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{\rho}^{(n+1)}}{\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(n)}}\, \vec{p}^{(n)}. \tag{7.534}$$

Deshalb kann man die A-orthogonalen Richtungen durch die einfache Rekursion erhalten

$$\vec{p}^{(0)} = -\nabla F\left(\vec{x}^{(0)}\right) = \vec{\rho}^{(0)}, \tag{7.535a}$$

$$\vec{p}^{(n+1)} = \vec{\rho}^{(n+1)} - \frac{\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{\rho}^{(n+1)}}{\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(n)}}\, \vec{p}^{(n)}. \tag{7.535b}$$

Diese Richtungen werden im Laufe der Iteration (7.523) erzeugt. Wenn man schon alle Richtungen bis $\vec{p}^{(n)}$ iteriert hat, geht in die Berechnung der neuen Suchrichtung $\vec{p}^{(n+1)}$ nur die jeweils letzte Suchrichtung $\vec{p}^{(n)}$ ein, sowie das aktuelle und das letzte Residuum $\vec{\rho}^{(n)}$. Dies wirkt sich natürlich günstig auf die Rechenzeit aus. Wenn $-\nabla F\left(\vec{x}^{(n+1)}\right) = \vec{\rho}^{(n+1)} = 0$ ist, bricht die Iteration ab und man ist am Extremum von F bzw. an der Lösung $\vec{x} = \mathsf{A}^{-1} \cdot \vec{b}$ angelangt.

Krylov-Räume

Die Menge der Suchrichtungen $\{\vec{p}^{(j)}\}$, $j = 0, ..., n-1 \leq N-1$, spannen einen n-dimensionalen Unterraum $\mathcal{V}_n$ des gesamten Lösungsraumes auf. Bei jedem Iterationsschritt wird eine neue Dimension eröffnet. Dies liegt, wie bei der Gram-Schmidt-Orthogonalisierung (7.530), an dem ersten Summanden $\vec{\rho}^{(n+1)}$ in (7.535b). Denn der zweite Summand zeigt ja nur in die alte Richtung $\vec{p}^{(n)}$. Der erste Summand lautet

$$\begin{aligned}\vec{\rho}^{(n+1)} &= \vec{b} - \mathsf{A}\cdot\vec{x}^{(n+1)} = \vec{b} - \mathsf{A}\cdot\left(\vec{x}^{(n)} + \alpha^{(n)}\vec{p}^{(n)}\right) = \underbrace{\vec{b} - \mathsf{A}\cdot\vec{x}^{(n)}}_{\vec{\rho}^{(n)}} - \alpha^{(n)}\mathsf{A}\cdot\vec{p}^{(n)} \\ &= \vec{\rho}^{(n)} - \alpha^{(n)}\mathsf{A}\cdot\vec{p}^{(n)}. \end{aligned} \tag{7.536}$$

Hieran sieht man, daß das Residuum $\vec{\rho}^{(n+1)}$ im Laufe der Iteration bei jedem Schritt $n \to n+1$ in einem größeren Unterraum des $\mathbb{R}^N$ betrachtet wird. Die jeweils neu hinzugekommene Richtung ist durch $\mathsf{A}\cdot\vec{p}^{(n)}$ gegeben. Dasselbe gilt wegen (7.535b) für $\vec{p}^{(n+1)}$. Der erste Summand $\vec{\rho}^{(n)}$ in (7.536) stellt keine neue Richtung dar, denn $\vec{\rho}^{(n)}$ wird durch dieselbe Gleichung (7.536) beschrieben, nur mit einem um 1 reduzierten Index, und es gilt $\vec{p}^{(0)} = \vec{\rho}^{(0)}$. Daher sind $\vec{\rho}^{(n)}$ und $\vec{p}^{(n)}$ in demselben Unterraum des $\mathbb{R}^N$ enthalten. In Tabelle 7.1 sind die Abhängigkeiten skizziert.

Man schreibt nun den n-ten Unterraum $\mathcal{V}_n$ in der folgenden Form auf

$$\begin{aligned}\mathcal{V}_n &= \text{span}\left\{\vec{\rho}^{(0)}, \mathsf{A}\cdot\vec{\rho}^{(0)}, \mathsf{A}^2\cdot\vec{\rho}^{(0)}, ...\mathsf{A}^{n-1}\cdot\vec{\rho}^{(0)}\right\} \\ &= \text{span}\left\{\mathsf{A}^j\vec{\rho}^{(0)} \mid 0 \leq j \leq n-1\right\}. \end{aligned} \tag{7.537}$$

Diese Unterräume sukzessive höherer Dimension nennt man nach dem russischen Schiffsbauingenieur Krylov *Krylov-Unterräume*. Die Vektoren $\mathsf{A}^j\vec{\rho}^{(0)}$ mit $j \leq n$ sind nicht A-orthogonal. Sie spannen aber denselben Unterraum auf, wie die A-orthogonalen Suchrichtungen $\vec{p}^{(j)}$ mit $j \leq n$

$$\begin{aligned}\mathcal{V}_n &= \text{span}\left\{\vec{\rho}^{(0)}, \vec{\rho}^{(1)}, ...\vec{\rho}^{(n)}\right\} \\ &= \text{span}\left\{\vec{\rho}^{(0)}, \mathsf{A}\cdot\vec{\rho}^{(0)}, ...\mathsf{A}^{n-1}\cdot\vec{\rho}^{(0)}\right\} \\ &= \text{span}\left\{\vec{p}^{(0)}, \mathsf{A}\cdot\vec{p}^{(0)}, ...\mathsf{A}^{n-1}\cdot\vec{p}^{(0)}\right\} \end{aligned} \tag{7.538}$$

Tabelle 7.1.: Skizze der entscheidenden Abhängigkeiten von $\vec{p}^{(n+1)}$.

n	$\vec{p}^{(n)}$	neue Richtung in $\vec{p}^{(n+1)}$
0	$\vec{\rho}^{(0)}$	$\vec{p}^{(1)} = \mathsf{A}\cdot\vec{p}^{(0)} = \mathsf{A}\cdot\vec{\rho}^{(0)}$
1	$\vec{\rho}^{(1)} + \left(\sim \vec{p}^{(0)}\right)$	$\vec{p}^{(2)} = \ldots\mathsf{A}\cdot\vec{p}^{(1)}\ldots = \ldots\mathsf{A}\cdot\vec{\rho}^{(1)}\ldots$
$\vdots$		
n	$\vec{\rho}^{(n-1)} + \ldots$	$\vec{p}^{(n+1)} = \ldots\mathsf{A}\cdot\vec{p}^{(n)}\ldots = \ldots\mathsf{A}\cdot\vec{\rho}^{(n)}\ldots$

Dies kann man mittels Induktion und (7.536) zeigen: Angenommen es gilt

$$\vec{\rho}^{(n)}, \vec{p}^{(n)} \in \text{span}\left\{\vec{\rho}^{(0)}, \mathsf{A}\cdot\vec{\rho}^{(0)}, \ldots \mathsf{A}^{n-1}\cdot\vec{\rho}^{(0)}\right\}, \tag{7.539}$$

dann folgt nach (7.536)

$$\vec{\rho}^{(n+1)} \in \text{span}\left\{\vec{\rho}^{(0)}, \mathsf{A}\cdot\vec{\rho}^{(0)}, \ldots \mathsf{A}^{n}\cdot\vec{\rho}^{(0)}\right\}, \tag{7.540}$$

und dann ist auch $\vec{p}^{(n+1)}$ aus diesem Unterraum.

Im Laufe der Iteration wird das Residuum daher in Krylov-Unterräumen mit wachsender Dimension $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_N$ zum Verschwinden gebracht. Wegen (7.526) bleiben diejenigen Komponenten des Residuums, die in einem Schritt eliminiert wurden, auch weiterhin Null, wenn man weiteriteriert und zum nächsthöheren Krylov-Unterraum übergeht. Die Iteration bricht damit nach höchstens N Schritten ab. Dann ist $\vec{\rho}^{(n)} = 0$.

CG-Algorithmus und Abwandlungen

Damit stellt sich der Algorithmus der konjugierten Gradienten folgendermaßen dar:[56]

CG-Algorithmus:

SELECT $\vec{x}^{(0)}$ (Startwert)

SET $\vec{p}^{(0)} = \vec{\rho}^{(0)} = \vec{b} - \mathsf{A}\cdot\vec{x}^{(0)}$ (1. Suchrichtung)

LOOP: $n = 0, 1, \ldots$

$$\alpha^{(n)} = \frac{\vec{p}^{(n)}\cdot\vec{\rho}^{(n)}}{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)}} = \frac{\vec{\rho}^{(n)}\cdot\vec{\rho}^{(n)}}{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)}} \quad \text{(Schrittweite) (7.522)}$$

$$\vec{x}^{(n+1)} = \vec{x}^{(n)} + \alpha^{(n)}\vec{p}^{(n)} \quad \text{(Iteration) (7.523)}$$

$$\vec{\rho}^{(n+1)} = \vec{\rho}^{(n)} - \alpha^{(n)}\mathsf{A}\cdot\vec{p}^{(n)} \quad \text{(Residuum)}$$

IF $|\vec{\rho}^{(n+1)}|^2 < \epsilon$ STOP

$$\beta^{(n)} = -\frac{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{\rho}^{(n+1)}}{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)}} = \frac{\vec{\rho}^{(n+1)}\cdot\vec{\rho}^{(n+1)}}{\vec{\rho}^{(n)}\cdot\vec{\rho}^{(n)}} \quad \text{(Koeffizient) (7.533)}$$

$$\vec{p}^{(n+1)} = \vec{\rho}^{(n+1)} + \beta^{(n)}\vec{p}^{(n)} \quad \text{(neue Richtung) (7.535b)}$$

[5] Das CG-Verfahren geht zurück auf Hestenes and Stiefel (1952). Es ist einer der wichtigsten Algorithmen der numerischen linearen Algebra.

[6] Man kann zeigen

$$\frac{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{\rho}^{(n+1)}}{\vec{p}^{(n)}\cdot\mathsf{A}\cdot\vec{p}^{(n)}} = -\frac{\vec{\rho}^{(n+1)}\cdot\vec{\rho}^{(n+1)}}{\vec{\rho}^{(n)}\cdot\vec{\rho}^{(n)}}.$$

Es ist

$$\vec{p}^{(n)} \cdot \vec{\rho}^{(n)} = \left(\vec{\rho}^{(n)} - \beta^{(n-1)}\vec{p}^{(n-1)}\right) \cdot \vec{\rho}^{(n)} = \vec{\rho}^{(n)} \cdot \vec{\rho}^{(n)} - \beta^{(n-1)} \underbrace{\vec{p}^{(n-1)} \cdot \vec{\rho}^{(n)}}_{=0}. \quad (7.541)$$

Da das n-te Residuum $\vec{\rho}^{(n)}$ keine Komponente mehr in Richtung der früheren Suchrichtungen (insbesondere $\vec{p}^{(n-1)}$) besitzt (vgl. 7.525), verschwindet das zweite Skalarprodukt. Damit gilt das blaue Gleichheitszeichen (=) in dem obigen Algorithmus.

Da die Gleichungen in der Strömungsmechanik im allgemeinen nicht symmetrisch sind (bis auf die Poisson-Gleichung für den Druck oder im Limes der reinen Wärmeleitung), müssen die Gleichungen symmetrisiert werden. Das kann man erreichen, indem man das doppelt so große System betrachtet

$$\begin{pmatrix} 0 & \mathsf{A} \\ \mathsf{A}^{\mathrm{T}} & 0 \end{pmatrix} \cdot \begin{pmatrix} \vec{y} \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}, \quad (7.542)$$

dessen Matrix *per constructionem* symmetrisch ist. Die Lösung $\vec{y}$ ist dabei irrelevant. Wenn man dieses System analog zum CG-Verfahren behandelt, erhält man das sogenannte *Bi-konjugierte-Gradienten-Verfahren* (Bi-CG).

Es gibt einen ganzen Zoo verschiedener Varianten des CG-Verfahrens, die alle zu den *Krylov-Unterraum-Verfahren* zählen. Als Gleichungslöser erwähnenswert ist noch das *GMRES-Verfahren* (***g**eneralized **m**inimal **res**iduum*), für welches die Matrix A nicht positive definit sein muß (Golub and van Loan, 1989; Saad, 2003).

Alexei Nikolajewitsch Krylow 1863–1945

Bei allen Krylov-Unterraum-Verfahren werden nur Matrix-Vektor-Multiplikationen und Skalarprodukte benötigt. Bei dünnbesetzten Matrizen kostet damit eine Matrix-Vektor-Multiplikation und auch ein Iterationsschritt $O(N)$ Operationen. Wenn man nur sehr wenige Iterationen benötigt, ist das Verfahren immer noch $O(N)$, jedoch mit einem großen Vorfaktor (Anzahl der Iterationen). Daher eignen sich diese Verfahren nur für große dünnbesetze Matrizen (ab ca. 10^4 Unbekannten).

Damit man eine gute Näherung nach möglichst wenigen Iterationen erhält, ist es von wesentlicher Bedeutung, das Problem in eine geeignete Form zu transformieren. Dies ist die Aufgabe der *Vorkonditionierung*. Bei einer guten Vorkonditionierung wird das Residuum zuerst bezüglich derjenigen Richtungen minimiert (d.h. eliminiert), die einen wesentlichen Beitrag zum Residuum liefern. Bei einer linearen Vorkonditionierung wird das Gleichungssystem von links mit einer Matrix multipliziert, die A^{-1} möglichst gut approximieren sollte. Im Prinzip kann das von jedem iterativen Verfahren wie dem Jacobi- oder dem GS-Verfahren geleistet werden. Beim Krylow-Unterraum-Verfahren (CG-Verfahren) ist es günstig, wenn die Matrix A eine kleine Konditionszahl besitzt (siehe Kap. 2.1.4), bzw. eine *gute* Verteilung der Eigenwerte.

8. Ergänzende Themen

8.1. Ausblick: Partielle Differentialgleichungen

Die meisten raum-zeitlichen Vorgänge in Natur und Technik werden in kompakter Weise durch partielle Differentialgleichungen beschrieben. Die Vielfalt der Phenomene spiegelt sich in der Struktur der Differentialgleichungen und der erforderlichen Lösungsstrategie wieder, wodurch das Gebiet sehr umfangreich ist. Daher soll in diesem Kapitel nur ein kurzes Schlaglicht auf die numerische Behandlung partieller Differentialgleichungen geworfen werden.

8.1.1. Allgemeine Bemerkungen

Partielle Differentialgleichungen sind Differentialgleichungen in mindestens zwei unabhängigen Variablen. Beispiele sind

$$\frac{\partial T}{\partial t} - \kappa \frac{\partial^2 T}{\partial x^2} = 0, \qquad \text{(1D-Wärmeleitungsgleichung)} \tag{8.543a}$$

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0, \qquad \text{(2D-Potentialgleichung)} \tag{8.543b}$$

$$\frac{\partial^2 H}{\partial t^2} - c^2 \frac{\partial^2 H}{\partial x^2} = 0. \qquad \text{(1D-Wellengleichung)} \tag{8.543c}$$

Ein etwas komplizierteres Beispiel ist die inkompressible *Navier-Stokes-* und Kontinuitätsgleichung

$$\frac{\partial u}{\partial t} + \left(u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z} \right) u = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) u, \tag{8.544a}$$

$$\frac{\partial v}{\partial t} + \left(u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z} \right) v = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) v, \tag{8.544b}$$

$$\frac{\partial w}{\partial t} + \left(u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z} \right) w = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) z, \tag{8.544c}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0. \tag{8.544d}$$

für das Geschwindigkeitsfeld $\vec{u}(x, y, z, t) = u\vec{e}_x + v\vec{e}_y + w\vec{e}_z$ und das Druckfeld $p(x, y, z, t)$ in bewegten fluiden Medien dar. Hierbei sind die einzelnen Geschwindigkeitskomponenten miteinander gekoppelt. Außerdem taucht der Druck als Variable auf, der so bestimmt werden muß, daß die Massenerhaltung gewährleistet ist.

Die meisten nichtlinearen partiellen Differentialgleichungen können nicht mit analytischen Methoden gelöst werden. D.h., man kann ihre Lösungen nicht in geschlossener Form angeben. Daher ist man auf numerische Näherungslösungen angewiesen. Eine weitere Komplikation besteht darin, daß die Eigenschaften partieller Differentialgleichungen sehr unterschiedlich sein können, weshalb verschiedenen Gleichungen i.a. unterschiedliche numerische Methoden erfordern. Man kann partielle Differentialgleichungen in drei Typen einteilen. Sie werden klassifiziert als

- hyperbolisch,
- parabolisch, oder
- elliptisch.

Hyperbolische Gleichungen wie (8.543c) beschreiben wellenartige Vorgänge. Eine Störung an einem bestimmten Raumpunkt breitet sich mit der endlichen Geschwindigkeit c im Volumen aus. *Elliptische* Gleichungen wie (8.543b) beschreiben Gleichgewichtszustände. Hierbei wirkt sich eine Änderung an einem bestimmten Raumpunkt ohne Verzögerung auf die Lösung an sämtlichen Punkten des Raumes aus, wie zum Beispiel bei der Deformation einer belasteten Platte. *Parabolische* Gleichungen wie (8.543a) stellen gewissermaßen den Grenzbereich zwischen hyperbolischen und elliptischen partiellen Differentialgleichungen dar. Sie beschreiben Ausgleichsvorgänge.

Das Gebiet der partiellen Differentialgleichungen ist sehr umfangreich und reichhaltig. Für sehr viele Fälle existieren leider keine mathematischen Beweise über die Existenz und/oder die Anzahl von Lösungen. Dies gilt im besonderen Maße für nichtlineare partielle Differentialgleichungen.

Zur Lösung partieller Differentialgleichungen muß man kompatible Rand- und Anfangsbedingungen zur Verfügung stellen. Manchmal sind auch noch gewisse physikalische Forderungen sicherzustellen, etwa die Inkompressibilität des Fluid (Massenerhaltung) in (8.544d), die selbst als weitere Differentialgleichung formuliert ist, oder die Positivität einer Größe wie z.B. der Filmdicke bei der Dynamik von sich ausbreitenden dünnen Filmen.

Die numerische Lösung von partiellen Differentialgleichungen gliedert sich in zwei wesentliche Schritte,

1. die Diskretisierung und
2. die Lösung eines (großen) algebraischen Gleichungssystems.

Für den zweiten Schritt (2) hatte wir elementare Methoden in Kap. 2 diskutiert. An dieser Stelle sei bemerkt, daß effiziente Algorithmen umso wichtiger werden, je größer die Anzahl der Unbekannten ist. Je größer ein Problem ist, das numerisch behandelt werden kann, desto wichtiger ist es, effiziente Algorithmen zu verwenden, denn der Vorteil in der Anzahl der zur Lösung erforderlichen Operationen skaliert mit $O(N^{\alpha-\beta})$, wobei N die Anzahl der Unbekannten ist, und α und β die Exponenten des Grundalgorithmus bzw. des effizienteren Algorithmus. Für den ersten

Schritt (1) gibt es sehr viele verschiedene Varianten. Hier ist im wesentlichen die Diskretisierung im Raum zu nennen. Beispiele sind

- finite Differenzen,
- finite Volumen,
- finite Elemente,
- Randelemente,
- spektrale Methoden,
- pseudo-spektrale Methoden,
- gitterfreie Methoden (vortex methods),
- Gitter-Boltzmann-Methoden
- u.a.m.

8.1.2. Beispiel: Eindimensionale Wärmeleitungsgleichung

Am Beispiel der parabolischen eindimensionalen Wärmeleitungsgleichung (8.543a) auf einem Raumgebiet $x \in [0,1]$ wollen wir einige elementare Methoden und Befunde illustrieren. Um eine Diskretisierung in finiten Differenzen zu erhalten, überziehen wir das (x,t)-Gebiet mit einem homogenen Gitternetz (Abb. 8.1). Anstelle einer Betrachtung der unendlich vielen Werte der Temperatur an allen Punkten in einem Intervall $x \in [0,1]$ und zu allen Zeitpunkten $t \in [0,\infty]$ aus dem halbunendlichen Streifen $(x,t) \in [0,1] \times [0,\infty]$ betrachten wir nur die Temperaturwerte an den diskreten Gitterpunkten. Seien diese Punkte gegeben durch

$$x \longrightarrow x_j = j\Delta x, \qquad j \in [0,J], \qquad \Delta x = 1/J, \tag{8.545a}$$

$$t \longrightarrow t_n = n\Delta t, \qquad n \in [0,\infty]. \tag{8.545b}$$

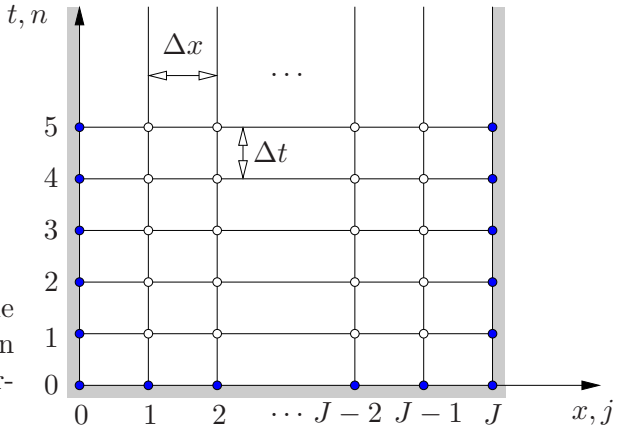


Abbildung 8.1.: Äquidistantes Gitter. Die Funktionswerte an den blauen Punkten werden durch die Randbedingungen vorgegeben.

Ein explizites Verfahren: FTCS

Um von den kontinuierlichen Variablen in (8.543a) zu einer Gleichung für die diskreten Variablen $T_j^n = T(x_j, t_n)$ zu kommen, betrachten wir stellvertretend für alle Gitterpunkte den Punkt (n, j) und entwickeln die Funktion T an den umgebenden Punkten in eine Taylor-Reihe um den Entwicklungspunkt $(x_j, t_n) = (j, n)$. Dann erhalten wir

$$T(x_j, t_{n+1}) = T_j^{n+1} = T_j^n + \Delta t \left(\frac{\partial T}{\partial t}\right)_j^n + \frac{\Delta t^2}{2}\left(\frac{\partial^2 T}{\partial t^2}\right)_j^n + O(\Delta t^3). \tag{8.546}$$

Wenn wir die Gleichung nach $\partial^2 T/\partial t^2$ auflösen, erhalten wir für die erste Zeitableitung

$$\left(\frac{\partial T}{\partial t}\right)_j^n = \frac{T_j^{n+1} - T_j^n}{\Delta t} - \underbrace{\frac{\Delta t}{2}\left(\frac{\partial^2 T}{\partial t^2}\right)_j^n + O(\Delta t^2)}_{O(\Delta t)}. \tag{8.547}$$

Wenn wir die Zeitableitung durch den ersten Summanden approximieren, machen wir demnach einen Fehler der Größenordnung $O(\Delta t)$.[1]

In ähnlicher Weise können wir die zweite räumliche Ableitung bilden. Dazu entwickeln wir T_{j-1}^n und T_{j+1}^n

$$T_{j-1}^n = T_j^n - \Delta x \left(\frac{\partial T}{\partial x}\right)_j^n + \frac{\Delta x^2}{2}\left(\frac{\partial^2 T}{\partial x^2}\right)_j^n - O(\Delta t^3), \tag{8.548a}$$

$$T_{j+1}^n = T_j^n + \Delta x \left(\frac{\partial T}{\partial x}\right)_j^n + \frac{\Delta x^2}{2}\left(\frac{\partial^2 T}{\partial x^2}\right)_j^n + O(\Delta t^3). \tag{8.548b}$$

Wenn wir die Summe beider Gleichungen bilden, fällt die erste Ableitung heraus und wir können nach der zweiten Ableitung auflösen.

$$T_{j+1}^n + T_{j-1}^n - 2T_j^n = \Delta x^2 \left(\frac{\partial^2 T}{\partial x^2}\right)_j^n + O(\Delta x^4), \tag{8.549}$$

was auf

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{\Delta x^2} + O(\Delta x^2) \tag{8.550}$$

führt. Wenn wir nun die kontinuierliche zweite Ableitung durch die diskrete ersetzen machen wir einen Fehler der Größe $O(\Delta x^2)$. Wir erhalten damit den sogenannten *FTCS-Algorithmus* (***f**orward in **t**ime and **c**entered in **s**pace*) der eindimensionalen Wärmeleitungsgleichung

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} - \kappa \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{\Delta x^2} = 0. \tag{8.551}$$

[1] Wir gehen davon aus, daß die Vorfaktoren (exakten Ableitungen) von der Größenordnung eins sind.

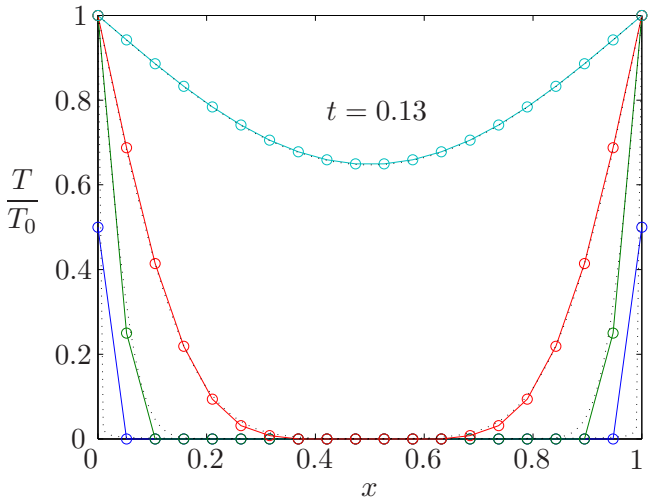


Abbildung 8.2.: Lösung der transienten eindimensionalen Wärmeleitung mittels FTCS-Schema für $s = 0.50$ und $J = 20$ ($\Delta x = 1/19$). Es wurde $\kappa = 1$ gesetzt. Blau: Anfangsbedingung $t = 0$, grün: $t = \Delta t$, rot: $t = 6\Delta t$ und cyan: $t = 94\Delta t \approx 0.13$. Die gepunkteten Linien stellen die *nahezu exakten* Lösungen dar.

Dieses Verfahren ist von der Genauigkeit $O(\Delta t, \Delta x^2)$. In diese Gleichung geht einzig der Parameter $s = \kappa\Delta t/\Delta x^2$ ein. Er ist ein Maß für die Größe des Zeitschrittes. Man sieht: Wenn man die Temperatur an allen Raumpunkten zu einem gewissen Zeitpunkt n kennt, kann man die Temperatur zu jedem neuen Zeitpunkt $n + 1$ explizit angeben, denn man kann die Gleichung nach T_j^{n+1} auflösen. Da man für $t = 0$ ohnehin eine Anfangsverteilung $T(x, t = 0) = T_j^0$ für $j = 0, \ldots, J$ vorgeben muß, kann man das gesamte Temperaturfeld Schritt für Schritt explizit berechnen. In die Werte T_1^n und T_{J-1}^n in Randnähe gehen die Randwerte T_0^n bzw. T_J^n ein. Wenn man diese vorgibt, entspricht dies einer vorgegebenen Randtemperatur. Experimentell ließe sich dies durch temperierte Kupferblöcke erreichen. Ein Zeitschritt würde bei diesem expliziten Verfahren $O(J)$ Operationen kosten. Billiger geht es nicht.

Wenn man das Verfahren (8.551) implementiert, kann man das in Abb. 8.2 oder 8.3 gezeigte Verhalten finden. Bei genauerer Analyse findet man, daß der FTCS-Algorithmus für $s > 0.5$ nicht funktioniert und nicht-physikalische Oszillationen mit explodierender Amplitude liefert. Es handelt sich hierbei um eine *numerische Instabilität*. Die Stabilitätsbedingung $s \leq 1/2$ schränkt die realisierbare zeitliche Schrittweite ein, was zu einem höheren numerischen Aufwand führt. Eine Verdopplung der räumlichen Gitterpunkte erfordert dann eine Vervierfachung der erforderlichen Zeitschritte, um die Berechnung bis zu einer vorgegebenen Zeit durchzuführen. Generell ist die Frage nach der Stabilität eines Algorithmus wesentlich. Es gibt Methoden, mit deren Hilfe man die Stabilität von Algorithmen für lineare oder linearisierte Gleichungen systematisch untersuchen kann (siehe Vorlesung über *Numerische Methoden der Strömungs- und Wärmetechnik*, LVA-Nr. 302.017).

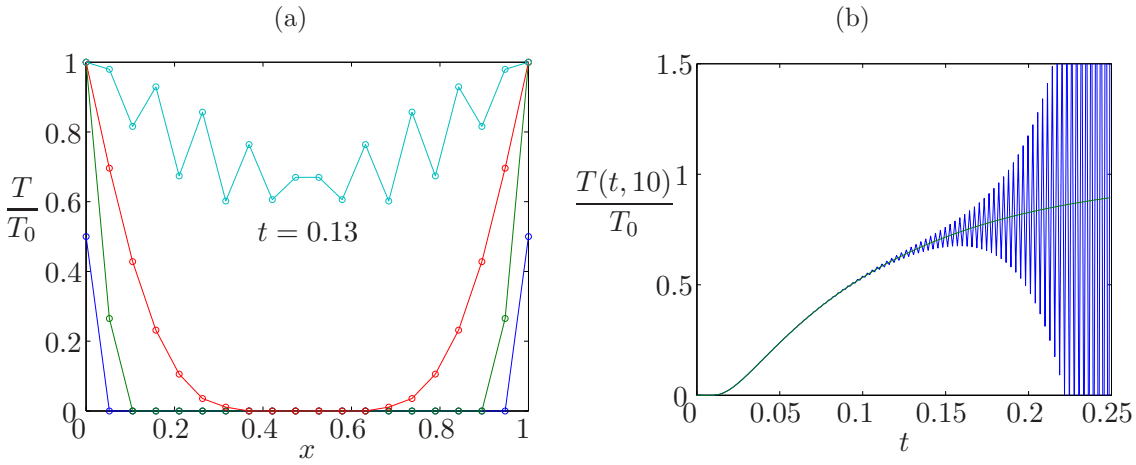


Abbildung 8.3.: Instabilität des FTCS-Algorithmus für die zeitabhängige eindimensionale Wärmeleitung für $s = 0.53$ ($J = 20$). (a) zeigt die räumliche Abhängigkeit der Lösung (vgl. mit Abb. 8.2 für das stabile Verfahren mit $s = 0.50$) und (b) zeigt die zeitliche Entwicklung am Punkt $j = 10$ (blau) im Vergleich zur stabilen Lösung (grün, $s = 0.5$).

Eine einfache implizite Diskretisierung

Eine andere Möglichkeit der Diskretisierung mit finiten Differenzen besteht darin, die zweite räumliche Ableitung nicht zum Zeitpunkt n sondern zum Zeitpunkt $n+1$ auszuwerten. Dann erhält man den Algorithmus

$$T_j^{n+1} - T_j^n - s\left(T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}\right) = 0, \tag{8.552}$$

oder

$$-sT_{j+1}^{n+1} + (1+2s)T_j^{n+1} - sT_{j-1}^{n+1} = T_j^n. \tag{8.553}$$

Hierbei wurden die Terms so sortiert, daß alle Unbekannten (Zeitpunkt $n+1$) auf der linken Seite und alle bekannten Größen (Zeitpunkt n und früher) auf der rechten Seite der Gleichung stehen. Dieses implizite Verfahren ist aufwendiger als das explizite, weil man in jedem Zeitschritt ein Gleichungssystem lösen muß. Die Matrix ist allerdings tridiagonal. Damit kann man den Thomas-Algorithmus zur schnellen Lösung verwenden.

Das in jedem Schritt zu lösenden Gleichungssystem hat die folgende Form

$$\begin{pmatrix} (1+2s) & -s & & & \\ -s & (1+2s) & -s & & \\ & \ddots & \ddots & \ddots & \\ & & -s & (1+2s) & -s \\ & & & s & (1+2s) \end{pmatrix} \cdot \begin{pmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \vdots \\ T_{J-2}^{n+1} \\ T_{J-1}^{n+1} \end{pmatrix} = \begin{pmatrix} T_1^n + sT_0^{n+1} \\ T_2^n \\ \vdots \\ T_{J-2}^n \\ T_{J-1}^n + sT_J^{n+1} \end{pmatrix}. \tag{8.554}$$

Dies ist ein Gleichungssystem nur für die inneren Punkte. Die roten Einträge stammen von den als vorgeben betrachteten Randbedingungen.

Man kann zeigen, daß das implizite Verfahren (8.553) uneingeschränkt stabil ist. D.h. man kann beliebig große Schrittweiten Δt verwenden, ohne daß unphysikalische Störungen exponentiell mit der Zeit anwachsen. Wenn man nur an der asymptotischen Lösung für $t \to \infty$ interessiert ist, kann man sehr große Schrittweiten wählen. Ist man jedoch an dem genauen zeitlichen Verhalten interessiert, darf der Zeitschritt nicht zu groß gewählt werden, da ansonsten der Diskretisierungsfehler zu groß wird.

Wärmeleitung in mehr als einer Dimension

Die Wärmeleitungsgleichung lautet in zwei Raumdimensionen

$$\frac{\partial T}{\partial t} = \kappa \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) T. \tag{8.555}$$

Wenn wir nun die Ableitung in y-Richtung in Analogie zur x-Richtung (8.550) bilden, erhalten wir

$$\frac{T_{j,k}^{n+1} - T_{j,k}^n}{\Delta t} = \kappa \left(\frac{T_{j+1,k}^n - 2T_{j,k}^n + T_{j-1,k}^n}{\Delta x^2} + \frac{T_{j,k+1}^n - 2T_{j,k}^n + T_{j,k-1}^n}{\Delta y^2} \right) = 0. \tag{8.556}$$

Dies ist wieder ein explizites Verfahren. Man kann zeigen, daß dieses Verfahren stabil ist, solange

$$s_1 + s_2 \leq \frac{1}{2}, \tag{8.557}$$

wobei $s_1 = \kappa \Delta t / \Delta x^2$ und $s_2 = \kappa \Delta t / \Delta y^2$. Die Zeitschrittweite ist also gegenüber dem eindimensionalen Fall reduziert. In drei Dimensionen findet man eine weitere Reduktion mit der Bedingung $s \leq 1/6$ ($s_1 = s_2 = s_3$).

Ein stabileres implizites Verfahren lautet (es sei $s = s_1 = s_2$)

$$T_{j,k}^{n+1} - T_{j,k}^n = s \left(T_{j+1,k}^{n+1} - 2T_{j,k}^{n+1} + T_{j-1,k}^{n+1} + T_{j,k+1}^{n+1} - 2T_{j,k}^{n+1} + T_{j,k-1}^{n+1} \right) = 0. \tag{8.558}$$

Wenn man die Unbekannten wie in Abb. 2.2 in der SW-Ecke beginnend bis in die NO-Ecke durchnumeriert, erhält man ein lineares Problem mit der pentadiagonalen Matrix wie in Abb. 2.3 illustriert.

Alternativ zur Lösung des pentadiagonalen Systems, kann man versuchen, die relativ starke Kopplung der Variablen zu reduzieren. Dazu kürzen wir die diskrete zweite Ableitung ab mit

$$\frac{\delta^2 T_{j,k}}{\delta x^2} = \frac{T_{j+1,k} - 2T_{j,k} + T_{j-1,k}}{\Delta x^2}. \tag{8.559}$$

Die zweite Ableitung $\delta^2 T_{j,k} / \delta y^2$ wird entsprechend definiert. Dann betrachten wir die halb-implizite Formulierung

$$\left(1 - \frac{\kappa \Delta t}{2} \frac{\delta^2}{\delta x^2} - \frac{\kappa \Delta t}{2} \frac{\delta^2}{\delta y^2} \right) T_{j,k}^{n+1} = \left(1 + \frac{\kappa \Delta t}{2} \frac{\delta^2}{\delta x^2} + \frac{\kappa \Delta t}{2} \frac{\delta^2}{\delta y^2} \right) T_{j,k}^n. \tag{8.560}$$

Wenn man nun die beiden Klammerausdrücke quadratisch ergänzt, erhält man

$$\begin{aligned}&\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)T_{j,k}^{n+1} \\ =&\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)T_{j,k}^{n}+\frac{(\kappa\Delta t)^2}{4}\frac{\delta^2}{\delta x^2}\frac{\delta^2}{\delta y^2}\left(T_{j,k}^{n+1}-T_{j,k}^{n}\right).\end{aligned} \tag{8.561}$$

Mit $T_{j,k}^{n+1}-T_{j,k}^{n}\sim\Delta t$ ist der letzte Summand $\sim\Delta t^3$ und kann für hinreichend kleine Werte von Δt vernachlässigt werden. Dann lautet das Problem

$$\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\underbrace{\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)T_{j,k}^{n+1}}_{=\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)T_{j,k}^{*}}=\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\underbrace{\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)T_{j,k}^{n}}_{=\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)T_{j,k}^{*}}. \tag{8.562}$$

Wenn man dann einen Zwischenzustand T^* definiert und die durch die Unterklammerung definierten Relationen fordert, ist die Gleichung offenbar identisch gelöst. Die beiden Forderungen lauten

$$\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)T_{j,k}^{*}=\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)T_{j,k}^{n} \tag{8.563a}$$

$$\left(1-\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)T_{j,k}^{n+1}=\left(1+\frac{\kappa\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)T_{j,k}^{*} \tag{8.563b}$$

Um sie zu erfüllen, muß man zwei Teilschritt berechnen, wobei $T_{j,k}^{*}$ eine Zwischenlösung darstellt. Der Vorteil dieser *Splitting-Methode* liegt darin, daß in jedem Teilschritt nur tridiagonale Matrizen zu lösen sind, im Gegensatz zur pentadiagonalen Matrix im Ausgangsproblem. In jedem Teilschritt wird nur eine Raumrichtung implizit behandelt. Die beiden Raumrichtungen werden also alternierend implizit behandelt. Diese recht populäre Methode wird daher auch *ADI-Methode* (***a**lternating **d**irections **i**mplicit*) genannt. Sie geht zurück auf Peaceman and Rachford Jr. (1955).

Diese ADI-Methode läßt sich auch auf drei Dimensionen erweitern. Dann werden insgesamt drei Teilschritte benötigt, wobei in jedem Teilschritt nur jeweils eine Raumrichtung implizit behandelt wird.

8.2. Fourier-Transformationen

Motivation: Lösen von linearen Differentialgleichungen, Signalanalyse, Numerische Lösung von partiellen Differentialgleichungen

Da die schnellen Transformationen zwischen Orts- und Spektralraum eine Schlüsselstellung einnehmen, soll beispielhaft die schnelle Fourier-Transformation behandelt werden. Entsprechende schnelle Transformationen gibt es auch für andere Funktionensysteme, insbesondere auch für Chebyshev-Polynome, die im nächsten Unterkapitel vorgestellt werden.

8.2.1. Kontinuierliche Fouriertransformation

Sei $f(x)$ eine Funktion auf der reellen Achse $-\infty < x < \infty$. Dann definiert man die *Fourier-Transformation* als Abbildung $f(x) \to \hat{f}(k)$

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) \mathrm{e}^{\mathrm{i}kx} \mathrm{d}x. \tag{8.564}$$

Die Umkehr-Abbildung ist die *inverse Fourier-Transformation*

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) \mathrm{e}^{-\mathrm{i}kx} \mathrm{d}k. \tag{8.565}$$

Dies kann man durch Einsetzen überprüfen, denn es ist

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) \mathrm{e}^{-\mathrm{i}kx} \mathrm{d}k = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(x') \mathrm{e}^{\mathrm{i}kx'} \mathrm{d}x' \right) \mathrm{e}^{-\mathrm{i}kx} \mathrm{d}k \tag{8.566}$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} \underbrace{\left(\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}k(x'-x)} \mathrm{d}k \right)}_{2\pi\delta(x'-x)} f(x') \mathrm{d}x' = \int_{-\infty}^{\infty} \delta(x'-x) f(x') \mathrm{d}x' = f(x). \tag{8.567}$$

Man nennt $\hat{f}(k)$ das *Spektrum* von $f(x)$, wobei $k \in \mathbb{R}$ *Wellenzahl* heißt.

Die Fourier-Transformation ist eine lineare Abbildung. Ist $\hat{f}_1$ die FT von f_1, dann ist $\hat{f}_1 + \hat{f}_2$ die FT von $f_1 + f_2$. Beispiel: Es ein $f(x) = A\cos(bx)$. Dann lautet die FT

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{A}{2} \left(\mathrm{e}^{\mathrm{i}bx} + \mathrm{e}^{-\mathrm{i}bx} \right) \mathrm{e}^{\mathrm{i}kx} \mathrm{d}x \tag{8.568}$$

$$= \frac{A}{\sqrt{4\pi}} \left(\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(k+b)x} \mathrm{d}x + \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(k-b)x} \mathrm{d}x \right) \tag{8.569}$$

$$= \frac{A}{2} \left[\delta(k+b) + \delta(k-b) \right]. \tag{8.570}$$

Das Spektrum besitzt δ-Peaks bei $k = -b$ und bei $k = b$.

Ist die Funktion $f(x)$ 2π-periodisch, dann ....

8.2.2. Diskrete Fouriertransformation

Die schnelle Fourier-Transformation (FFT) basiert auf der *diskreten Fourier-Transformation (DFT)*. Wir betrachten eine 2π-periodische Funktion $u(x)$, die an N äquidistanten Punkten x_j ausgewertet (ge*sampled*) wird.[2] Die Intervallänge ist dann $\Delta x = 2\pi/N$ und die Stützpunkte befinden sich bei

$$x_j = \frac{2\pi j}{N}, \qquad \text{mit} \quad j = 1, ..., N. \tag{8.571}$$

[2] hk: Und was ist im allgemeinen Fall?

Die Werte an den Stützstellen sind

$$u_j = u(x_j). \tag{8.572}$$

Dann approximieren wir die Funktion $u(x)$ durch die diskrete *Fourier-Reihe*[3]

$$u_j = u(x_j) = \sum_{k=-K}^{K} \hat{u}_k \mathrm{e}^{\mathrm{i}kx_j}, \quad j = 1, ..., N. \tag{8.573}$$

Diese N Gleichungen bieten Informationen zur Bestimmung von $N = 2K+1$ unbekannten Amplituden $\hat{u}_k$. Sie sind gegeben durch die *diskrete Fourier-Transformation*

$$\hat{u}_k = \frac{1}{N} \sum_{m=1}^{N} u_m \mathrm{e}^{-\mathrm{i}kx_m}, \quad k = -K, ..., K. \tag{8.574}$$

Um diese Relation beweisen zu können, benötigen wir die *diskrete Orthogonalitätsrelation* für die harmonischen Funktionen $\mathrm{e}^{\mathrm{i}kx_j}$ des Fourier-Systems

$$\sum_{j=1}^{N} \mathrm{e}^{-\mathrm{i}m(2\pi j/N)} \mathrm{e}^{\mathrm{i}k(2\pi j/N)} = \sum_{j=1}^{N} \mathrm{e}^{\mathrm{i}(k-m)2\pi j/N} = \begin{cases} N, & \text{falls } k-m = nN,\ n \in \mathbb{Z} \\ 0, & \text{sonst.} \end{cases} \tag{8.575}$$

Diese Orthogonalitätsrelation kann man sich leicht klarmachen. Denn wenn $k-m = nN$ ist, dann ist der Exponent ein ganzzahliges Vielfaches von $2\pi\mathrm{i}$ und damit lauten alle Summanden $\mathrm{e}^{\mathrm{i}nj2\pi} = 1$. Der Summand mit Wert 1 taucht dann genau N-mal auf. Falls jedoch $k-m = q \in \mathbb{Z} \neq nN$ ist, dann liegen die Summanden $\mathrm{e}^{\mathrm{i}q2\pi(j/N)}$ für $j = 1, ..., N$ gleichmäßig verteilt auf dem Einheitskreis in der komplexen Ebene. Die komplexen Zeiger überstreichen dabei den Winkelbereich von $2\pi(q/N)$ bis $2\pi q$ mit dem Inkrement $2\pi(q/N)$. Der Einheitskreis wird also q-mal überstrichen. Wegen der Gleichverteilung der Zeiger kompensieren sie sich zu Null.

Die *Orthogonalitätsrelation* (8.575) können wir nun verwenden, um die diskrete Fourier-Transformation zu beweisen. Wenn wir (8.574) in (8.573) einsetzen, erhalten wir[4]

$$\begin{aligned} u_j &= \sum_{k=-K}^{K} \left(\frac{1}{N} \sum_{m=1}^{N} u_m \mathrm{e}^{-\mathrm{i}kx_m} \right) \mathrm{e}^{\mathrm{i}kx_j} = \frac{1}{N} \sum_{m=1}^{N} u_m \sum_{k=-K}^{K} \mathrm{e}^{\mathrm{i}k(x_j - x_m)} \\ &= \frac{1}{N} \sum_{m=1}^{N} u_m \underbrace{\sum_{k=-K}^{K} \mathrm{e}^{2\pi\mathrm{i}k(j-m)/N}}_{=N\delta_{j,m},\ (8.575)} = \sum_{m=1}^{N} u_m \delta_{j,m} = u_j. \end{aligned} \tag{8.576}$$

[3] Beachte, daß man die Summationsbereiche $k = -K, ..., K$ und $j = 1, ..., N$ beliebig verschieben kann, wenn die Funktionswerte u_j periodisch in j fortgesetzt werden (mit Periode N), was wir hier annehmen. Denn der Faktor $\mathrm{e}^{\mathrm{i}kx_j} = \mathrm{e}^{\mathrm{i}kj2\pi/N}$ ist periodisch in k mit Periode N, weil $k \to k+N$ nur den Zusatzfaktor $\mathrm{e}^{\mathrm{i}j2\pi} = 1$ liefert. Außerdem ist auch die Fourier-Transformierte $\hat{u}_k$ N-periodisch in k (siehe (8.574)).

[4] Beachte, daß der Summationsbereich in der Orthogonalitätsrelation keine Rolle spielt, solange er über N zusammenhängende ganze Zahlen geht.

Damit haben wir bewiesen, daß (8.574) die Umkehrung von (8.573) ist.

Wenn $u_j \in \mathbb{R}$ reell ist, dann ist $u_j = u_j^*$ und es gilt

$$\hat{u}_{-k} = \frac{1}{N}\sum_{m=1}^{N} u_m \mathrm{e}^{+\mathrm{i}kx_m} \overset{u_m=u_m^*}{=} \left(\frac{1}{N}\sum_{m=1}^{N} u_m \mathrm{e}^{-\mathrm{i}kx_m}\right)^* = \hat{u}_k^*. \tag{8.577}$$

Damit sind nur noch $K+1$ Amplituden unabhängig voneinander, was den Rechenaufwand verringert.

8.2.3. Diskrete Fouriertransformation als lineare Operation

Wenn wir die Gitterpunkte (8.571) einsetzen, erhalten wir aus (8.573) für den Vektor der Funktionswerte

$$\vec{u} = u_j = \sum_{k=-K}^{K} \hat{u}_k \underbrace{\mathrm{e}^{\mathrm{i}kj2\pi/N}}_{G_{jk}} = \mathsf{G}\cdot\hat{\vec{u}}. \tag{8.578}$$

Wir sehen also, daß die Abbildung $\hat{\vec{u}} \to \vec{u}$ der Amplituden auf die Funktionswerte (Transformation aus dem Fourier-Raum in den Ortsraum) eine lineare Transformation ist, die durch eine Multiplikation mit der Matrix

$$\mathsf{G} = G_{jk} = \mathrm{e}^{\mathrm{i}kj2\pi/N} \tag{8.579}$$

erreicht wird. Diese Transformation kostet $O(N^2)$ Operationen (Multiplikationen). Entsprechend kann man auch die diskrete Fouriertransformation (8.574) als Multiplikation einer Matrix mit dem Vektor der Funktionswerte schreiben

$$\hat{\vec{u}} = \mathsf{F}\cdot\vec{u} \tag{8.580}$$

mit der Transformationsmatrix (vgl. (8.574))

$$\mathsf{F} = F_{km} = \frac{1}{N}\mathrm{e}^{-\mathrm{i}km2\pi/N} \tag{8.581}$$

Mit Hilfe der diskreten Orthogonalitätsrelation kann man zeigen, daß F die Inverse von G ist: $\mathsf{F} = \mathsf{G}^{-1}$. Denn es gilt[5]

$$\mathsf{F}\cdot\mathsf{G} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{e}^{2\pi\mathrm{i}kj/N}\mathrm{e}^{-2\pi\mathrm{i}km/N} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{e}^{2\pi\mathrm{i}k(j-m)/N} = \delta_{j,m} = \mathsf{I}. \tag{8.582}$$

[5] Der Summationsindex kann verschoben werden, da die Matrizen in jedem Index periodisch sind mit Periode N.

8.2.4. Schnelle Fourier-Transformation

Wie wir gesehen haben, kostet die Berechnung der Fourier-Transformierten $\hat{\vec{u}}$ aus den Funktionswerten $\vec{u}$ (bzw. umgekehrt) eine Anzahl von $O(N^2)$ Operationen (Matrix-Multiplikation), wenn man die normale Matrix-Vektor-Multiplikation verwendet. Die Anzahl der Operationen kann man reduzieren, wenn man anders vorgeht.

Bei der T *schnellen Fouriertransformation* (FFT) wird die ursprüngliche Fouriertransformation für N Punkte sukzessive auf Fourier-Transformationen mit jeweils der halben Anzahl von Punkten zurückgeführt ($N \to N/2 \to N/4 \to ...$). Diese Aufspaltung wird solange fortgesetzt bis nur noch die triviale Fouriertransformation für einen einzigen Punkt übrig bleibt.[6]

Mit Hilfe der FFT kann man die Anzahl der erforderlichen Operationen auf $O(N \ln N)$ reduzieren. Bei großen Werten von N, sagen wir $N = 100$, ist die FFT damit um einen Faktor von $N/\ln N \approx 22$ schneller als die Matrix-Multiplikation.[7] In drei Dimensionen ergibt sich damit schon der beträchtliche Faktor $22^3 \approx 10^4$. Bei einer Anzahl von 10^6 Stützstellen in einer Dimension hat man eine Ersparnis um den Faktor $\approx 7 \times 10^4$.

Um die Vorteile der FFT zu nutzen, darf N keine Primzahl sein. Die größte Beschleunigung erhält man, wenn N eine Potenz von 2 ist: $N = 2^p$, $p \in \mathbb{N}$.[8] Dann kann man die diskrete Fourier-Transformation für N Stützstellen auf zwei diskrete Fourier-Transformationen mit jeweils $N/2$ Stützstellen zurückführen, wenn wir die Summe in zwei Teilsummen spalten, die sich nur über die geraden bzw. die ungeraden Indizes erstrecken. Wenn wir die Summation von 0 bis $N-1$ laufen lassen und den Faktor N^{-1} weglassen, erhalten wir aus (8.574)

$$\begin{aligned}\hat{u}_k &= \sum_{j=0}^{N-1} u_j \mathrm{e}^{-\mathrm{i}k2\pi j/N} = \sum_{j=0}^{N/2-1} u_{2j} \mathrm{e}^{-\mathrm{i}k2\pi(2j)/N} + \sum_{j=0}^{N/2-1} u_{2j+1} \mathrm{e}^{-\mathrm{i}k2\pi(2j+1)/N} \\ &= \underbrace{\sum_{j=0}^{N/2-1} u_{2j} \mathrm{e}^{-\mathrm{i}k2\pi j/(N/2)}}_{:=\hat{u}_k^{\mathrm{even}}:=\hat{u}_k^{(0)}} + \underbrace{\mathrm{e}^{-\mathrm{i}k2\pi/N}}_{:=W^k} \underbrace{\sum_{j=0}^{N/2-1} u_{2j+1} \mathrm{e}^{-\mathrm{i}k2\pi j/(N/2)}}_{:=\hat{u}_k^{\mathrm{odd}}:=\hat{u}_k^{(1)}} . \end{aligned} \tag{8.583}$$

Man sieht, daß man die Fourier-Transformierte $\hat{u}_k$ als Summe darstellen kann, die aus den Fourier-Transformierten der Funktionswerte mit geraden Indizes und derjenigen mit ungeraden Indizes besteht. Beide, FTs haben dann jeweils die halbe Länge $N/2$. Dadurch haben wir schon eine Ersparnis: Die beiden Fourier-Transformationen der Länge $N/2$ kosten nur $2 \times (N/2)^2 = 2 \times N^2/4 = N^2/2$ Operationen. Hinzu kommen N Multiplikationen mit einem komplexen Exponentialfaktor und N Additionen

[6] hk: Siehe auch Boyd (2000) (pdf file) für eine gute Einführung inkl. FFT für Chebyshev.
[7] Dies gilt nur im Prinzip. In der Praxis kommen noch maschinenabhängige Faktoren hinzu.
[8] In (8.573) war N ungerade. Man kann aber auch N gerade wählen und die Summe in (8.573) von 1 bis N oder von 0 bis $N-1$ nehmen, vgl. Fußnote 3 auf S. 186.

(wenn man sie mitrechnen will); in Summe also

$$\frac{N^2}{2} + 2N < N^2, \quad \text{falls } N > 4. \tag{8.584}$$

Bei dieser Betrachtung haben wir ausgenutzt, daß wir die Produkte (in den Summen über j) nur für die halbe Anzahl $N/2$ der k-Werte bilden müssen, und nicht für alle k-Werte aus dem eigentlichen Bereich $[0, N-1]$. Denn die Summanden in den FT's der halben Länge $\hat{u}_k^{\text{odd}}$ und $\hat{u}_k^{\text{even}}$ sind periodisch in k mit der Periode $N/2$. Denn für festes j gilt

$$\mathrm{e}^{-\mathrm{i}k2\pi j/(N/2)} \overset{k\to k+N/2}{=} \mathrm{e}^{-\mathrm{i}k2\pi j/(N/2)} \underbrace{\mathrm{e}^{-\mathrm{i}2\pi j}}_{=1}. \tag{8.585}$$

Wir können nun die Zerlegung weiterführen und die beiden Summen in (8.583) wieder in zwei Summen über die geraden und die ungeraden Indizes aufspalten. Als Beispiel sei die zweite Unterteilung durchgeführt, wobei wir den Faktor $W = \mathrm{e}^{-2\pi\mathrm{i}/N}$ einführen,[9]

$$\begin{aligned}
\hat{u}_k &\overset{(8.583)}{=} \underbrace{\sum_{j=0}^{N/4-1} u_{2(2j)}\mathrm{e}^{-\mathrm{i}k2\pi(2j)/(N/2)} + \sum_{j=0}^{N/4-1} u_{2(2j+1)}\mathrm{e}^{-\mathrm{i}k2\pi(2j+1)/(N/2)}}_{\hat{u}_k^{(0)}} \\
&\quad + W^k \underbrace{\left(\sum_{j=0}^{N/4-1} u_{2(2j)+1}\mathrm{e}^{-\mathrm{i}k2\pi(2j)/(N/2)} + \sum_{j=0}^{N/4-1} u_{2(2j+1)+1}\mathrm{e}^{-\mathrm{i}k2\pi(2j+1)/(N/2)} \right)}_{\hat{u}_k^{(1)}} \\
&= \sum_{j=0}^{N/4-1} u_{2(2j)}\mathrm{e}^{-\mathrm{i}k2\pi j/(N/4)} + \sum_{j=0}^{N/4-1} u_{2(2j+1)}\mathrm{e}^{-\mathrm{i}k(2j+1)\pi/(N/4)} \\
&\quad + W^k \left(\sum_{j=0}^{N/4-1} u_{2(2j)+1}\mathrm{e}^{-\mathrm{i}k2j\pi/(N/4)} + \sum_{j=0}^{N/4-1} u_{2(2j+1)+1}\mathrm{e}^{-\mathrm{i}k(2j+1)\pi/(N/4)} \right) \\
&= \underbrace{\underbrace{\sum_{j=0}^{N/4-1} u_{2(2j)}\mathrm{e}^{-\mathrm{i}k2\pi j/(N/4)}}_{=\hat{u}_k^{(00)}} + W^{2k} \underbrace{\sum_{j=0}^{N/4-1} u_{2(2j+1)}\mathrm{e}^{-\mathrm{i}k2j\pi/(N/4)}}_{=\hat{u}_k^{(01)}}}_{\hat{u}_k^{(0)}} \\
&\quad + W^k \underbrace{\left(\underbrace{\sum_{j=0}^{N/4-1} u_{2(2j)+1}\mathrm{e}^{-\mathrm{i}k2j\pi/(N/4)}}_{=\hat{u}_k^{(10)}} + W^{2k} \underbrace{\sum_{j=0}^{N/4-1} u_{2(2j+1)+1}\mathrm{e}^{-\mathrm{i}k2j\pi/(N/4)}}_{=\hat{u}_k^{(11)}} \right)}_{\hat{u}_k^{(1)}}
\end{aligned}$$

[9] Beachte: $W^k = \left(\mathrm{e}^{-2\pi\mathrm{i}/N}\right)^k = \mathrm{e}^{-2\pi\mathrm{i}k/N}$.

Gitterpunkt j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	0	1	0	1	0	1	0	1
FFT-Zuordnung	00	10	01	11	00	10	01	11
	000	100	010	110	001	101	011	111
Bit-Inversion	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**

Tabelle 8.1.: Zuordnung der Werte an den Gitterpunkten j zu den einzelnen FFTs bei sukzessiver Halbierung der Punktzahl der FFTs. Eine 0 steht für gerade Punkte und eine 1 für ungerade Punkte. Bei jeder Aufteilung in zwei Fouriertransformation der geraden und ungeraden Punkte, aber der halben Länge, wird dem Index (Bezeichnung der FFT) eine 0 (gerade) oder eine 1 (ungerade) angehängt, je nachdem, ob es sich bei der neuen Anordnung um eine gerade oder eine ungerade Position in der momentanen Fourierreihe handelt.

$$= \ldots \tag{8.586}$$

Hierbei können die Exponentialfaktoren in den Summen ebenfalls als Potenzen von W ausgedrückt werden. Wenn $N = 2^q$ eine ganzzahlige Potenz von 2 ist, kann man diese Aufteilung bis zu dem Punkt durchführen, bei dem die Summe nur noch über einen einzigen Funktionswert zu bilden ist. Diese Prozedur ist anschaulich in Tab. 8.1 dargestellt. Bei jeder Zerlegung in gerade bzw. ungerade Terme erhält der zur Kennzeichnung hochgestellte Index die Ziffer 0 (gerade) oder 1 (ungerade) angehängt. Bei $N = 8$ muß man die Fouriertransformation drei mal ($= \log_2(8)$) in gerade und ungerade Beiträge zerlegen. In der Tabelle ist die jeweilige Zuordnung jedes einzelnen Summanden entweder zu den geraden oder ungeraden Punkten der jeweiligen Transformation dargestellt. Wenn man auf dem niedrigsten Niveau angekommen ist, besteht die Fourier-Transformation nur noch aus einem einzigen Punkt. Wenn man die Kodierung (den hochgestellten Index) der resultierenden 1-Punkt-FFTs bitweise invertiert (*Bit-Inversion*), erhält man gerade die Binär-Darstellung des Gitter-Index j des jeweiligen Funktionswertes. Der FFT-Algorithmus besteht dann in einer sukzessiven gewichteten Addition der im Baumdiagramm Abb. 8.4 benachbarten Werte.[10]

Um den numerischen Aufwand abzuschätzen, beachten wir, daß wir in jedem Schritt N Werte berechnen müssen, zum Beispiel

$$W_{00}\hat{u}_k^{(00)} + W_{01}\hat{u}_k^{(01)} + W_{10}\hat{u}_k^{(10)} + W_{11}\hat{u}_k^{(11)} \tag{8.587}$$

Dazu benötigt man bei *jeder* Unterteilung $O(N)$ Operationen, denn die Anzahl der Summanden erhöht sich zwar mit der Tiefe der Unterteilung, aber in demselben Maße nimmt die Zahl der Werte k, für welche man die Summen berechnen muß, ab (wegen der Periodizität des Ausdrucks). In dem Beispiel haben wir $O(4)$ Summanden, die Summe muß man aber nur für $N/4$ k-Werte berechnen. Da wir $\log_2 N$ Unterteilungen haben, beträgt der Gesamtaufwand zur Berechnung von $\hat{u}_k$

[10] Die immer wiederkehrenden Operationen können auch durch das sogenannte *Butterfly*-Diagramm symbolisiert werden.

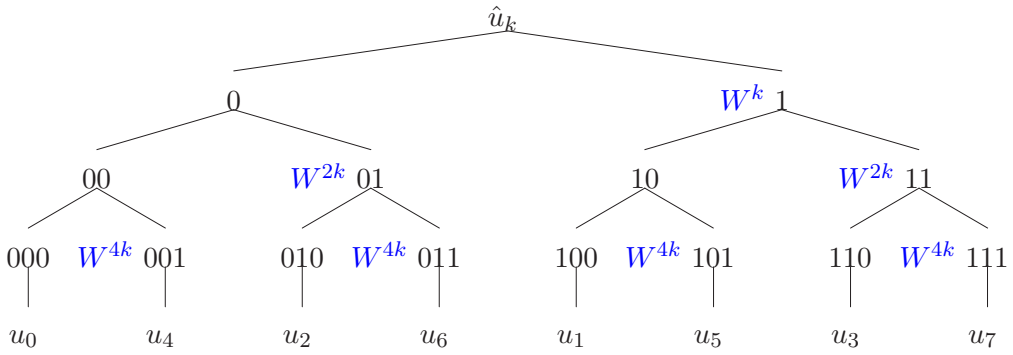


Abbildung 8.4.: Baumdiagramm der Aufspaltung einer FFT für $N = 8$. Auf jeder Stufe muß man eine gewichtete Summe über die beiden Terme nehmen, die von der Stelle verzweigen. Die Anzahl der k-Werte, für die man das machen muß, halbiert sich aber mit jeder Stufe. Die Ausgangsreihenfolge ergibt sich durch die Binärinversion des Gitterindex.

nur $O(N \log_2 N)$ Operationen. Dies muß mit den $O(N^2)$ Operationen für die DFT verglichen werden.

A. Algebraische Grundlagen

Die lineare Algebra ist für das numerische Rechnen essentiell. Daher sollen in diesem Anhang die wichtigsten Regeln wiederholt werden. Das Kapitel hält sich sehr eng an das Buch von Golub and Ortega (1996).

A.1. Vektoren und Matrizen

Ein *Spaltenvektor* ist ein n-Tupel von reellen oder komplexen Zahlen

$$\vec{x} = x_i = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{A.588}$$

Der entsprechende *Zeilenvektor* ist $\vec{x}^{\mathrm{T}} = (x_1, ..., x_n)$. Der hochgestellte Index T bezeichnet das *Transponierte* (s.u.). Eine $m \times n$-Matrix ist ein Feld

$$\mathsf{A} = A_{ij} = \begin{pmatrix} A_{11} & \dots & A_{1n} \\ \vdots & & \vdots \\ A_{m1} & \dots & A_{mn} \end{pmatrix} \tag{A.589}$$

mit m Zeilen und n Spalten, wobei jedes Element $A_{ij} \in \mathbb{R}$ reell oder $A_{ij} \in \mathbb{C}$ komplex sein kann. Die natürlichen Zahlen m und n bezeichnen die *Dimensionen der Matrix*. Ist $m = n$ so heißt die Matrix quadratisch, ansonsten rechteckig.

Eine *Untermatrix* von A entsteht, wenn man irgendwelche Zeilen und/oder Spalten der Matrix ersatzlos streicht. Ein Vektor kann als eine Matrix mit $n = 1$ aufgefaßt werden. Eine Matrix kann man auch als einen Zeilenvektor auffassen, dessen Elemente aus Spaltenvektoren bestehen

$$\mathsf{A} = \left(\vec{A}_1, \dots, \vec{A}_n\right), \quad \text{mit} \quad \vec{A}_i = \begin{pmatrix} A_{1i} \\ \vdots \\ A_{mi} \end{pmatrix} \tag{A.590}$$

Gleichberechtigt können wir A auch als einen Spaltenvektor auffassen, dessen Elemente aus Zeilenvektoren bestehen.

Die zu einer gegebenen Matrix A *transponierte Matrix* A^{T} erhält man, wenn man die beiden Indizes vertauscht, d.h. wenn man Zeilen und Spalten miteinander vertauscht. Dies ist gleichbedeutend mit einer Spiegelung der Matrix um die *Hauptdiagonale* (Abb. A.1).

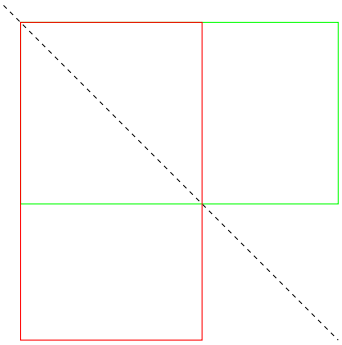


Abbildung A.1.: Die Transponierte (rot) einer Matrix (grün).

Damit ist

$$\mathsf{B} = \mathsf{A}^{\mathrm{T}} = A_{ij}^{\mathrm{T}} = A_{ji} = \begin{pmatrix} B_{11} = A_{11} & \dots & B_{1m} = A_{m1} \\ \vdots & & \vdots \\ B_{n1} = A_{1n} & \dots & B_{nm} = A_{mn} \end{pmatrix}. \tag{A.591}$$

Dies gilt auch, wenn die Matrix komplex ist. Für komplexe Matrizen ist jedoch neben der transponierten auch die *konjugiert transponierte Matrix* von Interesse, die durch Transponieren und Bilden des konjugiert Komplexen entsteht

$$\mathsf{A}^{\dagger} = (A_{ij})^{\dagger} = A_{ji}^{*} = \begin{pmatrix} A_{11}^{*} & \dots & A_{m1}^{*} \\ \vdots & & \vdots \\ A_{1n}^{*} & \dots & A_{mn}^{*} \end{pmatrix}. \tag{A.592}$$

Hierbei bezeichnet * bei Zahlen das konjugiert Komplexe und † bei Matrizen das konjugiert Transponierte.[1] Die konjugiert transponierte Matrix wird auch *hermitesche konjugierte Matrix* oder *adjungierte Matrix* genannt.

Charles Hermite*
1822–1901

Eine Matrix A heißt *symmetrisch*, wenn $\mathsf{A} = \mathsf{A}^{\mathrm{T}}$ ist, d.h. $A_{ij} = A_{ji}$. Nur eine quadratische Matrix kann auch symmetrisch sein. Wenn $\mathsf{A} = \mathsf{A}^{\dagger}$ bzw. $A_{ij} = A_{ji}^{*}$, dann heißt die Matrix *hermitesch* oder *selbstadjungiert*. Beispiele sind

$$\underbrace{\begin{pmatrix} 1 & 3 & \mathrm{i} & 7 \\ 3 & 9 & -2\mathrm{i} & -8 \\ \mathrm{i} & -2\mathrm{i} & 1 & 0 \\ 7 & -8 & 0 & -\mathrm{i} \end{pmatrix}}_{\text{(symmetrisch)}} \quad \text{und} \quad \underbrace{\begin{pmatrix} 1 & 1 & \mathrm{i} & 2\mathrm{i} \\ 1 & 9 & -2 & -8\mathrm{i} \\ -\mathrm{i} & -2 & 0 & 7 \\ -2\mathrm{i} & 8\mathrm{i} & 7 & 0 \end{pmatrix}}_{\text{(hermitesch)}}. \tag{A.593}$$

[1] Zum Beispiel ist

$$\begin{pmatrix} 1+2\mathrm{i} & 3 \\ 2-5\mathrm{i} & 4 \end{pmatrix}^{\dagger} = \begin{pmatrix} 1-2\mathrm{i} & 2+5\mathrm{i} \\ 3 & 4 \end{pmatrix}.$$

*Charles Hermite war französischer Mathematiker. Er arbeitete über Zahlentheorie, Algebra, orthogonale Polynome und elliptische Funktionen und erzielte wichtige Ergebnisse über doppelt periodische Funktionen und Invarianten quadratischer Formen. 1858 löste er eine algebraische Gleichung 5ten Grades mit Hilfe elliptischer Funktionen. Im Jahre 1873 bewies er, daß die Eulersche Zahl e transzendent ist. Erst darauf aufbauend wurde 1882 die Transzendenz von π von Carl Louis Ferdinand von Lindemann bewiesen.

Leopold Kronecker 1823–1891

Eine quadratische $n \times n$ Matrix heißt *diagonale Matrix*, wenn nur die Diagonalelemente von Null verschieden sind, wenn also

$$\mathsf{A} = \begin{pmatrix} A_{11} & & \\ & \ddots & \\ & & A_{nn} \end{pmatrix} = a_i \delta_{ij}, \tag{A.594}$$

wobei δ_{ij} das *Kronecker Symbol* ist, mit

$$\delta_{ij} = \begin{cases} 1, & \text{falls } i = j, \\ 0, & \text{falls } i \neq j. \end{cases} \tag{A.595}$$

Wichtige Matrizen sind auch die obere und untere *Dreiecksmatrix*. Bei diesen $n \times n$ Matrizen sind entweder die Elemente oberhalb oder die Elemente unterhalb der Hauptdiagonalen gleich Null. Es sind also

$$\mathsf{L} = L_{ij} = \begin{pmatrix} L_{11} & & \\ \vdots & \ddots & \\ L_{n1} & \dots & L_{nn} \end{pmatrix}, \qquad \mathsf{U} = U_{ij} = \begin{pmatrix} U_{11} & \dots & U_{1n} \\ & \ddots & \vdots \\ & & U_{nn} \end{pmatrix}. \tag{A.596}$$

A.2. Operationen mit Vektoren und Matrizen

Vektoren und Matrizen können mit einer Zahl α multipliziert werden. Es gilt

$$\alpha \vec{x} = \begin{pmatrix} \alpha x_1 \\ \vdots \\ \alpha x_n \end{pmatrix} \qquad \text{und} \qquad \alpha \mathsf{A} = \begin{pmatrix} \alpha A_{11} & \dots & \alpha A_{1n} \\ \vdots & & \vdots \\ \alpha A_{m1} & \dots & \alpha A_{mn} \end{pmatrix}. \tag{A.597}$$

Vektoren können addiert und skalar multipliziert werden, wenn ihre Dimensionen identisch sind. Für die Addition gilt

$$\vec{x} + \vec{y} = \begin{pmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{pmatrix}, \tag{A.598}$$

für die skalare Multiplikation (durch den Punkt $\cdot$ symbolisiert) gilt

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i = x_i y_i = x_1 y_1 + \ldots + x_n y_n. \tag{A.599}$$

Oft wird das Summenzeichen weggelassen. Dann gilt: Wenn zwei Indizes in einem Produkt doppelt vorkommen (hier i), dann muß die Summe aller Produkte gebildet werden, wobei i von 1 bis n läuft. Dies nennt man auch *Einstein-Konvention*.

Matrizen können miteinander addiert werden, wenn alle entsprechenden Dimensionen der beiden Matrizen gleich sind. Matrizen können auch miteinander (skalar)

multipliziert werden, wenn die Dimensionen der beiden Matrizen, über welche die Summe gebildet werden muß, gleich ist. Ist zum Beispiel A_{ij} eine $m \times n$-Matrix und B_{ij} eine $n \times l$-Matrix, dann gilt

$$\mathsf{A} \cdot \mathsf{B} = \sum_{j=1}^{n} A_{ij} B_{jk} = A_{ij} B_{jk} = \begin{pmatrix} A_{11} & \dots & A_{1n} \\ \vdots & & \vdots \\ A_{m1} & \dots & A_{mn} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & \dots & B_{1l} \\ \vdots & & \vdots \\ B_{n1} & \dots & B_{nl} \end{pmatrix}$$
$$= \mathsf{C} = \begin{pmatrix} \sum_n A_{1n} B_{n1} & \dots & \sum_n A_{1n} B_{nl} \\ \vdots & & \vdots \\ \sum_n A_{mn} B_{n1} & \dots & \sum_n A_{mn} B_{nl} \end{pmatrix} = \begin{pmatrix} C_{11} & \dots & C_{1l} \\ \vdots & & \vdots \\ C_{m1} & \dots & C_{ml} \end{pmatrix}. \quad \text{(A.600)}$$

Es wird also summiert über den Spaltenindex von A und den Zeilenindex von B. Beide müssen identisch sein. Sonst ist das Skalarprodukt nicht definiert. Man erhält also ein $m \times l$-Matrix C_{ij}.[2]

Für die Addition und Multiplikation von Matrizen gilt das Assoziativ- und Distributivgesetz. Das Kommutativgesetz gilt aber nur für die Addition von Matrizen, nicht für die Multiplikation. Im allgemeinen ist

$$\mathsf{A} \cdot \mathsf{B} \neq \mathsf{B} \cdot \mathsf{A}, \quad \text{(A.601)}$$

d.h. die Reihenfolge der Faktoren muß bei Umformungen immer gleich bleiben. Außerdem gilt für die Transposition

$$(\mathsf{A} + \mathsf{B})^{\mathrm{T}} = \mathsf{A}^{\mathrm{T}} + \mathsf{B}^{\mathrm{T}} \qquad \text{aber} \qquad (\mathsf{A} \cdot \mathsf{B})^{\mathrm{T}} = \mathsf{B}^{\mathrm{T}} \cdot \mathsf{A}^{\mathrm{T}} \quad \text{(Reihenfolge!)}. \quad \text{(A.602)}$$

A.2.1. Spezialfälle

Ein Grenzfall der Matrix-Multiplikation tritt auf, wenn die beiden Matrizen A und B die Dimension $1 \times n$ und $n \times 1$ haben. Dann gilt offensichtlich

$$(A_1, \dots, A_n) \cdot \begin{pmatrix} B_1 \\ \vdots \\ B_n \end{pmatrix} = \sum_{i=1}^{n} A_i B_i. \quad \text{(A.603)}$$

Dies ist nichts anderes als das normale Skalarprodukt zwischen zwei Vektoren. Es wird auch *inneres Produkt* genannt. Sind die Dimensionen umgekehrt $n \times 1$ und $1 \times n$, dann ist

$$\begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix} (B_1, \dots, B_n) = \begin{pmatrix} A_1 B_1 & \dots & A_1 B_n \\ \vdots & & \vdots \\ A_n B_1 & \dots & A_n B_n \end{pmatrix}. \quad \text{(A.604)}$$

[2] Wir werden den Punkt $(\cdot)$ des Skalarprodukts immer ausschreiben um anzudeuten, daß die Summe über einen Index zu nehmen ist. In anderen Publikationen wird der Punkt oft weggelassen. Es ist aber übersichtlicher, ihn beizubehalten, denn manchmal treten auch andere Produkte auf, die anders gebildet werden (Kreuzprodukt oder dyadisches Produkt).

Diese Produkt heißt *dyadisches Produkt*. Das dyadische Produkt zweier Vektoren wird bei uns dadurch angedeutet, daß kein Operator zwischen den Vektoren steht: $\vec{a}\vec{b} = \mathsf{C}$. Bei dieser Schreibweise ist es nicht sofort klar, ob es sich um einen Zeilen- oder einen Spaltenvektor handelt. Alternativ kann man man anstelle von $\vec{a}\cdot\vec{b}$ schreiben $\vec{a}^{\mathrm{T}}\vec{b} = \alpha$ und für $\vec{a}\vec{b}$ die Form $\vec{a}\vec{b}^{\mathrm{T}} = \mathsf{C}$ unter weglassen des Punktes $\cdot$ verwenden.

A.3. Orthogonalität und lineare Unabhängigkeit

Zwei Vektoren $\vec{x}$ und $\vec{y}$ heißen *orthogonal*, wenn ihr Skalarprodukt verschwindet, d.h. wenn

$$\vec{x}\cdot\vec{y} = \sum_i x_i y_i = 0. \tag{A.605}$$

Sie werden *orthonormal* genannt, wenn zusätzlich noch gilt $\vec{x}\cdot\vec{x} = \vec{y}\cdot\vec{y} = 1$. Ein System von Vektoren $\{\vec{x}_1, \ldots, \vec{x}_n\}$ heißt orthogonal, wenn für alle $i \neq j$ gilt $\vec{x}_i \cdot \vec{x}_j = 0$. Falls sogar

$$\vec{x}_i \cdot \vec{x}_j = \begin{cases} 1, & \text{falls } i = j, \\ 0, & \text{falls } i \neq j, \end{cases} \tag{A.606}$$

gilt, so bilden die Vektoren ein *Orthonormalsystem*.

Wenn man einen Satz von orthogonalen Vektoren hat, so kann man durch Skalierung daraus einen Satz von orthonormalen Vektoren herhalten, wenn man alle Vektoren auf die Länge 1 normiert. Dies erreicht man durch Division des Vektors $\vec{x}$ durch seine Länge (*Eukilidische Norm*) $\sqrt{\vec{x}\cdot\vec{x}} = \sqrt{x_1^2 + \ldots + x_n^2}$. Der Vektor

$$\vec{y} = \frac{\vec{x}}{\sqrt{\vec{x}\cdot\vec{x}}} \tag{A.607}$$

besitzt dann die Länge 1, denn $\vec{y}\cdot\vec{y} = 1$. Für einen komplexen Vektor erhält man seine Länge durch $\sqrt{\vec{x}\cdot\vec{x}^*}$.

Ein Satz von m Vektoren $\{\vec{x}_1, \ldots, \vec{x}_m\}$ der Dimension n heißt *linear abhängig*, falls es Zahlen c_i gibt, die nicht alle Null sind, so daß

$$\sum_{i=1}^{m} c_i \vec{x}_i = 0. \tag{A.608}$$

Anschaulich bedeutet dies, daß man (A.608) nach einem Vektor, dessen Vorfaktor nicht verschwindet, auflösen kann. Es gibt dann gewisse Vektoren, die sich als eine Linearkombination aus den restlichen Vektoren darstellen lassen. Wenn jedoch (A.608) nur dann erfüllt ist, wenn alle $c_i = 0$ sind, dann heißen die Vektoren $\vec{x}_i$ *linear unabhängig*.

Die Menge $\mathbb{U}$ aller Linearkombinationen einer Menge von m n-dimensionalen Vektoren $\{\vec{x}_1, \ldots, \vec{x}_m\}$, d.h.

$$\mathbb{U} = \mathrm{span}\{\vec{x}_1, \ldots, \vec{x}_m\} = \{c_1\vec{x}_1 + \ldots + c_m\vec{x}_m\}, \qquad c_i \in \mathbb{R} \tag{A.609}$$

bildet einen *Unterraum* des Raumes aller n-dimensionaler Vektoren. Für reelle Vektoren ist also $\mathbb{U} \subseteq \mathbb{R}^n$.

Sind die Vektoren $\{\vec{x}_1, \ldots, \vec{x}_m\}$ linear unabhängig so hat der Unterraum $\text{span}\{\vec{x}_1, \ldots, \vec{x}_m\}$ die Dimension m. Dann bilden die Vektoren $\{\vec{x}_1, \ldots, \vec{x}_m\}$ eine *Basis* des m-dimensionalen Raumes, im reellen Fall also des $\mathbb{R}^m$.

Eine wichtige Eigenschaft von Matrizen ist die Anzahl der linear unabhängigen *Spaltenvektoren*. Diese Zahl wird als *Rang* der Matrix bezeichnet. Man kann dann zeigen, daß auch die Zahl der linear unabhängigen *Zeilenvektoren* gleich dem Rang der Matrix ist. Wenn A eine $m \times n$-Matrix ist und p die minimale Dimension $p = \min\{m, n\}$. Dann gilt $\text{Rang}(\mathsf{A}) \leq p$. Falls $\text{Rang}(\mathsf{A}) = p$ besitzt die Matrix A den *vollen Rang*.

A.4. Eigenschaften von Matrizen und Determinanten

A.4.1. Orthogonale Matrizen

Eine quadratische $n \times n$ Matrix heißt *orthogonal*, wenn gilt

$$\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A} = \mathsf{A} \cdot \mathsf{A}^{\mathrm{T}} = \mathsf{I}, \tag{A.610}$$

wobei

$$\mathsf{I} = \text{diag}(1, \ldots, 1) = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \tag{A.611}$$

die n-dimensionale Einheitsmatrix ist. Die Elemente des Produkts zweier $n \times n$-Matrizen bestehen aus den Skalarprodukten aller Zeilenvektoren der ersten mit den Spaltenvektoren der zweiten Matrix. Die Zeilenvektoren der Transponierten sind aber die Spaltenvektoren. Daher bestehen die Elemente von $\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A}$ aus allen Skalarprodukten der Spaltenvektoren von A untereinander und $\mathsf{A} \cdot \mathsf{A}^{\mathrm{T}}$ aus allen Skalarprodukten der Zeilenvektoren von A. Die Bedingung (A.610) ist dann identisch mit der Forderung, daß die Spalten- bzw. die Zeilenvektoren *orthonormal* sind.

Falls für eine komplexe Matrix mit orthonormalen Spalten (Zeilen) gilt

$$\mathsf{A}^{\dagger} \cdot \mathsf{A} = \mathsf{A} \cdot \mathsf{A}^{\dagger} = \mathsf{I}, \tag{A.612}$$

wird die Matrix A *unitäre Matrix* genannt.

A.4.2. Inverse Matrix

Durch eine Multiplikation mit der Einheitsmatrix I ändert sich eine Matrix A nicht. Offenbar gilt

$$\mathsf{A} = \mathsf{I} \cdot \mathsf{A} = \mathsf{A} \cdot \mathsf{I}. \tag{A.613}$$

Die Einheitsmatrix kann auch als Matrix von Spaltenvektoren $\vec{e}_i$ aufgefaßt werden, bei denen eine 1 in der i-ten Komponente erscheint und bei dem alle anderen Komponenten Null sind (*Einheitsvektoren*). Damit ist

$$\mathsf{I} = \left(\vec{e}_1, \quad \ldots, \vec{e}_n\right). \tag{A.614}$$

Wir betrachten nun die n inhomogenen linearen Gleichungssysteme

$$\mathsf{A} \cdot \vec{x}_i = \vec{e}_i, \qquad i = 1, \ldots, n \tag{A.615}$$

und nehmen an, daß das Gleichungssystem für jeden Wert $i \in \{1, \ldots, n\}$ eine eindeutige Lösung besitzt. Wenn wir die jeweiligen Lösungsvektoren $\vec{x}_i$ als Spalten einer Matrix X auffassen, lassen sich alle diese Gleichungssysteme zusammenfassen und in der Form

$$\mathsf{A} \cdot \mathsf{X} = \mathsf{I} \tag{A.616}$$

schreiben. Offenbar muß dann $\mathsf{X} = \mathsf{A}^{-1}$ die zu A *inverse Matrix* sein. Die Inverse wird als A^{-1} geschrieben.

Um ein lineares Gleichungssystem

$$\mathsf{A} \cdot \vec{x} = \vec{b} \tag{A.617}$$

zu lösen, könnte man mathematisch korrekt von links mit der Inversen A^{-1} multiplizieren. Man erhält dann die Lösung

$$\vec{x} = \mathsf{A}^{-1} \cdot \vec{b}. \tag{A.618}$$

In der Praxis ist aber die Berechnung der Inversen zu aufwendig und die Lösung des linearen Gleichungssystems erfolgt anders (siehe Kap. 2). Beachte, daß im Spezialfall einer orthogonalen Matrix ($\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A} = \mathsf{I}$) die Inverse identisch mit der Transponierten ist: $\mathsf{A}^{-1} = \mathsf{A}^{\mathrm{T}}$.

A.4.3. Determinanten

Determinanten von quadratischen $n \times n$-Matrizen können im Prinzip mit der Leibniz-Formel (siehe auch (1.5))

$$\det \mathsf{A} = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma)\, A_{1\sigma(1)} \ldots A_{n\sigma(n)} \tag{A.619}$$

berechnet werden. Hierbei ist σ eine *Permutation* von $(1, \ldots, n)$. Die Menge aller derartigen Permutationen wird als S_n bezeichnet und

$$\mathrm{sgn}(\sigma) = \begin{cases} 1, & \text{falls } \sigma \text{ eine gerade Permutation von } (1, \ldots, n) \text{ ist,} \\ -1, & \text{falls } \sigma \text{ eine ungerade Permutation von } (1, \ldots, n) \text{ ist.} \end{cases} \tag{A.620}$$

Da die Anzahl aller Permutationen von $(1,\ldots,n)$ $n!$ beträgt, ist die Summe über $n!$ derartiger Produkte zu erstrecken. Für $n=2$ gibt es nur die gerade Permutation $(1,2)$ und die ungerade Permutation $(2,1)$, so daß

$$\det \mathsf{A} = A_{11}A_{22} - A_{12}A_{21}. \tag{A.621}$$

Für $n=3$ gibt es die $3!=6$ Permutationen (hier schon mit Vorzeichen geschrieben) $(1,2,3)$, $-(1,3,2)$, $-(2,1,3)$, $(2,3,1)$, $(3,1,2)$ und $-(3,2,1)$ und wir erhalten

$$\det \mathsf{A} = A_{11}A_{22}A_{33} - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31}. \tag{A.622}$$

Dies Ergebnis erhält man auch aus der Entwicklung nach Unterdeterminanten.

Für Determinanten gelten die die folgenden Regeln:

1. Sind 2 Spalten oder Zeilen von A identisch oder ist eine Spalte oder Zeile gleich Null, dann ist $\det \mathsf{A} = 0$.
2. Werden zwei Spalten oder Zeilen von A vertauscht, dann wechselt $\det \mathsf{A}$ das Vorzeichen.
3. Wird eine Spalte oder eine Zeile mit einer Zahl α multipliziert, dann wird die Determinante mit α multipliziert. Dies bedeutet, daß $\det(\alpha\mathsf{A}) = \alpha^n \det \mathsf{A}$ ist.
4. $\det \mathsf{A}^{\mathrm{T}} = \det \mathsf{A}$ und $\det \mathsf{A}^{\dagger} = (\det \mathsf{A})^*$.
5. Wird ein skalares Vielfaches einer Spalte (Zeile) zu einer anderen Spalte (Zeile) addiert, so ändert sich die Determinante nicht.
6. Für zwei $n \times n$-Matrizen gilt $\det(\mathsf{A}\cdot\mathsf{B}) = \det \mathsf{A} \det \mathsf{B}$.
7. Die Determinante einer einer Dreiecksmatrix ist das Produkt der Hauptdiagonalelemente, z.B. $\det \mathsf{U} = U_{11}\ldots U_{nn}$.
8. Sei A_{ij} die $(n-1)\times(n-1)$-Untermatrix von A; Die durch Streichen der i-ten Zeile und j-ten Spalte aus A hervorgeht, dann ist

$$\det \mathsf{A} = \sum_{j=1}^{n} A_{ij}(-1)^{i+j} \det \mathsf{A}_{ij}, \quad \text{für jedes } i, \tag{A.623a}$$

$$\det \mathsf{A} = \sum_{i=1}^{n} A_{ij}(-1)^{i+j} \det \mathsf{A}_{ij}, \quad \text{für jedes } j. \tag{A.623b}$$

Dies ist nichts anderes als die Entwicklung einer Determinante nach Zeilen bzw. Spalten. Die Unterdeterminante (mit Vorzeichen) $(-1)^{i+j}\det \mathsf{A}_{ij}$ wird *Adjunkte* von A_{ij} genannt.

A.4.4. Reguläre Matrizen

Wenn also eine zu A inverse Matrix A^{-1} existiert, dann gilt (A.616). Daraus erhalten wir mit Hilfe der obigen Regel 4

$$\det \mathsf{A} \det \mathsf{A}^{-1} = \det \mathsf{I} = 1. \tag{A.624}$$

Daher darf für eine invertierbare Matrix die Determinante nicht verschwinden: $\det \mathsf{A} \neq 0$. Eine invertierbare Matrix nennt man auch *reguläre Matrix*. Ist eine Matrix nicht regulär, wird sie *singulär* genannt. Die folgenden Aussagen über eine $n \times n$-Matrix sind äquivalent:

1. A ist regulär.
2. A^{-1} existiert.
3. $\det \mathsf{A} \neq 0$.
4. Das lineare System $\mathsf{A} \cdot \vec{x} = 0$ besitzt nur die triviale Lösung $\vec{x} = 0$.
5. Für jeden Vektor $\vec{b}$ besitzt das lineare System $\mathsf{A}\vec{x} = \vec{b}$ eine eindeutige Lösung.
6. Die Spalten (Zeilen) von A sind linear unabhängig.
7. $\mathrm{Rang}(\mathsf{A}) = n$.

Wegen (A.610) und Regel 4 sind auch orthogonale Matrizen regulär; ihre Determinanten verschwinden nicht.

A.4.5. Positiv und negativ definite Matrizen

Eine wichtige Rolle in der numerischen linearen Algebra spielt die Bedingung

$$\vec{x}^{\mathrm{T}} \cdot \mathsf{A} \cdot \vec{x} > 0, \qquad \text{für } \vec{x} \neq 0 \text{ und } \vec{x} \in \mathbb{R}^n. \tag{A.625}$$

Ein reelle Matrix, die diese Bedingung erfüllt, wird *positiv definit* genannt. Positiv definite Matrizen sind regulär.[3]

[3] Wäre die Matrix A nicht regulär, so gäbe es nach Regel 4 eine nicht-triviale Lösung $\vec{x}$ mit $\mathsf{A} \cdot \vec{x} = 0$. Dies würde aber im Widerspruch zu (A.625) stehen. Beachte, daß eine positiv definite Matrix nicht unbedingt symmetrisch sein muß. Die antisymmetrische Matrix

$$\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

ist positiv definit.

In ähnlicher Weise definiert man *negativ definit* für eine reelle Matrix als

$$\vec{x}^{\mathrm{T}} \cdot \mathsf{A} \cdot \vec{x} < 0, \qquad \text{für } \vec{x} \neq 0 \text{ und } \vec{x} \in \mathbb{R}^n. \tag{A.626}$$

Falls in den Relationen (A.625) und (A.626) nur die schwächeren Bedingungen $\geq$ bzw. $\leq$ gelten, heißen die Matrizen *positiv semi-definit* bzw. *negativ semi-definit*.

Eine Matrix A ist positiv (negativ) definit, genau dann wenn ihr symmetrischer Anteil positiv (negativ) definit ist. Um das zu sehen, zerlegt man die Matrix in einen symmetrischen und einen antisymmetrischen Anteil

$$\mathsf{A} = \underbrace{\frac{1}{2}\left(\mathsf{A} + \mathsf{A}^{\mathrm{T}}\right)}_{S_{ij},\ \text{symmetrisch}} + \underbrace{\frac{1}{2}\left(\mathsf{A} - \mathsf{A}^{\mathrm{T}}\right)}_{C_{ij},\ \text{antisymmetrisch}} . \tag{A.627}$$

Wegen $C_{ij} = -C_{ji}$ gilt

$$\vec{x}^{\mathrm{T}} \cdot \mathsf{C} \cdot \vec{x} = x_i C_{ij} x_j = -x_i C_{ji} x_j = -x_j C_{ji} x_i = 0. \tag{A.628}$$

Damit ist

$$\vec{x}^{\mathrm{T}} \cdot \mathsf{A} \cdot \vec{x} = \vec{x}^{\mathrm{T}} \cdot \mathsf{S} \cdot \vec{x} = \frac{1}{2}\vec{x}^{\mathrm{T}} \cdot \left(\mathsf{A} + \mathsf{A}^{\mathrm{T}}\right) \cdot \vec{x}. \tag{A.629}$$

Für eine reelle und symmetrische Matrix A sind die folgenden Aussagen äquivalent:

- A ist positiv definit,
- alle Eigenwerte von A sind positiv,
- alle Hauptunterdeterminanten von A sind positiv.

Ist eine $N \times N$ Matrix reell und symmetrisch, dann kann man sie in der speziellen Form

$$\mathsf{A} = \mathsf{G} \cdot \mathsf{G}^{\mathrm{T}} \tag{A.630}$$

schreiben, wobei $\mathsf{G} \in \mathbb{R}^{N \times N}$ eine eindeutig bestimmte untere Dreiecksmatrix ist, die positive Diagonalelemente besitzt. Diese Zerlegung wird auch *Cholesky-Zerlegung* genannt. Diese Zerlegung spielt eine große Rolle in der Lösung linearer Gleichungssysteme.

A.5. Eigenwerte

Eigenwertprobleme stellen eine wichtige Problemklasse dar und tauchen vielen technischen Zusammenhängen auf (zum Beispiele bei der Bestimmung vor Resonanzfrequenzen von elastischen Strukturen und der zugehörigen Schwingungsformen). Ein gewöhnliches Eigenwert Problem für eine gegebene quadratische $n \times n$-Matrix A lautet

$$\mathsf{A} \cdot \vec{x} = \lambda \vec{x}. \tag{A.631}$$

Die Aufgabe besteht darin, komplexe Zahlen λ (*Eigenwert*) komplexe Vektoren $\vec{x}$ (*Eigenvektor*) zu finden, so daß (A.631) erfüllt ist. Man sucht dabei nicht-triviale Lösungen $\vec{x} \neq 0$, denn für $\vec{x} = 0$ ist die Gleichung immer und für beliebige Werte von λ erfüllt. Nicht-triviale Eigenvektoren sind nur bis auf einen unbestimmten skalaren Faktor bestimmt. Die Menge alle Eigenwerte $\{\lambda_i\}$ einer Matrix A wird auch *Spektrum* von A genannt. Die Menge aller Eigenvektoren $\{\vec{x}_i\}$ von A wird *Eigensystem* genannt.

Das Eigenwertproblem (A.631) kann man auch als homogenes lineares Problem schreiben

$$(\mathsf{A} - \lambda \mathsf{I}) \cdot \vec{x} = 0. \tag{A.632}$$

Oben hatte wir für reguläre Matrizen A gesehen, daß $\mathsf{A} \cdot \vec{x} = 0$ nur die triviale Lösung $\vec{x} = 0$ besitzt $\Leftrightarrow \det \mathsf{A} \neq 0$ ist. Die Negation besagt, daß mindestens eine nichttriviale Lösung $\vec{x} \neq 0$ existiert $\Leftrightarrow \det \mathsf{A} = 0$. Übertragen auf das Eigenwertproblem (A.632) bedeutet dies, daß es mindestens einen nichttrivialen Eigenvektor geben muß, genau dann wenn

$$\det (\mathsf{A} - \lambda \mathsf{I}) = 0. \tag{A.633}$$

Dies wird auch *Lösbarkeitsbedigung* des homogenen linearen Systems (A.632) genannt. Die Determinante ist ein Polynom n-ten Grades in λ. Man nennt dieses Polynom das *charakteristische Polynom*. Die Eigenwerte λ sind danach die Nullstellen des charakteristischen Polynoms. Als Beispiel betrachten wir eine 2×2-Matrix. Dann haben wir

$$\det (\mathsf{A} - \lambda \mathsf{I}) = \det \begin{pmatrix} A_{11} - \lambda & A_{12} \\ A_{21} & A_{22} - \lambda \end{pmatrix} = (A_{11} - \lambda)(A_{22} - \lambda) - A_{12} A_{21}. \tag{A.634}$$

Wie man sieht, ergeben sich die möglichen Werte von λ in diesem Fall als Nullstellen eines Polynoms zweiten Grades in λ.

Aus der Algebra ist bekannt, daß ein Polynom vom Grade n genau n Wurzeln (Nullstellen) besitzt, die reell oder komplex sein können. Die Wurzeln müssen nicht alle verschieden sein. Tritt eine Wurzel mehrfach auf, so sagt man, daß der entsprechende Eigenwert *entartet* ist.[4]

A.5.1. Eigenwerte und Determinante

Da die Determinante $\det (\mathsf{A} - \lambda \mathsf{I})$ das charakteristische Polynom ist, welches die Eigenwerte λ_i als Wurzeln hat, können wir schreiben

$$\det (\mathsf{A} - \lambda \mathsf{I}) = (\lambda_1 - \lambda) \ldots (\lambda_n - \lambda). \tag{A.635}$$

Wenn wir in dieser Gleichung $\lambda = 0$ setzen, erhalten wir die wichtige Beziehung

$$\det \mathsf{A} = \lambda_1 \ldots \lambda_n. \tag{A.636}$$

[4] Die Einheitsmatrix I besitzt nur den Eigenwert $\lambda = 1$, der aber n-fach entartet ist. Jeder Vektor ist ein Eigenvektor von I. Falls man die Eigenvektoren auf 1 normiert und orthogonal wählt, besteht das Eigensystem aus den n Einheitsvektoren.

Die Determinante einer Matrix ist gleich dem Produkt ihrer Eigenwerte. Daran erkennt man sofort, daß die Determinante von Null verschieden ist, genau dann wenn *alle* Eigenwerte von Null verschieden sind. Ist nur ein Eigenwert Null, so ist auch die Determinante Null. Bezogen auf die oben betrachtete Regularität einer Matrix können wir damit feststellen:

> Eine reelle oder komplexe $n \times n$-Matrix A ist genau dann regulär, wenn alle ihre Eigenwerte $\lambda_i \neq 0$ von Null verschieden sind.

A.5.2. Eigenwerte spezieller Matrizen

Eigenwerte der Transponierten

Wegen

$$\det\left(\mathsf{A} - \lambda\mathsf{I}\right) = \det\left(\mathsf{A} - \lambda\mathsf{I}\right)^{\mathrm{T}} = \det\left(\mathsf{A}^{\mathrm{T}} - \lambda\mathsf{I}\right) = 0 \tag{A.637}$$

besitzt A^{T} dieselben Eigenwerte wie A. Die Eigenvektoren von A^{T} sind aber im allgemeinen verschieden von denen von A. Mit der Existenz von Eigenvektoren von A^{T}, d.h. aus

$$\mathsf{A}^{\mathrm{T}} \cdot \vec{x} = \lambda\vec{x} \tag{A.638a}$$

folgt durch Transponieren

$$\left(\mathsf{A}^{\mathrm{T}} \cdot \vec{x}\right)^{\mathrm{T}} = \lambda\vec{x}^{\mathrm{T}} \tag{A.638b}$$

bzw.

$$\vec{x}^{\mathrm{T}} \cdot \mathsf{A} = \lambda\vec{x}^{\mathrm{T}}. \tag{A.638c}$$

Hierbei ist zu beachten, daß $\vec{x}^{\mathrm{T}}$ ein Zeilenvektor ist. Wenn man sich nicht sicher ist, kann man die Umformungen auch unter Verwendung der *Indexnotation* durchführen.[5]

Eigenwerte von Dreicksmatrizen

Da die Determinante einer Dreicksmatrix oder einer Diagonalmatrix einerseits aus dem Produkt der Diagonalelemente besteht und andererseits das Produkt der Eigenwerte ist, stehen die Eigenwerte bei diesen speziellen Matrizen auf der Diagonale.

---

[5] $$\sum_i A^{\mathrm{T}}_{ij} x_i = \lambda x_j \quad \Rightarrow \quad \sum_i A_{ji} x_i = \lambda x_j \quad \Rightarrow \quad \sum_i A^{\mathrm{T}}_{ji} x^{\mathrm{T}}_i = \lambda x^{\mathrm{T}}_j \quad \Rightarrow \quad \sum_i A_{ij} x^{\mathrm{T}}_i = \lambda x^{\mathrm{T}}_j.$$

Eigenwerte der Inversen

Für reguläre Matrizen A kann man einen Zusammenhang herstellen zwischen deren Eigenwerten λ und denjenigen der Inversen A^{-1}. Denn mit $\mathsf{A}\cdot\vec{x}=\lambda\vec{x}$ folgt dann durch Multiplikation von links mit A^{-1}

$$\vec{x}=\mathsf{A}^{-1}\cdot\mathsf{A}\cdot\vec{x}=\lambda\mathsf{A}^{-1}\cdot\vec{x}\qquad\Rightarrow\qquad\mathsf{A}^{-1}\cdot\vec{x}=\frac{1}{\lambda}\vec{x}. \tag{A.639}$$

Die Eigenwerte der Inversen sind also die inversen Eigenwerte und die Eigenvektoren von A^{-1} sind identisch mit denjenigen von A.

Eigenwerte symmetrischer Matrizen

Eigenwerte reeller Matrizen können reell oder komplex sein. Dies ist klar, wenn man daran denkt, daß das charakteristische Polynom mit reellen Koeffizienten durchaus komplexe Wurzeln haben kann. Wenn ein Eigenwert einer reellen Matrix komplex ist, dann ist es auch der zugehörige Eigenvektor.

Wir betrachten nun eine reelle symmetrische Matrix. Für sie gilt $\mathsf{A}^{\mathrm{T}}=\mathsf{A}$. Wenn wir nun die Eigenwertgleichung (A.631) für A von links mit dem adjungierten (transponierten und konjugiert komplexen) Vektor $\vec{x}^{\dagger}$ multiplizieren, erhalten wir

$$\vec{x}^{\dagger}\cdot\mathsf{A}\cdot\vec{x}=\lambda\vec{x}^{\dagger}\cdot\vec{x}=\lambda\underbrace{\|\vec{x}\|^2}_{>0}. \tag{A.640}$$

Da

$$\left(\vec{x}^{\dagger}\cdot\mathsf{A}\cdot\vec{x}\right)^{\dagger}=\vec{x}^{\dagger}\cdot\mathsf{A}^{\dagger}\cdot\left(\vec{x}^{\dagger}\right)^{\dagger}=\vec{x}^{\dagger}\cdot\mathsf{A}\cdot\vec{x}, \tag{A.641}$$

ist die linke Seite von (A.640) reell. Da auch $\vec{x}^{\dagger}\cdot\vec{x}\in\mathbb{R}$, muß λ reell sein. Also gilt:

> Eine symmetrische reelle Matrix hat reelle Eigenwerte. Man kann dann auch die Eigenvektoren reell wählen.

Entscheidend für die reellen Eigenwerte war, daß $\mathsf{A}^{\dagger}=\mathsf{A}$ ist. Derartige Matrizen, die komplex sein dürfen, nennt man *selbstadjungiert*. Allgemein gilt, daß selbstadjungierte Matrizen reelle Eigenwerte besitzen.

Eine Matrix heißt *schiefsymmetrisch* (*skew-symmetric*), wenn gilt $\mathsf{A}^{\mathrm{T}}=-\mathsf{A}$. Dann gilt für eine reelle schiefsymmetrische Matrix

$$\left(\vec{x}^{\dagger}\cdot\mathsf{A}\cdot\vec{x}\right)^{\dagger}=\vec{x}^{\dagger}\cdot\mathsf{A}^{\dagger}\cdot\left(\vec{x}^{\dagger}\right)^{\dagger}=-\vec{x}^{\dagger}\cdot\mathsf{A}\cdot\vec{x}. \tag{A.642}$$

Mit (A.640) ist dann

$$-\mathsf{A}\cdot\vec{x}=\lambda^{*}\vec{x}, \tag{A.643}$$

also $\lambda=-\lambda^{*}$. Dies bedeutet aber, daß λ rein imaginär ist.

Eine schiefsymmetrische reelle Matrix hat rein imaginäre Eigenwerte.

Sei A eine symmetrische positiv definite Matrix (siehe (A.625)) und λ ein Eigenwert mit zugehörigem Eigenvektor, dann ist

$$\lambda = \frac{\overbrace{\vec{x}^{\mathrm{T}} \cdot \mathsf{A} \cdot \vec{x}}^{>0}}{\vec{x}^{\mathrm{T}} \cdot \vec{x}} > 0. \tag{A.644}$$

Wir folgern

Alle Eigenwerte einer positiv definiten symmetrischen Matrix sind positiv.

Außerdem kann man zeigen:

Die Eigenwerte einer reellen symmetrischen $n \times n$-Matrix sind genau dann

- positiv, wenn A positiv definit ist,
- nicht negativ, wenn A positiv semi-definit ist,
- negativ, wenn A negativ definit ist,
- nicht positiv, wenn A negativ semi-definit ist,

A.5.3. Ähnlichkeitstransformationen

Unter einer *Ähnlichkeitstransformation* einer Matrix $\mathsf{A} \to \mathsf{B}$ versteht man eine Transformation der Form

$$\mathsf{B} = \mathsf{P} \cdot \mathsf{A} \cdot \mathsf{P}^{-1}, \tag{A.645}$$

mit Hilfe einer regulären (daher invertierbaren) Transformationsmatrix P. Die Ähnlichkeitstransformation spielt eine Rolle bei einer beliebigen Koordinatentransformation $\vec{x} \to \vec{x}' = \mathsf{P} \cdot \vec{x}$. Denn wenn man von dem Problem

$$\mathsf{A} \cdot \vec{x} = \vec{c} \tag{A.646}$$

ausgeht, erhält man das transformierte System $\mathsf{B} \cdot \vec{x}' = \vec{c}'$, indem man (A.646) von links mit P multipliziert und die Identität $\mathsf{I} = \mathsf{P}^{-1}\mathsf{P}$ einfügt

$$\underbrace{\mathsf{P} \cdot \mathsf{A} \cdot \overbrace{\mathsf{P}^{-1}}^{=\mathsf{I}}}_{\mathsf{B}} \cdot \underbrace{\mathsf{P} \cdot \vec{x}}_{\vec{x}'} = \underbrace{\mathsf{P} \cdot \vec{c}}_{\vec{c}'} \tag{A.647}$$

Eine wichtige Eigenschaft der Ähnlichkeitstransformation ist es, daß sich die Eigenwerte nicht ändern. D.h. $\mathsf{B} = \mathsf{P} \cdot \mathsf{A} \cdot \mathsf{P}^{-1}$ besitzt dieselben Eigenwerte wie A, denn das charakterisitische Polynom ist identisch

$$\begin{aligned} \det(\mathsf{A} - \lambda\mathsf{I}) &= \det\left(\mathsf{P} \cdot \mathsf{P}^{-1}\right) \det(\mathsf{A} - \lambda\mathsf{I}) = \det \mathsf{P} \det(\mathsf{A} - \lambda\mathsf{I}) \det \mathsf{P}^{-1} \\ &= \det\left(\mathsf{P} \cdot \mathsf{A} \cdot \mathsf{P}^{-1} - \lambda\mathsf{I}\right) = \det(\mathsf{B} - \lambda\mathsf{I}). \end{aligned} \tag{A.648}$$

Obwohl die Eigenwerte unverändert bleiben, ändern sich die Eigenvektoren! Sie transformieren sich wie alle Vektoren durch Multiplikation mit P.

A.5.4. Hauptachsentransformation (Diagonalisierung)

Die Ähnlichkeitstransformation kann man ausnutzen, um ein lineares Problem $\mathsf{A} \cdot \vec{x} = \vec{c}$ in eine Form zu bringen, in der die Matrix einfacher (am besten diagonal) ist. Dies gelingt, wenn die $n \times n$-Matrix n linear unabhängige Eigenvektoren $\vec{x}_i$ besitzt. In diesem Fall bildet man die Transformationsmatrix P als Zeilenvektor bestehend aus den Eigenvektoren

$$\mathsf{P} := (\vec{x}_1, \ldots, \vec{x}_n). \tag{A.649}$$

Dann wird

$$\begin{aligned} \mathsf{A} \cdot \mathsf{P} &= \mathsf{A} \cdot (\vec{x}_1, \ldots, \vec{x}_n) = (\mathsf{A} \cdot \vec{x}_1, \ldots, \mathsf{A} \cdot \vec{x}_n) = (\lambda_1 \vec{x}_1, \ldots, \lambda_n \vec{x}_n) \\ &= (\vec{x}_1, \ldots, \vec{x}_n) \cdot \mathsf{D} = \mathsf{P} \cdot \mathsf{D}, \end{aligned} \tag{A.650}$$

wobei

$$\mathsf{D} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \tag{A.651}$$

eine Diagonalmatrix ist, auf deren Diagonale genau die Eigenwerte stehen. Damit läßt sich A darstellen als

$$\mathsf{A} = \mathsf{P} \cdot \mathsf{D} \cdot \mathsf{P}^{-1}. \tag{A.652}$$

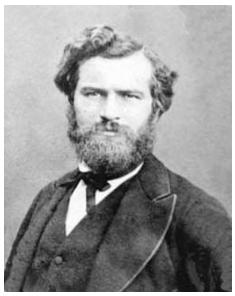
Die Matrizen A und D sind also ähnlich. Die Koordinatentransformation (A.649) erlaubt es also die Matrix A zu diagonalisieren.

A.5.5. Jordan-Normalform

Wenn eine $n \times n$-Matrix weniger als n linear unabhängige Eigenvektoren besitzt, dann kann man zeigen, daß gewisse Eigenwerte mehrfach auftreten (Multiplizität von Eigenwerte). Die Umkehrung gilt nicht: Es kann nämlich sein, daß Eigenwerte mehrfach auftreten, obwohl alle n Eigenvektoren linear unabhängig sind (Beispiel: I). Ein einfaches Beispiel für eine Matrix, die keine n linear unabhängigen Eigenvektoren besitzt ist

$$\mathsf{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}. \tag{A.653}$$

Denn der Eigenwert $\lambda_{1,2} = 1$ ist 2-fach entartet und der einzige Eigenvektor ist $(1, 0)^{\mathrm{T}}$. Daher läßt sich A nach (A.653) nicht diagonalisieren.



Marie Ennemond Camille Jordan 1838–1922

Allgemein ist es aber möglich, *jede* $n \times n$-Matrix durch eine Ähnlichkeitstransformation auf die sogenannte *Jordan-Normalform* zu bringen (ohne Beweis)

$$\mathsf{J} = \begin{pmatrix} \lambda_1 & \delta_1 & & & \\ & \lambda_2 & \delta_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \delta_{n-1} \\ & & & & \lambda_n \end{pmatrix}. \tag{A.654}$$

Hierbei sind λ_i die Eigenwerte und $\delta_i \in \{0, 1\}$, wobei

$$\delta_i = \begin{cases} 1 & \Leftrightarrow \quad \lambda_i = \lambda_{i+1} \\ 0 & \Leftrightarrow \quad \lambda_i \neq \lambda_{i+1}. \end{cases} \tag{A.655}$$

Hierbei kann man die Eigenwerte, die entartet sind, allesamt in sogenannte *Jordan-Blöcke* zusammenfassen. In dem Beispiel

$$\mathsf{J} = \left(\begin{array}{ccc|c|cc|ccc} 4 & 1 & & & & & & & \\ & 4 & 1 & & & & & & \\ & & 4 & & & & & & \\ \hline & & & -3 & & & & & \\ \hline & & & & \mathrm{i} & 1 & & & \\ & & & & & \mathrm{i} & & & \\ \hline & & & & & & 6 & & \\ & & & & & & & 4.2 & \\ & & & & & & & & -9 \end{array}\right) \tag{A.656}$$

sind die Jordan-Blöcke rot dargestellt. Der Eigenwert $\lambda_1 = 4$ ist dreifach und der Eigenwert $\lambda_3 = \mathrm{i}$ ist 2-fach entartet. Oft wird dann die Form

$$\mathsf{J} = \begin{pmatrix} \mathsf{J}_1 & & \\ & \ddots & \\ & & \mathsf{J}_p \end{pmatrix} \tag{A.657}$$

gewählt, wobei $p < n$ und die Block-Matrizen J_i aus Jordan-Böcken bestehen mit identischer Diagonale (Eigenwerte) und Einsen auf der ersten Nebendiagonale. Man sieht nun, daß das Beispiel (A.653) aus einem kleinen Jordan-Block besteht.

A.5.6. Transformation auf eine Dreiecksmatrix

Die Jordansche Normalform ist von theoretischer, aber nicht von großer praktischer Bedeutung. Aus numerischen Gründen ist es wünschenswert, mit unitären oder orthogonalen Matrizen zu arbeiten.

Mit unitären oder orthogonalen Matrizen kann man eine Matrix nicht immer auf eine Diagonalform bringen. Man kann sie aber auf Dreiecksmatrizen transformieren. Es gilt (ohne Beweis)

Satz von Schur: Für jede beliebige $n \times n$-Matrix A existiert eine unitäre Matrix U (dann ist $\mathsf{U}^{-1} = \mathsf{U}^\dagger$, siehe (A.612)), so daß

$$\mathsf{U} \cdot \mathsf{A} \cdot \mathsf{U}^{-1} = \mathsf{U} \cdot \mathsf{A} \cdot \mathsf{U}^\dagger = \mathsf{T}, \tag{A.658}$$

wobei T eine Dreiecksmatrix (*triangular matrix*) ist.

Für orthogonale Transformationen gilt

Satz von Murnaghan-Wintner: Für jede beliebige $n \times n$-Matrix A existiert eine orthogonale Matrix P (dann ist $\mathsf{P}^{-1} = \mathsf{P}^{\mathrm{T}}$, siehe (A.610)), so daß

$$\mathsf{P} \cdot \mathsf{A} \cdot \mathsf{P}^{-1} = \mathsf{P} \cdot \mathsf{A} \cdot \mathsf{P}^{\mathrm{T}} = \mathsf{T}, \tag{A.659}$$

wobei

$$\mathsf{T} = \begin{pmatrix} T_{11} & \dots & T_{1m} \\ & \ddots & \vdots \\ & & T_{mm} \end{pmatrix} \tag{A.660}$$

eine $m \times m$ Dreiecksmatrix ist ($m \leq n$), die aus 1×1- oder 2×2-Blöcken besteht.

Issai Schur
1875–1941

Beim Satz von Schur sind die Diagonalelemente von T identisch mit den Eigenwerten von A, da U eine Ähnlichkeitstransformation ist. Wenn A reell ist und komplexe Eigenwerte hat, so benötigt man zu dieser Ähnlichkeitstransformation komplexe Transformationsmatrizen U.

Will man nur mit reellen orthogonalen Transformationsmatrizen arbeiten, so stellt der Satz von Murnaghan-Wintner sicher, daß es eine Transformation gibt, welche eine reelle Matrix A beinahe in eine Dreiecksmatrix überführt. Die reellen Eigenwerte von A stehen dann in den 1×1-Böcken von T_{ii}. Die 2×2-Blöcke von T_{ii} besitzen dann je einen Satz konjugiert-komplexer Eigenwerte, die mit den konjugiert-komplexen Eigenwerte von A übereinstimmen.

A.5.7. Singulärwertzerlegung

Wenn man nur orthogonale oder unitäre Ähnlichkeitstransformationen verwenden möchte, um A in eine einfachere Gestalt zu bringen, so geben die obigen Sätze von Schur und von Murnaghan-Wintner die einfachsten allgemeinen Formen an, auf welche man A transformieren kann.

Wenn man jedoch auf eine Ähnlichkeitstransformation verzichtet, kann man *jede reelle* Matrix mittels zweier orthogonaler Matrizen auf Diagonalform bringen. Nur

stehen auf der Diagonale der Diagonalmatrix dann eben nicht mehr die Eigenwerte. Man nennt dies *Singulärwertzerlegung* (SVD).[6]

> *Singulärwertzerlegung*: Für jede reelle Matrix A gibt es orthogonale Matrizen U und V (also mit $\mathsf{U}^{-1} = \mathsf{U}^{\mathrm{T}}$ und $\mathsf{V}^{-1} = \mathsf{V}^{\mathrm{T}}$), so daß
>
> $$\mathsf{A} = \mathsf{U} \cdot \mathsf{D} \cdot \mathsf{V} \tag{A.661}$$
>
> mit $\mathsf{D} = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ wobei für alle i gilt $\sigma_i \geq 0$.

Die Größen σ_i werden *Singulärwerte* von A genannt. Der Zusammenhang zwischen den Singulärwerten von A und gewissen Eigenwerten ergibt sich aus $\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A}$

$$\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A} = \mathsf{V}^{\mathrm{T}} \cdot \mathsf{D} \cdot \underbrace{\mathsf{U}^{\mathrm{T}} \cdot \mathsf{U}}_{\mathsf{I}} \cdot \mathsf{D} \cdot \mathsf{V} = \mathsf{V}^{\mathrm{T}} \cdot \mathsf{D}^2 \cdot \mathsf{V}. \tag{A.662}$$

Die rechte Seite stellt eine Ähnlichkeitstransformation dar, und zwar von $\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A}$. Das Quadrat $\mathsf{D}^2 = \mathsf{D} \cdot \mathsf{D}$ ist, wie auch D, diagonal. Auf der Diagonale von D^2 stehen die Quadrate σ_i^2 der Singulärwerte von A. Daher sind die σ_i^2 die Eigenwerte von $\mathsf{A}^{\mathrm{T}} \cdot \mathsf{A}$.

Man kann zeigen, daß die Anzahl der nicht-verschwindenden Singulärwerte gleich dem Rang der Matrix ist. Auch läßt sich die Singulärwertzerlegung auf komplexe Matrizen erweitern. Dann sind U und V unitär. Die Singulärwerte σ_i bleiben aber reell.[7]

A.6. Normen

Um Vektoren miteinander vergleichen zu können, benötigt man ein Maß. Wir bezeichnen das Maß eines Vektors $\vec{x}$ als *Norm* $\|\vec{x}\|$. Dieses Maß sollte sinnvollerweise die folgenden Eigenschaften besitzen

$$\|\vec{x}\| \geq 0, \quad \text{für alle } \vec{x} \text{ und } \|0\| = 0, \tag{A.663a}$$
$$\|\alpha\vec{x}\| = |\alpha|\|\vec{x}\|, \quad \text{für alle } \alpha, \tag{A.663b}$$
$$\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|, \quad \text{für alle } \vec{x} \text{ und } \vec{y}. \tag{A.663c}$$

Die Eigenschaft (A.663c) heißt *Dreiecksungleichung*.

[6] ***S**ingular **V**alue **D**ecomposition.*

[7] Motiviert durch den Ausruf von George Forsythe (1917–1972): *'Will somebody please figure out how to compute the pseudo-inverse of a matrix?'* entwickelte Gene Golub 1965 einen stabilen Algorithmus zur Berechnung der Singuärwert-Zerlegung (SVD). Dieser hat sich zu einem der nützlichsten Algorithmen der Numerik entwickelt.

Euclid von Alexandrien ca. 325–265 v.Ch.

Die bekanntest Norm ist die *Eukilidische Norm* (l_2-Norm)

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{A.664}$$

Sie ist ein Spezialfall einer sogenannten *l_p-Norm*

$$\|\vec{x}\|_p = \left(\sum_{i=1}^{n} |x_i^p| \right)^{1/p}, \qquad p \in \mathbb{R}. \tag{A.665}$$

Im Grenzfall $p \to \infty$ erhält man die *Maximumnorm* l_∞

$$\|\vec{x}\|_\infty = \lim_{p\to\infty} \left(\sum_{i=1}^{n} |x_i^p| \right)^{1/p} = \max_{1\le i\le n} |x_i| \,. \tag{A.666}$$

In diesem Fall überlebt nur die Komponente von $\vec{x}$ mit dem größten Betrag.

Neben der Vektor-Norm ist auch die *Matrix-Norm* von Bedeutung. Sie wird definiert als

$$\|\mathsf{A}\| := \max_{\vec{x}\neq 0} \frac{\|\mathsf{A}\cdot\vec{x}\|}{\|\vec{x}\|} = \max_{\|\vec{x}\|=1} \|\mathsf{A}\cdot\vec{x}\| \,. \tag{A.667}$$

Geometrisch kann die Matrix-Norm gedeutet werden als die größte Länge, die $\mathsf{A}\cdot\vec{x}$ mit $\|\vec{x}\| = 1$ annehmen kann.

B. Bemerkungen zur Gauß-Quadratur

Eine streng mathematische Ableitung der Gauß-Quadratur kann man in Freund et al. (2007) finden. Hier soll nur kurz gezeigt werden, daß die Gauß-Quadratur

$$\int_{-1}^{1} w(x) f(x)\,\mathrm{d}x = \sum_{i=1}^{n} a_i f(x_i) \tag{B.668}$$

für jedes Polynom m-ten Grades $f(x) = p(x)$ mit $m \leq 2n-1$ exakt ist.

Um dies zu zeigen, setzen wir voraus, daß die Gewichte a_i definiert sind durch das lineare Gleichungssystem

$$\sum_{i=1}^{n} p_k(x_i) a_i = \begin{cases} \langle p_0 | p_0 \rangle\,, & \text{falls} \quad k = 0, \\ 0, & \text{falls} \quad k = 1, 2, \ldots, n-1, \end{cases} \tag{B.669}$$

wobei $p_k(x)$ ein orthogonales Polynom vom Grade k ist und

$$\langle f(x) | g(x) \rangle = \int_{-1}^{1} w(x) f(x) g(x)\,\mathrm{d}x \tag{B.670}$$

das zugehörige Skalarprodukt mit Gewichtsfunktion $w(x)$. Wir betrachten nun die allgemeine Darstellung eines beliebigen Polynoms der Ordnung $m \leq 2n-1$ in der Form

$$p(x) = p_n(x) q(x) + r(x), \tag{B.671}$$

wobei $p_n(x)$ ein orthogonales Polynom vom Grade n ist und

$$q(x) = \sum_{k=0}^{n-1} \alpha_k p_k(x) \qquad \text{und} \quad r(x) = \sum_{k=0}^{n-1} \beta_k p_k(x). \tag{B.672}$$

Wenn wir dieses allgemeine Polynom $p(x)$ von Grade $m \leq 2n-1$ mit der Gewichtsfunktion $w(x)$ integrieren, erhalten wir einerseits

$$\begin{aligned} \int_{-1}^{1} w(x) p(x)\,\mathrm{d}x &\overset{p_0=1}{=} \langle p_n(x) q(x) + r(x) | p_0 \rangle = \underbrace{\langle p_n(x) | q(x) \rangle}_{=0} + \langle r(x) | p_0 \rangle \\ &= \langle r(x) | p_0 \rangle = \beta_0 \langle p_0 | p_0 \rangle\,. \end{aligned} \tag{B.673}$$

Der Term $\langle p_n(x)|q(x)\rangle = 0$ verschwindet, weil $q(x)$ vom Grade $n-1$ ist und damit orthogonal zu dem orthogonalen Polynom n-ter Ordnung $p_n(x)$. Andererseits gilt wegen (B.669)

$$\sum_{i=1}^{n} a_i p(x_i) \overset{p_n(x_i)=0}{=} \sum_{i=1}^{n} a_i r(x_i) = \sum_{i=1}^{n} a_i \left(\sum_{k=0}^{n-1} \beta_k p_k(x) \right) = \sum_{k=0}^{n-1} \beta_k \left(\sum_{i=1}^{n} a_i p_k(x) \right)$$
$$= \beta_0 \langle p_0|p_0\rangle . \tag{B.674}$$

Mit (B.673) und (B.674) ist

$$\int_{-1}^{1} w(x)p(x)\,\mathrm{d}x = \sum_{i=1}^{n} a_i p(x_i), \tag{B.675}$$

für jedes Polynom $p(x)$, das höchstens vom Grade $2n-1$ ist. Damit ist die Behauptung bewiesen.

C. Koeffizienten der A-Orthogonalisierung für das CG-Verfahren

Um zu zeigen, daß fast alle Koeffizienten $\beta_{n+1,l}$ (7.533) in (7.532) verschwinden, sind ein paar Vorbetrachtungen nötig. Zunächst ist es nützlich den Fehlervektor $\vec{e}$ der Iteration zu definieren. Mit $\vec{e}^{(n)} = \vec{x}^* - \vec{x}^{(n)}$, wobei $\vec{x}^*$ die exakte Lösung des linearen Problems ist, gilt

$$\mathsf{A} \cdot \vec{e}^{(n)} = \underbrace{\mathsf{A} \cdot \vec{x}^*}_{\vec{b}} - \mathsf{A} \cdot \vec{x}^{(n)} = \vec{b} - \mathsf{A} \cdot \vec{x}^{(n)} = \vec{\rho}^{(n)}. \tag{C.676}$$

Wenn man nun zu Beginn der Rechnung den Fehler durch die konjugierten Richtungen darstellt

$$\vec{e}^{(0)} = \sum_{j=0}^{N-1} \delta_j \vec{p}^{(j)}, \tag{C.677}$$

dann folgt

$$\vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{e}^{(0)} = \sum_{j=0}^{N-1} \delta_j \vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(j)} = \delta_n \vec{p}^{(n)} \cdot \mathsf{A} \cdot \vec{p}^{(n)} \tag{C.678}$$

Damit ergibt sich, daß der Fehler nur in dem Komplement des Krylov-Raumes $\mathcal{V}_n = \text{span}\{\vec{p}^{(0)}, \ldots, \vec{p}^{(n-1)}\}$ lebt

$$\vec{e}^{(n)} = \ldots = \sum_{j=n}^{N-1} \delta_j \vec{p}^{(j)}. \tag{C.679}$$

Hieraus folgt wiederum, daß das Residuum $\vec{\rho}^{(n)}$ senkrecht ist zu $\mathcal{V}_n$, denn für $i < n$ gilt

$$\vec{p}^{(i)} \cdot \vec{\rho}^{(n)} = -\vec{p}^{(i)} \cdot \mathsf{A} \cdot \vec{e}^{(n)} = -\sum_{j=n}^{N-1} \delta_j \underbrace{\vec{p}^{(i)} \cdot \mathsf{A} \cdot \vec{p}^{(j)}}_{=0, \text{ für } i<j} = 0. \tag{C.680}$$

Damit können wir zeigen, daß Koeffizienten $\beta_{n+1,l}$ für $l < n$ verschwindet. Es reicht, den Zähler von (7.533), $\vec{p}^{(l)} \cdot \mathsf{A} \cdot \vec{\rho}^{(n+1)}$ zu betrachten. Dazu gehen wir aus von

$$\vec{\rho}^{(l+1)} = \vec{\rho}^{(l)} - \alpha^{(l)} \mathsf{A} \cdot \vec{p}^{(l)} \tag{C.681}$$

Damit gilt

$$\vec{\rho}^{(n+1)} \cdot \vec{\rho}^{(l+1)} = \vec{\rho}^{(n+1)} \cdot \left(\vec{\rho}^{(l)} - \alpha^{(l)} \mathsf{A} \cdot \vec{p}^{(l)}\right) = \vec{\rho}^{(n+1)} \cdot \vec{\rho}^{(l)} - \alpha^{(l)} \vec{\rho}^{(n+1)} \cdot \mathsf{A} \cdot \vec{p}^{(l)}. \quad \text{(C.682)}$$

Für den Zähler von (7.533) folgt also

$$\vec{\rho}^{(n+1)} \cdot \mathsf{A} \cdot \vec{p}^{(l)} = \frac{\vec{\rho}^{(n+1)} \cdot \vec{\rho}^{(l)} - \vec{\rho}^{(n+1)} \cdot \vec{\rho}^{(l+1)}}{\alpha^{(l)}} \quad \text{(C.683)}$$

Für $l \leq n$ verschwindet der erste Summand im Zähler, weil $\vec{\rho}^{(n)} \in \mathcal{V}_n$ und $\vec{\rho}^{(n)} \perp \mathcal{V}_n$. Der zweite Summand, und damit die gesamte rechte Seite verschwindet für $l < n$. Für $l = n$ verbleibt lediglich

$$\vec{\rho}^{(n+1)} \cdot \mathsf{A} \cdot \vec{p}^{(n)} = -\frac{\vec{\rho}^{(n+1)} \cdot \vec{\rho}^{(n+1)}}{\alpha^{(n)}}. \quad \text{(C.684)}$$

Dies sollte derselbe Term wie in (7.534) sein.

Literaturverzeichnis

Abramowitz, M. and Stegun, I. A. (1972), *Handbook of Mathematical Functions*, Dover.

Boyd, J. P. (2000), *Chebyshev and Fourier Spectral Methods*, Dover, 31 East 2nd Street Mineola, New York 11501.

Canuto, C., Hussaini, M. Y., Quarteroni, A. and Zhang, T. A. (1988), *Spectral Methods in Fluid Dynamics*, Springer.

Dahmen, W. and Reusken, A. (2006), *Numerik für Ingenieure und Naturwissenschaftler*, Springer, Berlin, Heidelberg.

Fletcher, C. A. J. (1991), *Computational Techniques for Fluid Dynamics*, Vol. I of *Springer Series in Computational Physics*, Springer.

Freund, R. W., Hoppe, R. H. W., Stoer, J. and Burlisch, R. (2007), *Stoer/Bulirsch: Numerische Mathematik 1*, Springer, Berlin, Heidelberg.

Golub, G. and Ortega, J. M. (1996), *Scientific Computing - Eine Einführung in das wissenschaftliche Rechnen und parallele Numerik*, Teubner, Stuttgart.

Golub, H. G. and van Loan, H. G. (1989), *Matrix Computations*, Johns Hopkins University Press.

Guckenheimer, J. and Holmes, P. (1983), *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Vol. 42 of *Applied Mathematical Sciences*, Springer.

Hanke-Bourgeois, M. (2002), *Grundlagen der numerischen Mathematik und des wissenschaftlichen Rechnens*, Teubner, Wiesbaden.

Hestenes, M. R. and Stiefel, E. (1952), 'Methods of conjugate gradients for solving linear systems', *J. Res. Natnl Bureau Standards* **49**, 409–436.

Huckle, T. and Schneider, S. (2006), *Numerische Methoden*, Springer, Berlin, Heidelberg.

Keller, H. B. (1977), *Numerical solution of bifurcation and nonlinear eigenvalue problems*, Academic Press, New York, pp. 359–384.

Lorenz, E. N. (1963), 'Deterministic non-periodic flow', *J. Atmos. Sci.* **20**, 130–141.

Meerbergen, K., Spence, A. and Roose, D. (1994), 'Shift-invert and Cayley transforms for detection of eigenvalues with largest real part of nonsymmetric matrices', *BIT* **34**, 409–423.

Moler, C. B. (2004), *Numerical Computing with MATLAB*, Society for Industrial and Applied Mathematics SIAM, Philadelphia.

Ortega, J. M. and Rheinboldt, W. C. (1970), *Iterative solution of nonlinear equations in several variables*, Academic Press, New York.

Peaceman, D. W. and Rachford Jr., H. H. (1955), 'The numerical solution of parabolic and elliptic differential equations', *SIAM J.* **3**, 28–41.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992), *Numerical Receipes in C*, Cambridge University Press, Cambridge.

Quarteroni, A. and Saleri, F. (2006), *Wissenschaftliches Rechnen mit MATLAB*, Springer, Berlin, Heidelberg.

Roache, P. J. (2009), 'Perspective: Validation — what does it mean?', *ASME J. Fluids Eng.* **131**, 034503–1–034503–4.

Saad, Y. (2003), *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia.

Squire, W. and Trapp, G. (1998), 'Using complex variables to estimate derivatives of real functions', *SIAM Rev.* **40**, 110–112.

Stoer, J. and Burlisch, R. (2005), *Numerische Mathematik 2*, Springer, Berlin, Heidelberg.

Trefethen, L. N. and Bau, III, D. (1997), *Numerical linear algebra*, SIAM, Philadelphia.