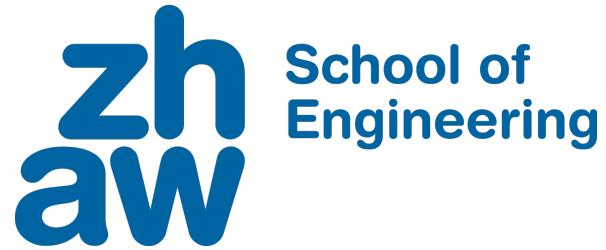


Zürcher Hochschule
für Angewandte Wissenschaften



Vorlesung Höhere Mathematik 1 & 2
Studiengang BSc. Informatik

Dr. sc. nat. Reto Knaack

19. Oktober 2022

Vorbemerkung

Das vorliegende Skript wurde, basierend auf einer früheren Version, für die ab Studienjahr 2020/21 stattfindenden Vorlesungen Höhere Mathematik 1 & 2 im Studiengang Informatik der ZHAW überarbeitet. Der Fokus liegt auf den für Informatik-Studierende relevanten Methoden der numerischen Mathematik. Neben der eigentlichen Anwendung der numerischen Algorithmen auf Problemstellungen liegt ein weiterer Fokus der Vorlesung auch auf der korrekten Implementation der Algorithmen mit Python, da damit das wissenschaftliche Programmieren geübt werden kann, was für Informatik-Studierende ebenfalls relevant ist.

Das Skript basiert auf den folgenden Quellen:

- [1] 'Numerische Mathematik: Eine beispielorientierte Einführung', M. Knorrenchil, Hanser Verlag (6. Auflage), 2017
- [2] 'Computermathematik', W. Gander, Birkhäuser, 1992
- [3] Skript 'Numerische Mathematik I' von Lars Grüne (Mathematisches Institut, Universität Bayreuth)
- [4] Skript 'Einführung in die Numerische Mathematik' von F. Natterer (Institut für Numerische und instrumentelle Mathematik, Universität Münster),
- [5] Skript 'Numerische Mathematik 1' von E. Novak (Universität Jena).
- [6] 'Numerik Algorithmen', G. Engeln-Müllges, K. Niederdrenk, R. Wodicka, Springer-Verlag (10. Auflage), 2011
- [7] 'Numerik für Informatiker', Huckle, Schneider, Springer, 2002
- [8] 'Mathematik für Ingenieure und Naturwissenschaftler: Band 2', L. Papula, Vieweg + Teubner Verlag (15. Auflage), 2018
- [9] 'Numerical Methods for Engineers and Scientists', A. Gilat, V. Subramaniam, Wiley, 2014
- [10] 'Numerische Mathematik', Folien zur Vorlesung, R. Massjung, 2013
- [11] 'Einführung in die Theorie der Fourierreihen und Fouriertransformationen', A. Spaenhauer, M. Steiner-Curtis, Skript, 2014
- [12] 'Mathematik für Ingenieure und Naturwissenschaftler: Band 1', L. Papula, Springer Vieweg (15. Auflage), 2018
- [13] 'Mathematik für Ingenieure 2', M. Knorrenchil, Hanser Verlag, 2014
- [14] 'Höhere Mathematik in Rezepten', C. Karpfinger, Springer (3. Auflage), 2017

Teilweise wurden Text und Abbildungen eins zu eins aus den obigen Quellen übernommen, insbesondere aus [1] und [7]. Das Skript ist deshalb nicht als eigenständiges Werk zu sehen, sondern als Unterrichts- und Arbeitshilfe, kondensiert aus einer Vielzahl anderer Skripte und Bücher.

Es wurde versucht, die behandelten Konzepte und Algorithmen in einen historischen Kontext zu setzen, um die Verbindungen der Numerischen Mathematik mit der Informatik und der Entwicklung des Computers hervorzuheben. Die Kapitel zur historischen Entwicklung sind in diesem Sinne als Hintergrundinformationen zu verstehen:

"It is an extremely useful thing to have knowledge of the true origins of memorable discoveries, especially those that have been found not by accident but by dint of meditation. It is not so much that thereby history may attribute to each man his own discoveries and others should be encouraged to earn like commendation, as that the art of making discoveries should be extended by considering noteworthy examples of it."¹

¹Leibniz, in the opening paragraph of his Historia et Origio Calculi Differentialis [1]. The quote itself was taken from Erik Meijering (2002), "A Chronology of Interpolation".

Inhaltsverzeichnis

1 Einführung in die Numerische Mathematik	5
1.1 Was ist Numerische Mathematik	5
1.2 Historische Entwicklung ²	6
1.3 Typische Fragestellungen	9
2 Rechnerarithmetik	13
2.1 Zur Geschichte der Zahlendarstellung ³	13
2.2 Maschinenzahlen	15
2.3 Approximations- und Rundungsfehler	18
2.3.1 Rundungsfehler und Maschinengenauigkeit	19
2.3.2 Fehlerfortpflanzung bei Funktionsauswertungen / Konditionierung	22
3 Numerische Lösung von Nullstellenproblemen	27
3.1 Zur historischen Entwicklung	27
3.2 Problemstellung	28
3.3 Fixpunktiteration	29
3.4 Das Newton-Verfahren	33
3.4.1 Vereinfachtes Newton-Verfahren	35
3.4.2 Sekantenverfahren	35
3.5 Konvergenzgeschwindigkeit	35
3.6 Fehlerabschätzung	36
4 Numerische Lösung linearer Gleichungssysteme	39
4.1 Zur historischen Entwicklung	39
4.2 Problemstellung	43
4.3 Der Gauss-Algorithmus	45
4.4 Fehlerfortpflanzung beim Gauss-Algorithmus und Pivotisierung	48
4.5 Dreieckszerlegung von Matrizen	49
4.5.1 Die LR-Zerlegung	49
4.5.1.1 Die LR-Zerlegung mit Zeilenumtauschung	51
4.5.2 Die QR-Zerlegung ⁴	54
4.6 Fehlerrechnung und Aufwandabschätzung	61
4.6.1 Fehlerrechnung bei linearen Gleichungssystemen	61
4.6.2 Aufwandabschätzung ⁵	65
4.7 Iterative Verfahren	67
4.7.1 Das Jacobi-Verfahren	68
4.7.2 Das Gauss-Seidel-Verfahren	69
4.7.3 Konvergenz	71
4.8 Eigenwerte und Eigenvektoren	73
4.8.1 Einführung in die komplexen Zahlen	73
4.8.1.1 Grundbegriffe	73

²Hauptsächlich gemäss Kap. 1 aus [7], erweitert mit Zusatzinformationen aus Wikipedia.

³Übernommen in gekürzter und leicht abgeänderter Form von Kap. 1 und Kap. 5 aus [7]

⁴Kapitel hauptsächlich übernommen aus [1]

⁵Kapitel hauptsächlich übernommen aus [3]

4.8.1.2	Darstellungsformen	75
4.8.1.3	Grundrechenarten	76
4.8.1.4	Potenzieren und Radizieren / Fundamentalsatz der Algebra	78
4.8.2	Einführung in Eigenwerte und Eigenvektoren ⁶	80
4.8.2.1	Grundbegriffe	80
4.8.2.2	Eigenschaften von Eigenwerten	82
4.8.2.3	Eigenschaften von Eigenvektoren	84
4.8.3	Numerische Berechnung von Eigenwerten und Eigenvektoren	87
4.8.3.1	Das QR-Verfahren	89
4.8.3.2	Vektoriteration	91
5	Numerische Lösung nicht linearer Gleichungssysteme	93
5.1	Einleitendes Beispiel	93
5.2	Funktionen mit mehreren Variablen	94
5.2.1	Definition einer Funktion mit mehreren Variablen	94
5.2.2	Darstellungsformen	96
5.2.2.1	Analytische Darstellung	96
5.2.2.2	Darstellung durch Wertetabelle	96
5.2.2.3	Grafische Darstellung	96
5.2.3	Partielle Ableitungen	98
5.2.4	Linearisierung von Funktionen mit mehreren Variablen	101
5.3	Problemstellung zur Nullstellenbestimmung für nichtlineare Systeme	103
5.4	Das Newton-Verfahren für Systeme	104
5.4.1	Quadratisch-konvergentes Newton-Verfahren	104
5.4.2	Vereinfachtes Newton-Verfahren	106
5.4.3	Gedämpftes Newton-Verfahren	107
6	Ausgleichsrechnung	110
6.1	Zur historischen Entwicklung ⁷	110
6.2	Interpolation	113
6.2.1	Problemstellung	113
6.2.2	Polynominterpolation	114
6.2.3	Splineinterpolation	117
6.3	Ausgleichsrechnung	122
6.3.1	Problemstellung	122
6.3.2	Lineare Ausgleichsprobleme	124
6.3.3	Nichtlineare Ausgleichsprobleme	129
6.3.4	Das Gauss-Newton-Verfahren	131
7	Numerische Integration	137
7.1	Zur historischen Entwicklung ⁸	137
7.2	Numerische Integration	138
7.2.1	Problemstellung	138
7.2.2	Rechteck- und Trapezregel	138
7.2.3	Die Simpson-Regel	140
7.2.4	Der Fehler der summierten Quadraturformeln	142
7.2.5	Gauss-Formeln	142
7.2.6	Romberg-Extrapolation	143

⁶Kapitel hauptsächlich übernommen aus [13]

⁷Gemäß Wikipedia:http://de.wikipedia.org/wiki/Methode_der_kleinsten_Quadrate Gemäß Erik Merijering (2002), “A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing.”

⁸Hauptsächlich gemäß Wikipedia und http://www-history.mcs.st-andrews.ac.uk/HistTopics/The_rise_of_calculus.html

8 Einführung in gewöhnliche Differentialgleichungen	148
8.1 Zur historischen Entwicklung	148
8.2 Problemstellung	149
8.3 Beispiele aus den Naturwissenschaften	151
8.3.1 Der Freie Fall (aus Papula)	151
8.3.2 Harmonische Schwingung eines Federpendels (aus Papula)	153
8.4 Richtungsfelder für Differentialgleichungen 1. Ordnung	154
8.5 Das Euler-Verfahren	156
8.5.1 Das klassische Euler-Verfahren	157
8.5.2 Das Mittelpunkt-Verfahren	158
8.5.3 Das modifizierte Euler-Verfahren	160
8.6 Die Fehlerordnung eines Verfahrens	162
8.7 Runge-Kutta Verfahren	164
8.7.1 Das klassische vierstufige Runge-Kutta Verfahren	164
8.7.2 Das allgemeine s -stufige Runge-Kutta Verfahren	166
8.8 Erweiterung auf Systeme von Differentialgleichungen	167
8.8.1 Zurückführen einer DGL k -ter Ordnung auf k DGL 1. Ordnung	167
8.8.2 Lösen eines Systems von k DGL 1. Ordnung	169
8.9 Stabilität	171
8.10 Weitere Punkte	172
8.10.1 Einschritt- vs. Mehrschrittverfahren	172
8.10.2 Implizite vs. explizite Verfahren	173
8.10.3 Steife DGL	173
8.10.4 Schrittweitensteuerung	173
8.10.5 Python-Funktionen zur Lösung von Anfangswertproblemen	173

Kapitel 1

Einführung in die Numerische Mathematik

Dieses Kapitel gibt eine Einführung darin, was Numerische Mathematik ist, wo die Verbindungen zur Informatik (und umgekehrt) liegen, wie sie sich im geschichtlichen Kontext entwickelte und was einige typische Fragestellungen sind, die wir im Verlauf der Vorlesung behandeln werden.

Lernziele:

- Sie kennen die Definition der Numerischen Mathematik sowie die wichtigsten Anknüpfungspunkte zur Informatik.
- Sie kennen den geschichtlichen Hintergrund der Entwicklung der Numerischen Mathematik.
- Sie kennen einige der typischen Fragestellungen der Numerischen Mathematik.

1.1 Was ist Numerische Mathematik

Die Numerische Mathematik, kurz Numerik genannt, beschäftigt sich als Teilgebiet der Mathematik mit der Konstruktion und Analyse von Algorithmen für kontinuierliche mathematische Probleme. Hauptanwendung ist dabei die Berechnung von Lösungen mit Hilfe von Computern¹. Im Gegensatz zur analytischen Rechnung will man bei der Numerik keine geschlossenen Formeln oder algebraische Ausdrücke erhalten, sondern, wie der Name sagt, numerische Resultate. Unter einem Algorithmus verstehen wir dabei “eine endliche Menge genau beschriebener Anweisungen (arithmetische und logische Operationen und Ausführungshinweise), die in einer bestimmten Reihenfolge auszuführen sind, um mit Hilfe der eingegebenen Daten die gesuchten Ausgabedaten zu ermitteln” [6].

Interesse an solchen Algorithmen besteht meist aus einem der folgenden Gründe:

1. Es gibt zu dem Problem keine explizite Lösungsdarstellung (so zum Beispiel bei den Navier-Stokes-Gleichungen oder dem Dreikörperproblem) oder
2. die Lösungsdarstellung existiert, ist jedoch nicht geeignet, um die Lösung schnell zu berechnen oder liegt in einer Form vor, in der Rechenfehler sich stark bemerkbar machen (zum Beispiel bei vielen Potenzreihen).

Unterschieden werden zwei Typen von Verfahren: Einmal direkte, die nach endlicher Zeit bei unendlicher Rechengenauigkeit die exakte Lösung eines Problems liefern, und auf der anderen Seite Näherungsverfahren, die – wie der Name sagt – nur Approximationen liefern. Ein direktes Verfahren ist beispielsweise das gaußsche Eliminationsverfahren, welches die exakte Lösung eines linearen Gleichungssystems ermöglicht. Näherungsverfahren sind unter anderem Quadraturformeln, die den Wert eines Integrals näherungsweise berechnen oder auch das Newton-Verfahren, das iterativ bessere Approximationen einer Nullstelle einer Funktion liefert. Da in Anwendungen die Lösungen nur auf endliche Genauigkeit benötigt werden, kann ein iteratives Verfahren auch bei der Existenz eines direkten Verfahrens sinnvoller sein, wenn es in kürzerer Zeit diese Genauigkeit liefert. Unterschiedliche Verfahren werden nach Laufzeit, Stabilität und Robustheit verglichen.

¹Definition gemäß Wikipedia

Die Verbindung der Numerik mit der Informatik ist offensichtlich, ist doch die effiziente Berechnung numerischer Algorithmen ohne Computer meist nicht möglich. Umgekehrt kommt man bei der täglichen Arbeit mit dem Computer zwingend auf direkte oder indirekte Art mit Grundverfahren der Numerik in Berührung. Dies gilt insbesondere in den Bereichen (gemäss [7]):

- Zahldarstellung und -arithmetik
- Implementierung mathematischer Funktionen
- Computergraphik (Darstellung von Objekten)
- Bildverarbeitung (Kompression, Analyse, Bearbeitung)
- Neuronale Netze (Lernverfahren)
- Information Retrieval (Vektorraummodell)
- Chip Design (Algebraische Differentialgleichungen)
- stochastische Automaten und Markov-Ketten (Prozessverwaltung, Warteschlangen)

Die Numerische Mathematik erscheint manchmal als eine nicht sehr übersichtliche Sammlung von Rezepten für eine Vielzahl von numerischen Problemen. Dies ist irreführend. Definiert man die Informatik wie im englischen Sprachgebrauch als die "Wissenschaft vom Computer" (Computer Science), so ist die Numerische Mathematik in natürlicher Weise darin enthalten (vgl. [7]). Dies äusserst sich z.B. im wichtigen Bereich des Wissenschaftlichen Rechnens (Scientific Computing), in dem Numeriker und Informatiker zusammen mit Wissenschaftlern daran arbeiten, komplexe Anwendungsprobleme verschiedener Wissenschaftsgebiete fachübergreifend zu lösen (typisches Beispiel ist die Meteorologie mit Wetter- und Klimamodellen sowie den entsprechenden Vorhersagen). Die Numeriker beschäftigen sich dabei mit der Entwicklung effizienter Algorithmen und Methoden, die das mathematische Problem möglichst gut diskret approximieren. Die Informatiker sind zuständig für die effiziente Implementierung (bzgl. Rechenzeit, Speicherverwaltung, Cache, Parallelisierung etc., vgl. [7]). Allerdings verschwimmen die Grenzen zwischen den Disziplinen in diesem Bereich, d.h. Numeriker und Informatiker benötigen ein umfangreiches Wissen der jeweils anderen Disziplin.

1.2 Historische Entwicklung²

Die Anfänge der (numerischen) Mathematik reichen zurück bis zu den frühen Hochkulturen. Das erste logisch aufgebaute Zahlensystem, ein Additionssystem, dürfte auf die Ägypter zurückgehen (ca. 3000 v. Chr.). Die Babylonier kannten zu Beginn des 2. Jahrtausends vor Christus bereits ein Zahlensystem mit der Basis 60. Eine Näherung der Quadratwurzel $\sqrt{2}$ aus dieser Zeit ist in Abb. 1.1³ dargestellt,

Wir wollen auf die Entwicklung der Zahlensysteme an dieser Stelle nicht weiter eingehen, eine detaillierte Beschreibung findet sich in Kap. 2.1. Sowohl die Ägypter als auch die Babylonier kannten die vier Grundrechenarten sowie Näherungen für π . In der Antike (ca. 1200 v.Chr. bis ca. 600 n.Chr.) betrieben die Griechen Mathematik als Wissenschaft im Rahmen der Philosophie und prägten die strenge logische Beweisführung. Ihr Augenmerk lag dabei, wie schon bei den Ägyptern und Babylonien, vorwiegend auf der Geometrie (als eines der Hauptwerke galten die Schriften des Griechen Euklid von Alexandria, 3. Jhr. v.Chr.; noch heute sprechen wir von euklidischer Geometrie). Arabische Gelehrte bauten auf den griechischen und indischen Erkenntnissen auf (das uns bekannte moderne Dezimalsystem mit der Null wurde von indischen Mathematikern im Zeitraum 3. Jhr. v.Chr. bis 5. Jhr. n.Chr. entwickelt) und verbreiteten dieses Wissen über Spanien und Italien nach Europa.

Ab dem 13. Jhr. n.Chr. konnte sich das schriftliche Rechnen mit den indisch-arabischen Ziffern langsam durchsetzen und die römischen Ziffern verdrängen. Aber selbst der deutsche Rechenmeister Adam Ries (1492 - 1559) beschrieb in seinen Rechenbüchern neben dem schriftlichen Rechnen mit den arabischen Ziffern hauptsächlich noch das Rechnen mit Abakus, Linien und Steinen, das nur wenige, meist Verwaltungsbeamte, Kaufleute und Gelehrte beherrschten.

Michael Stiefel (deutscher Theologe und Mathematiker, 1487-1567) führte bereits kurze Zeit später die negativen Zahlen ein und prägte den Begriff Exponent. Dem Schweizer Jost Bürgi (Uhrmacher, Instrumentenbauer und Hofastronom, 1552-1632) wird die Einführung der Logarithmentafeln zugerechnet (ab 1588), die er 1605 dem Astronomen

²Hauptsächlich gemäss Kap. 1 aus [7], erweitert mit Zusatzinformationen aus Wikipedia.

³"Ybc7289-bw". Licensed under Creative Commons Attribution 2.5 via Wikimedia Commons - <http://commons.wikimedia.org/wiki/File:Ybc7289-bw.jpg#mediaviewer/File:Ybc7289-bw.jpg>

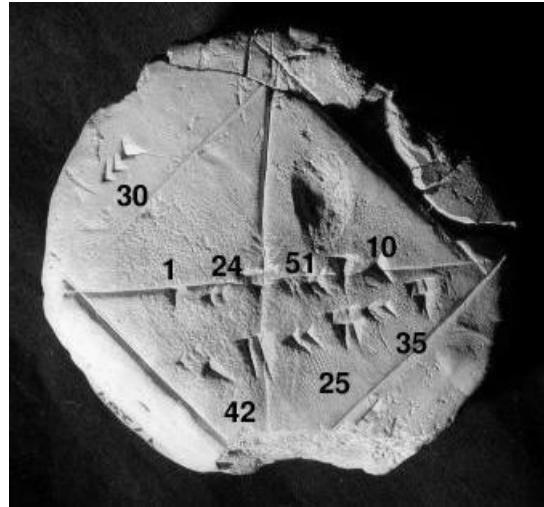


Abbildung 1.1: Babylonische Tontafel YBC 7289 von ca. 1800-1600 v.Chr. Die Näherung von $\sqrt{2}$ ist in der Diagonale eines Quadrates dargestellt mit den Symbolen für $1 + 24/60 + 51/60^2 + 10/60^3 = 1.41421296\dots$

Kepler (berühmt für seine drei Keplerschen Gesetze, 1571-1630) zugänglich machte. Allerdings publizierte er sie erst 1620, sechs Jahre später als Lord John Napier (schottischer Mathematiker, 1550-1617), der seither als Entdecker des natürlichen Logarithmus gilt. Die von ihm entwickelten Rechenstäbchen mit logarithmischer Skala ("Napier-Bones") erlaubten es, die Multiplikation von Zahlen auf einfache Weise auf die Addition zurückzuführen.⁴ Daraus entwickelten die englischen Theologen und Mathematiker Edmund Gunter (1581-1626) und William Oughtred (1574-1660) um 1622 den ersten funktionsfähigen Rechenschieber.

Der Tübinger Universalgelehrte Wilhelm Schickard (1592-1635) entwickelte 1623 die erste Zahnrad-betriebene Rechenmaschine, die die Addition und Subtraktion von bis zu sechsstelligen Zahlen beherrschte und zur Unterstützung von Multiplikation und Division Napier-Rechenstäbchen verwendete. Der Franzose Blaise Pascal (1623-1662) stellte 1642 das erste mechanische Rechenwerk für Addition und Subtraktion mit durchlaufendem Zehnerübertrag her. Im Jahre 1671 entwickelte der deutsche Universalgelehrte Gottfried Willhelm Leibniz (1646-1716) seine Rechenmaschine REPLICA mit zwölfstelligem Anzeigewerk, die alle vier Grundrechenarten beherrschte und beschrieb kurz darauf das binäre Zahlensystem. Zusammen mit Sir Isaac Newton (1643-1727) gilt er als Begründer der Infinitesimalrechnung. Einige der Algorithmen, die wir im Verlauf der Vorlesung kennen lernen werden, gehen auf Newton zurück (z.B. die Nullstellenbestimmung nichtlinearer Funktionen, Polynominterpolation u.a.), die er unter anderem für seine bahnbrechenden Arbeiten in der Mechanik benötigte. Im Jahr 1687 erschien sein Hauptwerk *Philosophia Naturalis Principia Mathematica*. Noch heute sprechen wir in der Physik in der klassischen Mechanik von den drei Newtonschen Gesetzen.

Auf den Werken von Leibniz und Newton setzte der bedeutende Schweizer Mathematiker und Physiker Leonhard Euler (1707 in Basel geboren, 1783 in Petersburg gestorben) auf⁵, der wegen seiner Beiträge in der Analysis und Zahlentheorie und weiteren Teilgebieten der Mathematik Berühmtheit erlangte. Nach ihm sind unter anderem die eulersche Zahl e oder die eulersche Konstante γ benannt (die ausreichend wichtig ist, um ihren Wert bis 2009 auf fast 30 Milliarden dezimale Nachkommastellen zu berechnen). Ein Grossteil der mathematischen Symbolik geht auf Euler zurück (z.B. e , π , die imaginäre Zahl i , \sum , $f(x)$). Seine Arbeit über das Königsberger Brückenproblem gilt als eine der ersten Arbeiten auf dem Gebiet der Graphentheorie, heute in der Informatik von grosser Bedeutung in der Komplexitätstheorie von Algorithmen.

Im 18. Jahrhundert wurden die mechanischen Rechenmaschinen weiterentwickelt (z.B. durch Johannes Polenius, Antonius Braun, Philipp Matthäus Hahn). Mit der serienmässigen Fertigung ab 1821 durch Charles Xavier Thomas (1785-1870) in Paris entstanden in der Folge Hunderte von verschiedenartigen mechanischen Rechenmaschinen.

Der englische Mathematiker Charles Babbage (1791-1871) entwarf 1822 seine Differenzenmaschine, die auch Logarithmen berechnen konnte. Die von ihm konzipierte, aber nie gebaute Analytical Engine (ab 1833) gilt als Vorläufer des modernen Computers. Er sah bereits getrennte Baugruppen für Speicher und Rechenwerk und sogar ein Druckwerk vor. Seine enge Mitarbeiterin Ada Lovelace (1815-1852), ebenfalls Mathematikerin, beschrieb die Programmierung der Maschine in der Theorie und gilt als erste Programmiererin. Nach ihr wurde die Programmier-

⁴Gemäss $x \cdot y = \exp(\ln(x) + \ln(y))$ bzw. $\ln(x \cdot y) = \ln(x) + \ln(y)$.

⁵Von 1976-1995 auf der Schweizer 10 Franken Note abgebildet.

sprache Ada und die Lovelace-Medal benannt (letztere wird von der British Computer Society seit 1998 vergeben). Die Maschine sollte über Lochkarten gesteuert werden, wie sie von dem französischen Erfinder Jean-Marie Jacquard (1752-1834) zur Steuerung von Webstühlen eingeführt worden waren und der damit entscheidend zur industriellen Revolution beitrug.

Der Zeitgenosse Johann Carl Friedrich Gauss (1777-1855), deutscher Mathematiker, Physiker, Astronom und Geodät, entwickelte eine Vielzahl von Algorithmen, die auch heute noch von Bedeutung sind und seinen Namen tragen. Einige davon werden wir im Rahmen dieser Vorlesung behandeln, so das gaußsche Eliminationsverfahren zur exakten Lösung linearer Gleichungssysteme und das Gauss-Seidel Verfahren zur iterativen Lösung. Daneben gibt es noch eine Vielzahl weiterer Algorithmen, die Sie zumindest teilweise in anderen Mathematik-Vorlesungen kennen lernen werden und die auf ihn zurückgehen, z.B. die gaußsche Normalverteilung in der Statistik und das entsprechende gaußsche Fehlerintegral.

In der Zeit von 1848 bis 1850 entwickelte der englische Mathematiker George Boole (1815-1864) die Boolesche Algebra, die Grundlage der heutigen binären Rechenschaltungen. Die technische Entwicklung wurde 1890 vorangetrieben durch die Lochkartenmaschine des amerikanischen Ingenieurs Herman Hollerith (1860-1929), die zur Volkszählung in den USA entwickelt wurde. In den dazugehörigen Lesegeräten steckten kleine Metallstäbe, die an den Stellen, wo die Karte gelocht war, einen Kontakt zuließen, so dass elektrischer Strom fliessen konnte. Die Auswertung der Daten dauerte mit dieser Technik statt mehrerer Jahre nur einige Wochen.

Der Höhepunkt der mechanischen Rechenmaschinen war wohl mit dem ersten Taschenrechner, der Curta, erreicht. Im Konzentrationslager Buchenwald vollendete der jüdische Häftling Curt Herzstark (1902-1988), ein österreichischer Erfinder und Sohn eines Rechenmaschinenherstellers, die Pläne für seinen Lilliput Rechner, der von der SS als Siegesgeschenk an den Führer vorgesehen war. Wieder in Freiheit wurde Herzstark in Liechtenstein technischer Direktor der Cortina AG zur Herstellung und Vertrieb der Curta, einer verbesserten Form des Lilliput.

Generell wurde während des zweiten Weltkrieges auf beiden Seiten erhebliche Ressourcen für die Entwicklung besserer Rechenmaschinen und numerischer Algorithmen eingesetzt. Die 1940er Jahre markieren deshalb den Beginn der 'modernen' Numerischen Mathematik.

Einen Meilenstein hierbei setzte der englische Mathematiker und Kryptoanalytiker Alain Turing (1912-1954) mit seinem Beitrag zur Entschlüsselung der mit der Enigma verschlüsselten deutschen Funksprüche. Unter anderem wegen seiner 1936 erschienenen Arbeit *On Computable Numbers with an Application to the "Entscheidungsproblem"* und dem damit verknüpften Begriff der Turingmaschine gilt er als einer der einflussreichsten Theoretiker der frühen Computerentwicklung und Informatik. Auf der deutschen Seite entwickelte 1941 der Bauingenieur und Erfinder Konrad Zuse (1910-1995) die Z3, den ersten Relaisrechner⁶ und damit ersten funktionsfähigen Digitalrechner weltweit. Als Kompromiss zwischen der Festkomma-Zahldarstellung, in der problemlos addiert werden kann, und einer logarithmischen Darstellung, die das Multiplizieren erlaubt, wurde die noch heute übliche Gleitpunktdarstellung verwendet. Der Rechner wurde zur Berechnung von komplexen Matrizen eingesetzt, die in der Aerodynamik (dem sogen. 'Flügelflattern', welches zum Absturz von Flugzeugen führte) eine Rolle spielten. Er wurde bei einem Bombenangriff der Alliierten zerstört und im März 1945, kurz vor der Kapitulation Deutschlands, durch die Zuse Z4 ersetzt⁷.

In den USA wurde 1944 der vom amerikanischen Computerpionier Howard Aiken (1900-1973) entwickelte, vollständig aus elektromechanischen Bauteilen zusammengesetzte Grossrechner Mark I in Betrieb genommen (er war ca. 2.5 Meter hoch und ca. 17 Meter lang, sein Gewicht betrug 5 Tonnen). Das Projekt wurde hauptsächlich von IBM finanziert. Der Rechner diente anfänglich dem in die USA geflüchteten, österreich-ungarischen Mathematiker John von Neumann (1903-1957) im Rahmen des Manhattan-Projekts für Berechnungen des Implosionsmechanismus der Plutoniumbombe. Von Neumann entwickelte dazu erste numerische Verfahren zur Lösung von partiellen Differentialgleichungen. 1946 stellte er (in loser Analogie zum menschlichen Hirn) die Fundamentalprinzipien eines frei programmierbaren Rechners auf (Prozessor, Speicher, Programm und Daten im Prozessor). Praktisch alle modernen Rechner beruhen auf von Neumanns Rechnerarchitektur.

Ebenfalls 1946 wurde ENIAC (Electronic Numerical Integrator and Computer), der erste rein elektronische Rechner, vorgestellt. Entwickelt worden war er von den amerikanischen Computeringenieuren John Eckert und John Mauchly, welche 1951 mit dem UNIVAC I (Universal Automatic Computer) den ersten in den USA hergestellten kommerziellen Computer auf den Markt brachten. Dieser benötigte eine Leistung von bis zu 125 kW und konnte knapp 2000 Rechenoperationen pro Sekunde ausführen. 1958 entstand der erste integrierte Schaltkreis (ein Flipflop). Die Telefunken TR 4 war 1962 der erste Computer auf Basis von Transistor Bausteinen und 1967 eroberte der erste Taschenrechner den Markt. 1976 wurde der erste Home-Computer Apple I geboren, und ab 1981 begann der

⁶Ein Relais ist ein mit Strom betriebener, elektromagnetischer Schalter mit normalerweise zwei Schaltstellungen

⁷Diese überstand die Kriegswirren und wurde von 1950 bis 1955 an der ETH als zentraler Rechner eingesetzt. Sie wurde durch die Eigenentwicklung ERMETH (Elektronische Rechenmaschine der ETH) ersetzt, welche 1956-1963 in Betrieb war.

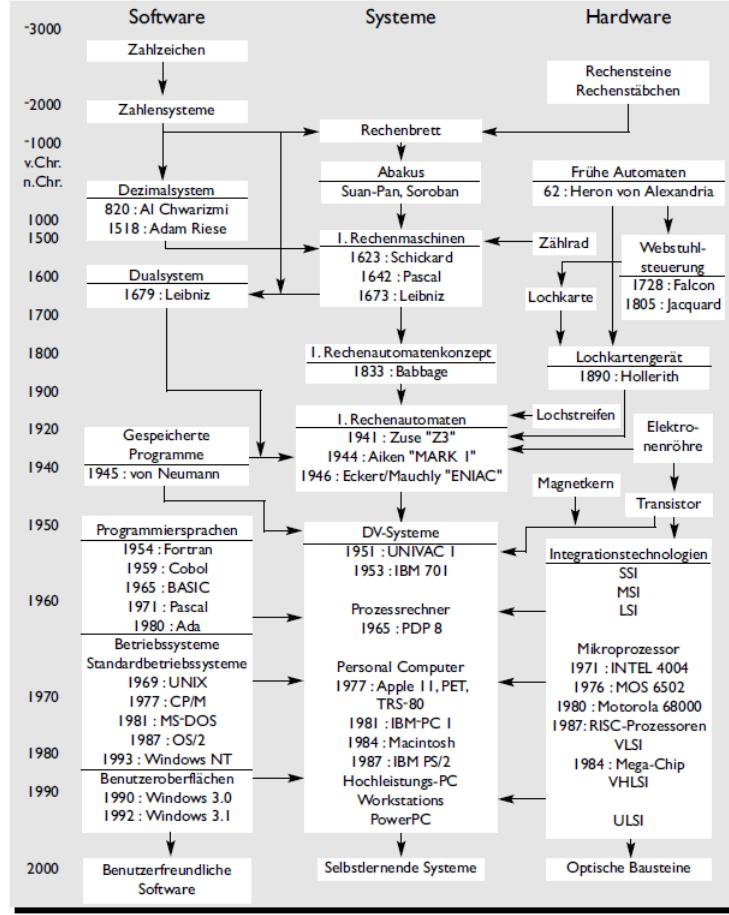


Abbildung 1.2: Zur geschichtlichen Entwicklung des Computers (aus Computergeschichte - Faszination über Jahrtausende, R. Weiss)

Siegeszug des PC.

Die Zukunft der Numerischen Mathematik wird voraussichtlich weiter stark geprägt werden von der weiteren Entwicklung des Computers. Das Moorsche Gesetz, welches die Verdopplung der Rechenleistung alle 18-24 Monate voraussagt, wird früher oder später durch physikalische Gründe ausser Kraft gesetzt werden. Ein alternativer Ansatz stellt die Idee eines Quantencomputers dar, welcher auf Basis der quantenmechanischen Zustände der Atome rechnet und dadurch unvorstellbar viele Operationen parallel durchführen kann.

1.3 Typische Fragestellungen

Im Folgenden betrachten wir einige typische Fragestellungen, die in den Vorlesungen Höhere Mathematik 1 und 2 behandelt werden:

1. Rechnerarithmetik (Kap. 2)

Wie wirkt sich die Beschränkung der Anzahl Bits für Zahlenformate auf Rechenergebnisse und die Fehlerfortpflanzung aus? Betrachten wir z.B. die IEEE-Formate »Single Precision« oder »Double Precision« in Abb. 1.3. Durch die Abbildung von reellen Zahlen auf maschinendarstellbare Zahlen (mit einer begrenzten Anzahl Bits) können erhebliche Rechenun genauigkeiten entstehen, unter Umständen mit katastrophalen Folgen wie bei der Explosion der unbemannten Ariane 5 Rakete bei ihrem ersten Flug am 4. Juni 1996.

single: (32 bit)

V EEEEEEEE	MMMMMM	MMMMMM	MMMMMM
0 1	8 9		31

double: (64 bit)

V EEEEEEEEEE	MMMMMM	MMMMMM	MMMMMM	MMMMMM	MMMMMM	MMMMMM	MM
0 1	11 12						63



Abbildung 1.3: IEEE-Zahlenformate (oben) sowie die Explosion der umbenannten Ariane 5 Rakete am 4. Juni 1996 (unten, von https://youtu.be/gp_D8r-2hwk).

2. Numerische Lösung von Nullstellenproblemen (Kap. 3)

Sei $f : \mathbb{R} \rightarrow \mathbb{R}$. Gesucht ist $x \in \mathbb{R}$ mit $f(x) = 0$.

Für die quadratische Gleichung $f(x) = x^2 + px + q$ sind die Nullstellen bekannt:

$$x_{1,2} = -p/2 \pm \sqrt{p^2/4 - q}.$$

Aber wie bestimmt man Nullstellen von komplizierteren Funktionen wie $f(x) = x^n + a_{n-1}x^{n-1} + \dots + a_0$ oder $f(x) = \exp(x) - \sin(x)$?

Beispiel: Iteratives Tangenten Verfahren nach Newton.

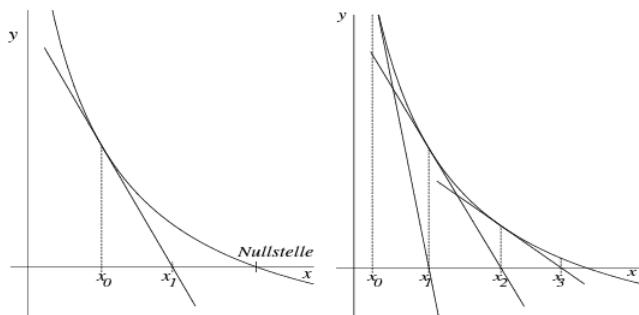


Abbildung 1.4: Newtonverfahren (aus [1])

Die Funktion $f(x)$ kann in der Umgebung von x_0 durch ihre Tangente angenähert werden: $f(x) \approx f(x_0) + f'(x_0)(x - x_0)$. Der Schnittpunkt x_1 der Tangenten mit der x-Achse ist eine erste Näherung für die Nullstelle, dort gilt $0 = f(x_0) + f'(x_0)(x_1 - x_0)$ und daraus ergibt sich $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$. Durch wiederholen des Verfahrens (siehe Abbildung 1.4) erhält man nach n Schritten die Iterationsformel

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \quad (n = 1, 2, 3, \dots).$$

3. Numerische Lösung von linearen Gleichungssystemen (Kap. 4).

Beispiel: $A(n, n)$ sei eine Matrix reeller Zahlen und $\vec{b} \in \mathbb{R}^n$ ein Vektor. Gesucht ist der Vektor $\vec{x} \in \mathbb{R}^n$, so dass $A\vec{x} = \vec{b}$. Dieses Problem kann gemäss der linearen Algebra gelöst werden mit $\vec{x} = A^{-1} \cdot \vec{b}$, doch ist diese Lösung so nicht effizient berechenbar. Wir werden für die Berechnung von \vec{x} effiziente numerische Verfahren kennen lernen.

4. Numerische Lösung von nicht linearen Gleichungssystemen (Kap. 5)

Dies ist im Prinzip eine Erweiterung des Nullstellenproblems in einer Dimension auf weitere Dimensionen. Zu lösen ist zum Beispiel das Problem:

$$f(x_1, x_2) := \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 + 4x_2 \\ 8x_2^3 + 4x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Am Term x_2^3 sieht man, dass das System nicht linear ist. Das oben vorgestellte Newton Verfahren lässt sich auf diese Fragestellung erweitern und für die Lösung solcher Systeme verwenden.

5. Interpolation (Kap. 6)

Beim typischen Interpolationsproblem sind n diskrete Wertepaare (x_i, y_i) gegeben und gesucht ist eine stetige Funktion f mit der Eigenschaft $f(x_i) = y_i$ für alle x_i . Beispiel: Es soll eine interpolierende Funktion für die beiden Werte $(0,1)$ und $(1,2)$ gefunden werden. Eine mögliche Lösung ist $f(x) = x+1$, aber auch $f(x) = \sqrt{x}+1$ oder $f(x) = \sin(\pi x) + x + 1$ kommen in Frage (vgl. Abb. 1.5). Tatsächlich lösen unendlich viele Funktionen dieses Problem, was zeigt, dass Interpolationsprobleme nicht per se eindeutig lösbar sind.

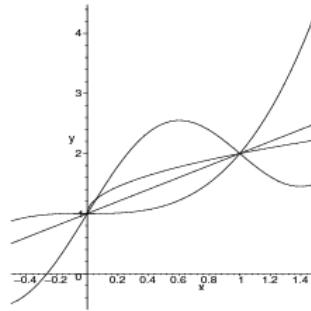


Abbildung 1.5: Interpolation durch die Punkte $(0,1)$ und $(1,2)$ (aus [1]).

6. Regression (auch 'Ausgleichsrechnung', Kap. 6)

Häufig sind n Wertepaare (x_i, y_i) das Resultat von Messungen und deshalb mit gewissen Unsicherheiten behaftet, welche sich z.B. als Streuung manifestieren können (sogenannte 'Scatter Plots'). Dann empfiehlt es sich, nicht eine Funktion zu suchen, die exakt durch die Wertepaare geht (wie bei der Interpolation in 5.), sondern möglichst 'nah' an alle beobachteten Werte rankommt. Ein theoretisches Beispiel ist in Abb. 1.6 gezeigt. Offenbar gibt es dort einen Trend, dass die gemessene Grösse y (z.B. die Fläche einer bestimmten Moosart) mit der Grösse x (z.B. ein Mass für den pH Wert des Bodens) etwa linear abnimmt. Dieser Trend lässt sich als Gerade $f(x)$ mittels der Methode der kleinsten Quadrate berechnen, so dass der Fehler $\sum(y_i - f(x_i))^2$ minimal wird (lineare Regression).

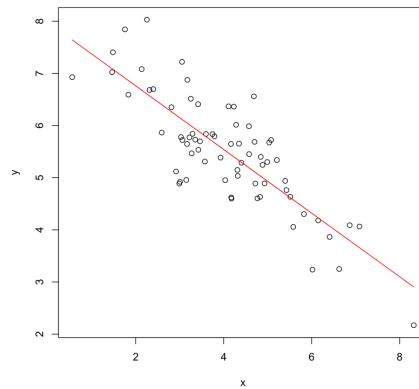


Abbildung 1.6: Sogenannter 'Scatter Plot' und die mittels linearer Regression bestimmte Näherungsgerade.

7. Numerische Integration (Kap. 7)

Gegeben ist $f : [a, b] \rightarrow \mathbb{R}$. Gesucht ist ein Näherungswert des bestimmten Integrals

$$I = \int_a^b f(x)dx$$

Beispiel: Mittelpunkts- oder Trapezregel wie in Abb. 1.7.

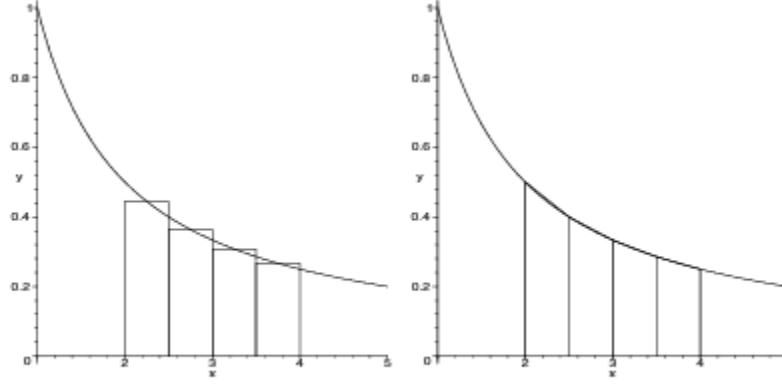


Abbildung 1.7: Beispiel zur Mittelpunkt- und Trapezregel (aus [1]).

8. Differentialgleichungen (Kap. 8)

Gegeben ist eine Funktion $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, ein Intervall $[a, b]$ und ein Anfangswert y_0 . Gesucht ist eine Funktion $y : [a, b] \rightarrow \mathbb{R}$ mit

$$y'(t) = f(t, y(t))$$

für alle $t \in [a, b]$ und $y(a) = y_0$. Das Anfangswertproblem besteht also darin, eine Lösung y der gewöhnlichen Differenzialgleichung zu finden, die an der Stelle $t = a$ den vorgegebenen Wert y_0 annimmt. Die Lösungen lassen sich anhand sogenannter Richtungsfelder wie in Abb. 1.8 veranschaulichen.

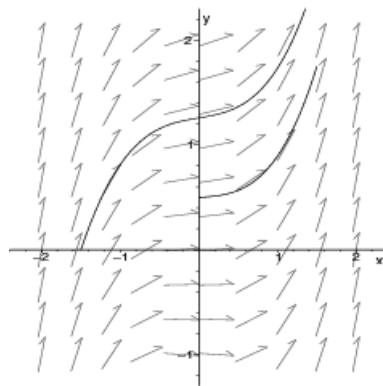


Abbildung 1.8: Richtungsfeld einer gewöhnlichen Differentialgleichung (aus [1]).

Kapitel 2

Rechnerarithmetik

Dieses Kapitel beschäftigt sich damit, wie Zahlen dargestellt werden, was für Fehler dabei entstehen können und wie sich diese bei Operationen fortpflanzen.

Lernziele:

- Sie verstehen die Definition der maschinendarstellbaren Zahlen.
- Sie können die Fehler, die beim Abbilden von reellen Zahlen auf Maschinenzahlen entstehen, sowie die Maschinengenauigkeit berechnen.
- Sie können die Fortpflanzung von Fehlern bei Funktionsauswertungen abschätzen und die Konditionszahl berechnen.

2.1 Zur Geschichte der Zahlendarstellung¹

In den frühen Hochkulturen entwickelten sich unterschiedliche Konzepte zur Darstellung von Zahlen, die nach Art der Zusammenstellung und der Anordnung der Ziffern in Additionssysteme und Positionssysteme (auch Stellenwertsysteme genannt) einteilbar sind. Additionssysteme ordnen jeder Ziffer eine bestimmte Zahl zu. Im Gegensatz dazu ordnen Positions- oder Stellenwertssysteme jeder Ziffer aufgrund der relativen Position zu anderen Ziffern eine Zahl zu. So haben die beiden Ziffern 2 und 5 im Dezimalsystem je nach Zusammenstellung entweder den Wert 25 oder 52. Alle Zahlensysteme bauen dabei auf einer sogenannten ganzzahligen Grundzahl² $B > 1$, auch Basis genannt, auf.

Das ägyptische Additionssystem dürfte mit ca. 5000 Jahren wohl das älteste logisch aufgebaute Zahlensystem sein. Es benutzt die Basis $B = 10$. Für die ersten sieben Stufenzahlen 10^i mit $0 \leq i \leq 6$ werden spezielle Hieroglyphen als Zahlzeichen verwendet, wie in der Abb. 2.1 zu sehen. Die Ziffer 0 existierte zu diesem Zeitpunkt noch nicht, was bei dieser Darstellungsart nicht als Mangel empfunden wird.

Zu Beginn des 2. Jahrtausends v. Chr. verwendeten die babylonischen Mathematiker eine Zahlenschrift, die nur einen Nagel für eine Eins und einen offenen Winkel für die 10 benutzte. Ein Nagel konnte dabei - je nach seinem Abstand zu den anderen Winkeln oder Nägeln - eine 1 oder eine 60 bedeuten. Die Zahl $3661 = 1 \cdot 60^2 + 1 \cdot 60^1 + 1 \cdot 60^0$ wurde durch drei Nägel mit Abstand angezeigt, während drei Nägel ohne Abstand einfach für 3 standen. Das Problem der eindeutigen Darstellung der Zahl $3601 = 1 \cdot 60^2 + 0 \cdot 60^1 + 1 \cdot 60^0$ wurde fast zwei Jahrtausende später gelöst, indem man als Platzhalter für eine fehlende Stelle zwei schräg hochgestellte Nägel einführte, also 3601 durch zwei Nägel, getrennt durch die beiden hochgestellten Nägel, markierte. Siehe dazu auch Abb. 2.2.

Sollen die Ziffern unabhängig von der Position verwendet werden, kommen wir also um die Darstellung der Ziffer Null nicht herum. Wir sind dann z.B. in der Lage, die beiden Zahlen $701 = 7 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0$ und $71 = 7 \cdot 10^1 + 1 \cdot 10^0$ zu unterscheiden. Die Ziffer 0 deutet das Auslassen einer "Stufenzahl" B^i an und ermöglicht

¹Übernommen in gekürzter und leicht abgeänderter Form von Kap. 1 und Kap. 5 aus [7]

²Vor allem wurden die Zahlen 2, 5, 10, 12, 20 und 60 benutzt. Die wohl wichtigsten Grundzahlen sind 2 und 10. Von besonderem Interesse für die Babylonier war die Zahl 60, da sie zugleich die Zahl 30, also ungefähr die Anzahl Tage in einem Monat, als auch die Zahl 12, die Anzahl Monate in einem Jahr, als Teiler besitzt.

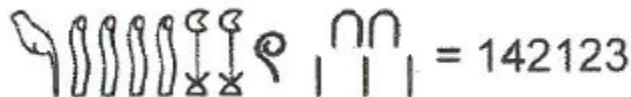


Abbildung 2.1: Symbole zum Darstellen von Zahlen bei den antiken Ägyptern: Ein Strich war ein Einer, ein umgekehrtes U ein Zehner, die Hunderter wurden durch eine Spirale, die Tausender durch die Lotosblüte mit Stil und die Zehntausender durch einen oben leicht angewinkelten Finger dargestellt. Dem Hunderttausender entsprach eine Kaulquappe mit hängendem Schwanz. Ergänzend ohne Bild hier: Die Millionen wird durch einen Genius, der die Arme zum Himmel erhebt, repräsentiert (aus [7]).



Abbildung 2.2: Babylonische Form der Zahl $46821 = 13 \cdot 60^2 + 0 \cdot 60^1 + 21 \cdot 60^0$ als $13 | 0 | 21$ (aus [7]).

eine übersichtlichere Darstellung in der modernen Nomenklatur

$$z = \sum_{i=0}^n z_i B^i.$$

Die Symbole für die Ziffern $z_i \in \{0, 1, \dots, b - 1\}$ können wir dann als Koeffizienten der Stufenzahlen B^i interpretieren. Die Einführung der für uns heute nicht mehr wegzudenkenden Null als eigenständige Ziffer verdanken wir indischen Mathematikern.

Im 3. Jahrhundert v.Chr. wurde in Nordindien ein Zehnersystem entwickelt, das für die Ziffern 1 bis 9 graphische Zeichen benutzte, und dabei für diese Ziffern in verschiedenen Zehnerpotenzen auch verschiedene Zeichen zur Verfügung hatte, also z.B. für 9, 90, 900, usw. jeweils ein eigenes Zeichen. Im 5. Jahrhundert n.Chr. fanden indische Mathematiker heraus, dass ihr Zahlensystem sich stark vereinfachen liess, wenn man auf diese Unterscheidung der Potenzen verzichtete und statt dessen eine eigene neue Ziffer, die Null, die Auslassung einer Zehnerpotenz anzeigen. Das indische Wissen wurde durch Araber wie den Universalgelehrten Mohammed Ibn Musa al-Charismi³ (etwa 780 - 850) nach Europa weitervermittelt (siehe Abb. 2.3).

³Gemäss Wikipedia auch al-Chwarzimi. Von seinem Namen leitet sich der Begriff Algorithmus ab. Sein Beiname al-Chwarzimi bedeutet "der Choresmier" und bezieht sich auf seine Herkunft aus diesem iranischen Volk. Sein arabisches Lehrbuch *Über das Rechnen mit indischen Ziffern* (um 825) beginnt in der mittelalterlichen lateinischen Übersetzung mit den Worten *Dixit Algorismi* (Algorismi hat gesagt).



Abbildung 2.3: Arabische und indische Symbole zum Darstellen von Zahlen: In der ersten Zeile sehen wir die indischen Ziffern des 2. Jahrhunderts n.Chr. Diese bildhaften Ziffern wurden erst von den Arabern übernommen (zweite Zeile) und später von den Europäern (dritte bis sechste Zeile: 12., 14., 15. und 16. Jhr.) immer abstrakter dargestellt (aus [7]).

In Europa galt die Ziffer Null lange als Teufelswerk und findet sich erstmalig in einer Handschrift von 976. Bis ins Mittelalter wurden in Europa Zahlen in lateinischen Grossbuchstaben geschrieben. Im römischen Zahlensystem standen I, V, X, L, C, D und M für 1, 5, 10, 50, 100, 500 und 1000. Grössere Zahlen wurden einfach zusammengesetzt, so steht z.B. MMMDCCLXXVI für die Zahl 3876. Zum Rechnen war dieses Zahlensystem allerdings kaum geeignet und wurde im Laufe des 13. Jahrhunderts von den arabischen Ziffern abgelöst. Dazu beigetragen hat Anfang des 13. Jahrhunderts vor allem der Mathematiker Leonardo Fibonacci⁴ aus Pisa, der die Kenntnis der arabischen Ziffern mit seinem Werk '*Liber Abaci*' verbreitete. Detailliert erklärt er darin das Multiplizieren, Dividieren, Bruchrechnen, Potenzrechnen, Wurzelziehen und die Lösung von quadratischen und kubischen Gleichungen mit den arabischen Ziffern. Aus dem arabischen Wort *as-sifr* (die Leere) kreierte er den Namen *zefirum*, aus dem sich sowohl das Wort Ziffer, als auch das englische *zero* ableitet.

Das neue Zahlensystem mit der Null ermöglichte auch das Automatisieren von Rechenschritten. Im Jahre 1671 entwickelte Gottfried Wilhelm Leibniz⁵ seine Rechenmaschine, die REPLICA, die bereits alle vier Grundrechenarten beherrschte. Kurz darauf beschrieb er das binäre (oder auch duale) Zahlensystem, ohne das die heutige elektronische Datenverarbeitung kaum vorstellbar wäre. Auf dieses und weitere Zahlensysteme und die Implikationen auf arithmetische Operationen wollen wir im weiteren näher eingehen.

2.2 Maschinenzahlen

Die Menge der reellen Zahlen \mathbb{R} hat unendlich viele Elemente. Jede Rechenmaschine ist aber ein endlicher Automat, d.h. er kann aufgrund der beschränkten Stellenzahl nicht alle Zahlen exakt darstellen und nur endlich viele Operationen ausführen. Für eine gegebene Basis $B \in \mathbb{N}$ ($B > 1$) kann jede reelle Zahl $x \in \mathbb{R}$ aber als

$$x = m \times B^e$$

dargestellt werden, wobei $m \in \mathbb{R}$ die Mantisse und $e \in \mathbb{Z}$ der Exponent genannt wird.

Computerintern wird üblicherweise die Basis $B = 2$ verwendet (als Binär- od. Dualzahlen benannt), dies als direkte Folge der Zustände 'Strom' / 'kein Strom' (bzw. 1 und 0) von mikroelektronischen Schaltungen. Man spricht hier von einem *Bit* ('binary digit')⁶, welches genau zwei Werte, eben 0 oder 1 annehmen kann. Werden Bits zu Einheiten von 8 Bits zusammengefasst, spricht man von einem *Byte*. Weitere übliche Basen sind $B = 8$ (Oktalzahlen), $B = 10$ (Dezimalzahlen) und $B = 16$ (Hexadezimalzahlen). Für letztere benötigt man 16 verschiedene Zeichen und verwendet die Ziffern 0,1,...,9 sowie A,...,F (wobei $A \triangleq 10$, $B \triangleq 11$ etc., auch Kleinbuchstaben sind erlaubt).

Aufgabe 2.1:

1. Überlegen Sie sich: wie viele verschiedene Möglichkeiten gibt es, mit Binärzahlen ein Byte zu füllen?
2. Wie viele Ziffern bräuchten Sie im Hexadezimalsystem, um die gleiche Anzahl Möglichkeiten zu erhalten?
3. Was folgern Sie daraus bzgl. der Vorteile des Hexadezimalsystems?

Im Rechner stehen natürlich nur endlich viele Stellen für m und e zur Verfügung, z.B. n Stellen für m und l Stellen für e . Wir schreiben (entsprechend der englischen Schreibweise wird das Komma durch einen Punkt ersetzt):

$$\begin{aligned} m &= \pm 0.m_1m_2m_3\dots m_n \\ \text{und } e &= \pm e_1e_2\dots e_l \end{aligned}$$

⁴Namensgeber der Fibonacci Folge 1, 1, 2, 3, 5, 8, 13, ..., der damit um 1202 das Wachstum einer Kaninchenpopulation beschrieb. Die Folge war auch vor ihm bereits den Arabern und Indern bekannt.

⁵1646 - 1716, deutscher Universalgelehrter und, zusammen mit seinem Zeitgenossen Sir Isaac Newton (1643 - 1727), unter anderem Urheber der Integral- und Differentialrechnung

⁶Gemäss [7] wurde der Begriff *bit* das erste Mal wahrscheinlich von John Tukey (amerikanischer Mathematiker, 1915 - 2000, Träger der IEEE 'Medal of Honor') verwendet, als kürzere Alternative zu *bigit* oder *binit*. Das Wort *digit* kommt aus dem Lateinischen und bedeutet Finger.

Definition 2.1: Maschinenzahlen / Gleitpunktzahlen

- Unter der zusätzlichen Normalisierungsbedingung $m_1 \neq 0$ (falls $x \neq 0$) ergibt sich eine eindeutige Darstellung der sogenannten **maschinendarstellbaren Zahlen M** zur Basis B :

$$M = \{x \in \mathbb{R} \mid x = \pm 0.m_1 m_2 m_3 \dots m_n \cdot B^{\pm e_1 e_2 \dots e_l}\} \cup \{0\}$$

Dabei gilt $m_i, e_i \in \{0, 1, \dots, B-1\}$ für $i \neq 0$ und $B \in \mathbb{N}, B > 1$)

- Der **Wert** \hat{w} einer solchen Zahl ist definiert als

$$\hat{w} = \sum_{i=1}^n m_i B^{\hat{e}-i}$$

und ergibt gerade die (nicht normalisierte) Darstellung der Zahl im Dezimalsystem. Dabei ist \hat{e} hier ebenfalls im Dezimalsystem zu nehmen, also $\hat{e} = \sum_{i=1}^l e_i B^{l-i}$ und es gilt $\hat{e} \in \mathbb{Z}$, d.h. \hat{e} kann natürlich auch negativ sein. Weiter gibt es eine obere und untere Schranke: $\hat{e}_{min} \leq \hat{e} \leq \hat{e}_{max}$.

- Man spricht bei x auch von einer **n -steligen Gleitpunktzahl zur Basis B** (engl: floating point). Zahlen, die nicht in dieser Menge M liegen, müssen durch Rundung in eine maschinendarstellbare Zahl umgewandelt werden.

Bemerkungen:

- Der Exponent $\hat{e} \in \mathbb{Z}$ definiert, wie wir es vom Dezimalsystem kennen, die Position des Dezimalpunktes, also z.B. $x = 112.78350 = 112.78350 \cdot 10^0 = 1127835.0 \cdot 10^{-4} = 0.11278350 \cdot 10^3$. Eine Verschiebung des Dezimalpunktes um n Stellen nach links führt (bei gleichbleibendem Wert der Zahl) zu einer Erhöhung des Exponenten auf $e + n$. Eine Verschiebung des Dezimalpunktes um n Stellen nach rechts führt entsprechend zu einer Reduktion des Exponenten auf $e - n$. Analoges gilt für sämtliche andere Basen, z.B. bei $B = 2$: $x = 11001.111 = 11001.111 \cdot 2^0 = 0.11001111 \cdot 2^5$.
- Um Missverständnisse zu vermeiden, kann die Basis explizit als Index zu einer Mantisse in Klammern angegeben werden. Wird kein Exponent angegeben, ist das gleichbedeutend mit $\hat{e} = 0$, z.B. $(1011100.111)_2 = 1011100.111 \cdot 2^0 = 0.1011100111 \cdot 2^7 = (0.1011100111)_2 \cdot 2^7$.
- Die in Definition 2.1 gewählte Normalisierungsbedingung kann auch durch andere ersetzt werden. Was wären weitere Möglichkeiten? Weshalb normalisiert man überhaupt? Studieren Sie hierzu die folgenden Beispiele.

Beispiele 2.1 (teilweise gemäss [1]):

- Normalisierte Gleitpunktzahlen (gemäss Definition 2.1):

- $x_1 = -0.2345 \cdot 10^3$ ist eine vierstellige Gleitpunktzahl im Dezimalsystem mit dem Wert

$$-\sum_{i=1}^4 m_i \cdot 10^{3-i} = -(2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 + 5 \cdot 10^{-1}) = -234.5 (= -0.2345 \cdot 10^3)$$

- $x_2 = 0.111 \cdot 2^3$ ist eine dreistellige Gleitpunktzahl im Binär-/Dualsystem mit dem Wert

$$\sum_{i=1}^3 m_i \cdot 2^{3-i} = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 7 (= 0.7 \cdot 10^1)$$

- $x_3 = 0.1001 \cdot 2^{-3}$ ist eine vierstellige Gleitpunktzahl im Binär-/Dualsystem mit dem Wert

$$\sum_{i=1}^4 m_i \cdot 2^{-3-i} = 2^{-4} + 2^{-7} = 0.0703125 (= 0.703125 \cdot 10^{-1})$$

(d) $x_4 = 0.71537 \cdot 8^2$ ist eine fünfstellige Gleitpunktzahl im Oktalsystem mit dem Wert

$$\sum_{i=1}^5 m_i \cdot 8^{2-i} = 7 \cdot 8^1 + 1 \cdot 8^0 + 5 \cdot 8^{-1} + 3 \cdot 8^{-2} + 7 \cdot 8^{-3} = 57.685546875 (= 0.57685546875 \cdot 10^2)$$

(e) $x_5 = 0.AB3C9F \cdot 16^4$ ist eine sechsstellige Gleitpunktzahl im Hexadezimalsystem mit dem Wert

$$\sum_{i=1}^6 m_i \cdot 16^{4-i} = 10 \cdot 16^3 + 11 \cdot 16^2 + 3 \cdot 16^1 + 12 \cdot 16^0 + 9 \cdot 16^{-1} + 15 \cdot 16^{-2} = 43836.62109375 (= 0.4383662109375 \cdot 10^5)$$

2. Nicht normalisierte Gleitpunktzahlen:

Die obigen Beispiele x_1 bis x_5 sind alle gemäss Definition 2.1 normalisiert, d.h. die erste Ziffer vor dem Punkt ist Null, die erste Ziffer nach dem Punkt ist ungleich Null für $x \neq 0$. Damit sind ihre Mantisse und der Exponent eindeutig definiert. Die folgenden Beispiele geben zur Illustration nicht normalisierte Varianten für x_1 und x_2 . Es ist offensichtlich, dass eine nicht normalisierte Darstellung bzgl. Mantisse und Exponent nicht mehr eindeutig ist (auch wenn der Wert immer gleich bleibt). Dies ist ein Zustand, der bei der Speicherung vermieden werden muss.

$$(a) \tilde{x}_1 = -0.002345 \cdot 10^5 = -23.45 \cdot 10^2 = -234500 \cdot 10^{-3} = \dots$$

$$(b) \tilde{x}_2 = 0.0111 \cdot 2^4 = 1.11 \cdot 2^2 = 111 \cdot 2^0 = 11100 \cdot 2^{-2} = \dots$$

3. IEC/IEEE⁷ - Gleitpunktzahlen. Es gibt zwei Grundformate zur Basis $B = 2$:

(a) Single precision (einfache Genauigkeit). Gesamtlänge der Zahl ist 32 Bit (4 Bytes), davon 1 Bit für das Vorzeichen, 23 Bit für die Mantisse, und 8 Bit für den Exponenten:

single: (32 bit)

V	EEEEEEE	MMMMMM	MMMMMM	MMMMMM
0	1	8	9	31
0	10000000	00000000000000000000000000000000	=	$+1.0 * 2^{128-127} = 2$
0	10000001	10100000000000000000000000000000	=	$+1.101 * 2^{129-127} = 6.5$
1	10000001	10100000000000000000000000000000	=	$-1.101 * 2^{129-127} = -6.5$
0	00000001	00000000000000000000000000000000	=	$+1.0 * 2^{1-127} = 2^{-126} = x_{\min}$
0	00000000	10000000000000000000000000000000	=	$+0.1 * 2^{-126} = 2^{-127}$
0	00000000	00000000000000000000000000000001	=	$+0.0\dots01 * 2^{-126} = 2^{-149}$ = kleinste darstellbare Zahl

(b) Double precision (doppelte Genauigkeit). Gesamtlänge der Zahl ist 64 Bit (8 Bytes), wobei 1 Bit für das Vorzeichen, 52 Bit für die Mantisse, und 11 Bit für den Exponenten:

double: (64 bit)

V	EEEEEEEEE	MMMMMM	MMMMMM	MMMMMM	MMMMMM	MMMMMM
0	1	11	12			63

Bei obigen IEEE-Formaten ist die Mantisse so normalisiert, dass das erste Bit vor dem Punkt ungleich 0 sein muss. Und da dieses Bit bei normalisierten Zahlen immer 1 ist, wird es nicht gespeichert (hidden bit). Man gewinnt so ein Bit an Mantissenlänge. Das Bit V erzeugt das Vorzeichen der Zahl: $V = 0$ entspricht einem positiven, $V = 1$ einem negativen Vorzeichen. Außerdem wird im Exponenten ein von der Anzahl Bits abhängiger Biaswert subtrahiert, womit der Exponent kein eigenes Vorzeichenbit benötigt. Der Wert einer Gleitpunkt-Zahl x im IEEE-Format lautet also:

$$x = (-1)^V \cdot (\underbrace{1}_{\text{hidden bit}} \cdot \underbrace{\text{.} MMM \dots MMM}_{\text{fraction}}) \cdot 2^{(E-E)-bias}$$

Mantisse

⁷Institute of Electrical and Electronics Engineers

4. Wieso rechnet man eigentlich mit Gleitpunktzahlen? Nimmt man, abgesehen vom Vorzeichen, an, dass 8 dezimale Speichereinheiten zur Verfügung stehen, so liessen sich damit in den folgenden Systemen die folgenden positiven Zahlen darstellen:

(a) Ganzzahlsystem:

- i. kleinste darstellbare positive Zahl: 0000 0001
- ii. grösste darstellbare positive Zahl: 9999 9999

Es lassen sich also im positiven Bereich alle ganzen Zahlen zwischen 1 und 99 999 999 ($= 10^8 - 1$) hinterlegen. Der Abstand zwischen aufeinanderfolgenden Zahlen ist konstant gleich 1.

(b) Festpunktsystem mit 4 Dezimalen:

- i. kleinste darstellbare positive Zahl: 0000.0001
- ii. grösste darstellbare positive Zahl: 9999.9999

Es lassen sich im positiven Bereich Zahlen zwischen 0.0001 und 9999.9999 ($= 10^4 - 10^{-4}$) darstellen. Der Abstand zwischen aufeinanderfolgenden Zahlen ist konstant gleich 10^{-4} .

(c) Normalisiertes Gleitpunktsystem mit 6 Mantissen- und 2 Exponentenziffern (mit Bias 50)

- i. kleinste darstellbare positive Zahl: $0.100000 \cdot 10^{-50}$
- ii. grösste darstellbare positive Zahl: $0.999999 \cdot 10^{49}$

Es lassen sich im positiven Bereich Zahlen von 10^{-51} bis $10^{49} - 1$ darstellen. Der Abstand zwischen aufeinanderfolgenden Zahlen ist allerdings variabel, wie in Kap. 2.3 gezeigt wird.

Offensichtlich ist der darstellbare Zahlenbereich bei gleichem Speicherbedarf bei Gleitpunktzahlen enorm gross.

Aufgabe 2.2 [1]:

1. Wie viele Stellen benötigt man für die Mantisse, um die folgenden Zahlen als n-stellige Gleitpunktzahlen im Dezimalsystem darzustellen? Wie gross ist der zugehörige Exponent?

$$x_1 = 0.00010001, x_2 = 1230001, x_3 = \frac{4}{5}, x_4 = \frac{1}{3}$$

2. Bestimmen Sie alle dualen positiven 3-stelligen Gleitpunktzahlen mit einstelligem positiven binären Exponenten sowie ihren dezimalen Wert.

3. Wie viele verschiedene Maschinenzahlen gibt es auf einem Rechner, der 20-stellige Gleitpunktzahlen mit 4-stelligen binären Exponenten sowie dazugehörige Vorzeichen im Dualsystem verwendet? Wie lautet die kleinste positive und die grösste Maschinenzahl?

4. Verstehen Sie den folgenden 'Witz'?

Es gibt 10 Gruppen von Menschen: Diejenigen, die das Binärsystem verstehen, und die anderen.

2.3 Approximations- und Rundungsfehler

Die Maschinenzahlen sind nicht gleichmässig verteilt. Ein Beispiel für alle binären normalisierten Gleitpunktzahlen mit 4-stelliger Mantisse und 2-stelligem Exponenten ist in Abb. 2.1 dargestellt. Zwangsläufig gibt es bei jedem Rechner eine grösste (x_{max}) und kleinste positive Maschinenzahl (x_{min}). Dabei gilt für normalisierte Gleitpunktzahlen:⁸

$$\begin{aligned} x_{max} &= B^{e_{max}} - B^{e_{max}-n} = (1 - B^{-n})B^{e_{max}} \\ x_{min} &= B^{e_{min}-1} \end{aligned}$$

⁸Wird auf die Normalisierung der Mantisse ($m_1 \neq 0$) verzichtet, führt dies zu sogenannten subnormalen Zahlen, die bis B^{m-n} hinunter reichen (IEEE Standard 754).

Aufgabe 2.3:

- Schreiben Sie die kleinste und grösste binäre positive Maschinenzahl für Abb. 2.4 explizit auf und berechnen Sie deren Wert. Stimmt das mit x_{max} und x_{min} überein?

Zahlen, die ausserhalb des Rechenbereichs $[-x_{max}, x_{max}]$ liegen, sind im *Überlaufbereich (overflow)* und führen zum Abbruch der Rechnung (mit IEEE 754 konforme Systeme geben die Bitsequenz *inf* aus). Zahlen ungleich 0, die innerhalb des Bereichs $[-x_{min}, x_{min}]$ liegen, führen zu einem *Unterlauf (underflow)*. Dann ist es sinnvoll, die Rechnung mit 0 weiterzuführen.

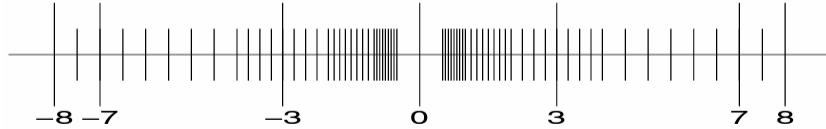


Abbildung 2.4: Alle binären Maschinenzahlen mit $n = 4$ und $0 \leq e \leq 3$ (aus [1])

Offensichtlich ist die Anzahl n der Mantissestellen von entscheidender Bedeutung für den Bereich der Zahlen, die abgebildet werden können. Dies wird eindrücklich illustriert in folgendem Beispiel:

Beispiel 2.2 [7]:

- Am 4. Juni 1996 startete zum ersten Mal eine Ariane 5-Rakete der ESA von Französisch Guyana aus. Die unbemannte Rakete hatte vier Satelliten an Bord. 36.7 Sekunden nach dem Start wurde in einem Programm versucht, den gemessenen Wert der horizontalen Geschwindigkeit von 64 Bit Gleitpunktdarstellung in 16 Bit Ganzahldarstellung (signed Integer) umzuwandeln. Da die entsprechende Masszahl grösser war als $2^{15} = 32768$, wurde ein Überlauf erzeugt. Das Lenksystem versagte daraufhin seine Arbeit und gab die Kontrolle an eine zweite, identische Einheit ab. Diese produzierte folgerichtig ebenfalls einen Überlauf. Da der Flug der Rakete instabil wurde und die Triebwerke abzubrechen drohten, zerstörte sich die Rakete selbst. Es entstand ein Schaden von ca. 500 Millionen Dollar durch den Verlust der Rakete und der Satelliten. Die benutzte Software stammte vom Vorgängermodell Ariane 4. Die Ariane 5 flog schneller und offensichtlich wurde dies bei der Repräsentation der Geschwindigkeit nicht beachtet.

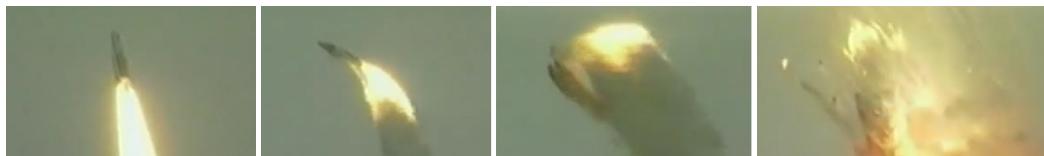


Abbildung 2.5: Explosion der Ariane 5 am 4. Juni 1996 (von https://youtu.be/gp_D8r-2hwk).

2.3.1 Rundungsfehler und Maschinengenauigkeit

Aus Abb. 2.1 wird deutlich, dass jede reelle Zahl, die von einem Rechner verwendet werden soll, aber selber keine Maschinenzahl ist, durch eine solche ersetzt werden muss. Dabei entstehen Fehler.

Definition 2.2: Absoluter / Relativer Fehler

- Hat man eine Näherung \tilde{x} zu einem exakten Wert x , so ist der Betrag der Differenz $|\tilde{x} - x|$ der **absolute Fehler**.
- Falls $x \neq 0$, so ist $|\frac{\tilde{x}-x}{x}|$ bzw. $\frac{|\tilde{x}-x|}{|x|}$ der **relative Fehler** dieser Näherung.

Bemerkung: In der Numerik ist der relative Fehler der wichtigere. Weshalb?

Natürlich sollte die Maschinenzahl dabei so gewählt werden, dass sie möglichst nahe bei der reellen Zahl liegt. Einfaches Abschneiden ist dazu nicht geeignet. Ein besseres Verfahren ist die Rundung (vgl. Aufgabe 2.4). Beim Runden einer Zahl x wird eine Näherung unter den Maschinenzahlen gesucht, die einen minimalen absoluten Fehler $|rd(x) - x|$ aufweist. Eine n -stellige dezimale Gleitpunktzahl $\tilde{x} = 0.m_1m_2m_3\dots m_n \cdot 10^e = rd(x)$, die durch die Rundung eines exakten Wertes x entstanden ist, hat also einen absoluten Fehler von höchstens

$$|rd(x) - x| \leq 0.\underbrace{00\dots 00}_n 5 \cdot 10^e = 0.5 \cdot 10^{e-n},$$

wobei die 5 an der Stelle $n+1$ nach dem Dezimalpunkt auftritt. Für eine beliebige (gerade) Basis gilt analog:

$$\text{falls } B \text{ gerade: } |rd(x) - x| \leq 0.\underbrace{00\dots 00}_n \frac{B}{2} \cdot B^e = \frac{B}{2} \cdot B^{e-n-1},$$

Beispiel 2.3 [6]:

- Sei $x = 180.1234567 = 0.1801234567 \cdot 10^3$. Gerundet auf eine siebenstellige Mantisse ($n = 7$) erhält man $rd(x) = 0.1801235 \cdot 10^3$ und es gilt wegen $e = 3$

$$|rd(x) - x| = 0.\underbrace{0000000}_7 433 \cdot 10^3 = 0.433 \cdot 10^{-4} \leq 0.5 \cdot 10^{-4}$$

Aufgabe 2.4

- Vergewissern Sie sich anhand einfacher Zahlenbeispiele, dass die Rundung ein besseres Verfahren für die Abbildung einer reellen Zahl auf eine Maschinenzahl darstellt als einfaches Abschneiden der überzähligen Ziffern. Was ist der maximale Fehler, der durch das Abschneiden auftreten kann?
- Wir kennen die (allgemeinen) Rundungsregeln für das Dezimalsystem. Verallgemeinern Sie diese für eine beliebige gerade Basis B . Runden Sie anschliessend die folgenden Zahlen auf eine vierstellige Mantisse, berechnen Sie den absoluten Fehler der Rundung und vergewissern Sie sich, dass $|rd(x) - x| \leq \frac{B}{2} \cdot B^{e-n-1}$. Gilt diese Relation (bei gleichen Rundungsregeln) auch für ungerade Basen?
 - $(11.0100)_2$
 - $(11.0110)_2$
 - $(11.111)_2$
 - $(120.212)_3$
 - $(120.222)_3$
 - $(0.FFFF)_16$

Für die Berechnungen bedeutet das, dass jede einzelne Operation (+, -, *, ...) auf $n+1$ Stellen genau gerechnet wird, und das Ergebnis auf n Stellen gerundet wird (*n-stellige Gleitpunktarithmetik*). Jedes Zwischenergebnis wird also gerundet, nicht erst das Endergebnis. Das bedeutet auch, dass die einzelnen Rundungsfehler durch die Rechnung weitergetragen werden und allenfalls das Endergebnis verfälschen können. Für den maximal auftretenden relativen Fehler bei der Rundung ergibt sich bei n -stelliger Gleitpunktarithmetik im Dezimalsystem:

$$\left| \frac{rd(x) - x}{x} \right| \leq 5 \cdot 10^{-n} \quad (\text{da } x \geq 10^{e-1}).$$

Dies führt zum Begriff der Maschinengenauigkeit:

Definition 2.3: Maschinengenauigkeit

- Die Zahl $eps := 5 \cdot 10^{-n}$ heisst **Maschinengenauigkeit**. Bei allgemeiner Basis B gilt

$$eps := \frac{B}{2} \cdot B^{-n} = \frac{1}{2} \cdot B^{1-n}.$$

Sie entspricht dem maximalen relativen Fehler, der durch Rundung entstehen kann.

Bemerkungen:

- Die Maschinengenauigkeit ϵ_{ps} gemäss Def. 2.3 kann auch interpretiert werden als die grösste positive Maschinenzahl, für die auf dem Rechner $1 + \epsilon_{\text{ps}} = 1$ gerundet wird.
- ACHTUNG: Im IEEE-754 Standard und damit auch in Python, Matlab, etc. wird die Maschinengenauigkeit anders definiert: Dort entspricht ϵ_{ps} dem Abstand zwischen 1 und der nächstgrösseren Maschinenzahl und ist damit genau doppelt so gross wie ϵ_{ps} gemäss Def. 2.3. Falls nicht explizit anders deklariert, gilt in diesen Unterlagen stets die Maschinengenauigkeit gemäss Def. 2.3.
- Die Maschinengenauigkeit ϵ_{ps} darf nicht mit der (viel kleineren) kleinsten positiven Maschinenzahl x_{\min} verwechselt werden. Ein Rechner kann also auch mit deutlich kleineren Zahlen $x < \epsilon_{\text{ps}}$ noch 'genau' (nämlich mit einem relativen Rundungsfehler $\leq \epsilon_{\text{ps}}$) rechnen.

Beispiel 2.4a:

- Für das Format double precision in IEEE-754 gilt für die Mantisse $n = 53$ (hidden bit!), und damit ist für die Basis $B = 2$ die Maschinengenauigkeit (gemäss Def. 2.3)

$$\epsilon_{\text{ps}} = \frac{1}{2} \cdot B^{1-n} = \frac{1}{2} \cdot 2^{1-53} = 2^{-53} = 1.110223\dots \cdot 10^{-16}.$$

Beispiel 2.4b:

- Am Freitag, dem 25. November 1983, schloss der Aktienindex von Vancouver bei 524.811 Punkten und eröffnete am folgenden Montag, dem 28. November, bei 1098.892 Punkten. Was war passiert? Seit dem Start im Januar 1982 bei 1000 Punkten war der Aktienindex kontinuierlich gefallen, trotz florierendem Handel und guter Wirtschaftslage. Der Index wurde ca. 3000 mal am Tag neu berechnet, jeweils auf vier Dezimalstellen genau. Doch statt auf drei Dezimalstellen zu runden, wurde die vierte Dezimalstelle einfach abgeschnitten. Der dabei maximal mögliche Fehler von 0.0009 mutet zwar klein an, doch bei 3000 Wiederholungen pro Tag konnte sich dieser Abschneidefehler auf bis zu $0.0009 \cdot 3000 = 2.7$ Punkte pro Tag aufsummieren. Über die Zeitspanne von fast zwei Jahren verlor der Index so fast die Hälfte seines Wertes. Dies wurde am 28. November basierend auf korrekter Rundung korrigiert. Grössere Auswirkungen hatte diese Korrektur offenbar nicht, da zum damaligen Zeitpunkt das Volumen an Derivaten gering war.

Aufgabe 2.5 [1]:

1. Gesucht ist eine Näherung \tilde{x} zu $x = \sqrt{2} = 1.414213562\dots$ mit einem absoluten Fehler von höchstens 0.001.
2. Es soll $2590 + 4 + 4$ in 3-stelliger Gleitpunktarithmetik gerechnet werden (im Dezimalsystem), einmal von links nach rechts und einmal von rechts nach links. Wie unterscheiden sich die Resultate?
Anhand der Lösung sieht man, dass es bei der n-stelligen Gleitpunktarithmetik auf die Reihenfolge der Operationen ankommt, im Unterschied zum exakten Rechnen. Als Faustregel kann man festhalten:
Beim Addieren sollte man die Summanden in der Reihenfolge aufsteigender Beträge sortieren
3. Berechnen Sie $s_{300} := \sum_{i=1}^{300} \frac{1}{i^2}$ sowohl auf- als auch absteigend, je einmal mit 3-stelliger und 5-stelliger Gleitpunktarithmetik. In Python müssen Sie dafür zuerst eine Funktion schreiben, die Ihnen eine reelle Zahl (hier $1/r^2$) auf die nächstgelegene Maschinenzahl mit $B = 10$ und $n = 3$ oder $n = 5$ runden.
4. Es ist $\lim_{n \rightarrow \infty} (1 + \frac{1}{n})^n = e$. Erstellen Sie eine Tabelle mit ihrem Rechner oder Python für $n = 1, 10, 100, \dots$ für den Ausdruck $(1 + \frac{1}{n})^n$ sowie den absoluten und relativen Fehler. Erklären Sie Ihre Beobachtungen.

Lösung:

n	$(1 + \frac{1}{n})^n$	absoluter Fehler	relativer Fehler
10^0			
10^2			
10^3			
10^4			
10^5			
10^6			
10^8			
10^9			
10^{10}			
10^{15}			
10^{16}			

5. Überlegen Sie sich einen kurzen iterativen Algorithmus, der die Maschinengenauigkeit Ihres Rechners prüft. Schliessen Sie aus dem Ergebnis, mit welcher Stellenzahl er operiert.

2.3.2 Fehlerfortpflanzung bei Funtionsauswertungen / Konditionierung

Wir haben gesehen, dass ein Rundungsfehler durch die Abbildung einer reellen Zahl x auf ihre Maschinenzahl \tilde{x} in die Berechnungen einfließt. Soll nun eine Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ an der Stelle x ausgewertet werden, wird ein zusätzlicher Fehler dadurch generiert, dass nicht $f(x)$, sondern $f(\tilde{x})$ berechnet wird⁹. Für den fehlerbehafteten Wert \tilde{x} können wir den Fehler quantifizieren als $\Delta x = \tilde{x} - x$ (vgl. Def. 2.2) oder

$$\tilde{x} = x + \Delta x$$

Nun wollen wir den absoluten Fehler $|f(\tilde{x}) - f(x)|$ und den relativen Fehler $\frac{|f(\tilde{x}) - f(x)|}{|f(x)|}$ dieser Funktionsauswertung berechnen. Unter der Annahme, dass die Funktion f stetig differenzierbar ist, können wir $f(\tilde{x})$ gemäss Taylor entwickeln. Aus der allg. Taylor-Reihe (bekannt aus der Analysis) einer Funktion $f(x)$ um den Entwicklungspunkt x_0

$$f(x) = \sum_{i=0}^{\infty} \frac{f^{(i)}(x_0)}{i!} (x - x_0)^i$$

erhalten wir für die Entwicklung von $f(\tilde{x})$ um den Entwicklungspunkt x

$$f(\tilde{x}) = f(x + \Delta x) = \sum_{i=0}^{\infty} \frac{f^{(i)}(x)}{i!} (\Delta x)^i = f(x) + f'(x)\Delta x + \frac{f''(x)}{2}(\Delta x)^2 + \dots$$

wobei wir in der Taylor-Reihe x durch \tilde{x} und x_0 durch x ersetzt haben.

Unter der Annahme $\Delta x \ll 1$ können die höheren Fehlerterme $(\Delta x)^n$ für $n \geq 2$ vernachlässigt werden und es ergibt sich die folgende Näherung

$$\begin{aligned} f(\tilde{x}) - f(x) &\approx f'(x)\Delta x \\ &\approx f'(x)(\tilde{x} - x) \end{aligned}$$

bzw. bei beidseitiger Division durch $f(x)$ und rechtsseitiger Multiplikation mit $\frac{x}{x}$:

$$\frac{f(\tilde{x}) - f(x)}{f(x)} \approx \frac{f'(x) \cdot x}{f(x)} \cdot \frac{\tilde{x} - x}{x}$$

Wir erhalten also die folgenden Näherungen:

⁹Tatsächlich ist die Situation noch schlimmer, denn bereits die Umsetzung der reellwertigen Funktion selbst kann in einem endlichen Rechner zu zusätzlichen Fehlern führen. Darauf wollen wir an dieser Stelle aber nicht eingehen.

- Näherung für den **absoluten Fehler bei Funktionsauswertungen**:

$$\underbrace{|f(\tilde{x}) - f(x)|}_{\text{absoluter Fehler von } f(x)} \approx |f'(x)| \cdot \underbrace{|\tilde{x} - x|}_{\text{absoluter Fehler von } x}$$

- Näherung für den **relativen Fehler bei Funktionsauswertungen**:

$$\underbrace{\frac{|f(\tilde{x}) - f(x)|}{|f(x)|}}_{\text{relativer Fehler von } f(x)} \approx \underbrace{\frac{|f'(x)| \cdot |x|}{|f(x)|}}_{\text{Konditionszahl } K} \cdot \underbrace{\frac{|\tilde{x} - x|}{|x|}}_{\text{relativer Fehler von } x}$$

Definition 2.4: Konditionszahl

- Den Faktor

$$K := \frac{|f'(x)| \cdot |x|}{|f(x)|}$$

nennt man **Konditionszahl**. Er gibt näherungsweise an, um wieviel sich der relative Fehler von x vergrössert bei einer Funktionsauswertung $f(x)$.

- Man unterscheidet **gut konditionierte Probleme**, d.h. die Konditionszahl ist klein, und **schlecht konditionierte Probleme** (ill posed problems) mit grosser Konditionszahl. Bei gut konditionierten Problemen wird der relative Fehler durch die Auswertung der Funktion nicht grösser.

Bemerkungen:

- \tilde{x} kann generell als fehlerbehafteter Näherungswert für x angesehen werden. Ob der Fehler nun durch Rundung oder andere Effekte verursacht wird (z.B. durch fehlerhafte Messungen) ist hierbei nicht von Belang.
- Bei Funktionsauswertungen pflanzt sich der absolute Fehler in x näherungsweise mit dem Faktor $f'(x)$ fort. Falls $|f'(x)| > 1$ wird der absolute Fehler grösser, falls $|f'(x)| < 1$ kleiner.
- Bei Funktionsauswertungen pflanzt sich der relative Fehler in x näherungsweise mit der Konditionszahl fort.

Beispiel 2.5 [1]:

- Was lässt sich über die Fehlerfortpflanzung des absoluten Fehlers für die Funktion $f(x) = \sin(x)$ aussagen? Da $|f'(x)| = |\cos(x)| \leq 1$, folgt dass der absolute Fehler in den Funktionswerten nicht grösser sein kann als in den x -Werten sondern eher kleiner.
- Bei der Funktion $f(x) = 1000 \cdot x$ wird wegen $f'(x) = 1000$ der absolute Fehler in der Funktionsauswertung um den Faktor 1000 grösser.
- Die Konditionszahl für das Quadrieren, also $f(x) = x^2$, ist $K = \frac{|2x| \cdot |x|}{|x^2|} = 2$, d.h. der relative Fehler verdoppelt sich in etwa. Dies ist aber noch keine schlechte Konditionierung.
- Als Beispiel für ein schlecht konditioniertes Problem betrachten wir das Wilson-Polynom, definiert als

$$P(x) = \prod_{k=1}^{20} (x - k) = (x - 1)(x - 2) \cdot \dots \cdot (x - 20)$$

mit den Nullstellen

$$x_k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}.$$

Ausmultipliziert erhält man

$$\begin{aligned}
P(x) = & x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16} - 1672280820x^{15} \\
& + 40171771630x^{14} - 756111184500x^{13} + 11310276995381x^{12} - 135585182899530x^{11} \\
& + 1307535010540395x^{10} - 10142299865511450x^9 + 63030812099294896x^8 \\
& - 311333643161390640x^7 + 1206647803780373360x^6 - 3599979517947607200x^5 \\
& + 8037811822645051776x^4 - 12870931245150988800x^3 + 13803759753640704000x^2 \\
& - 8752948036761600000x + 2432902008176640000
\end{aligned}$$

Wird nun der Koeffizient -210 von x^{19} um nur 2^{-23} ($\approx 1.1920928 \cdot 10^{-7}$) verkleinert (resp. "gestört") auf neu -210.0000001192093 , verändert sich der Funktionswert $P(20)$ von ursprünglich 0 auf $-6.24997 \cdot 10^{17}$ und die Nullstellen des gestörten Polynoms werden zu

$$\begin{aligned}
\tilde{x}_k \in & \{1.00000, 2.00000, 3.00000, 4.00000, 5.00000, 6.00001, 6.99970, 8.00727, 8.91725, \\
& 10.09527 \pm 0.64350i, 11.79363 \pm 1.65233i, 13.99236 \pm 2.51883i, 16.73074 \pm 2.81262i, \\
& 19.50244 \pm 1.94033i, 20.84691\},
\end{aligned}$$

zum Teil also sogar aus dem komplexen Zahlenraum \mathbb{C} (auf die komplexen Zahlen werden wir in Kapitel (4) näher eingehen). Während nur ein Koeffizient von P mit dem absoluten Fehler von $\sim 10^{-7}$ gestört wurde, haben sich die Nullstellen um bis zu ~ 1 verändert, d.h. die Störung wurde um einen Faktor von $\sim 10^7$ verstärkt. Die Verstärkung des relativen Fehlers liegt in einem ähnlichen Bereich. Die Nullstellen von P sind also schlecht konditioniert. Schaut man sich den Graphen von $P(x)$ und der gestörten Version in Abb. (2.6), sieht man die Auswirkungen der Störung.

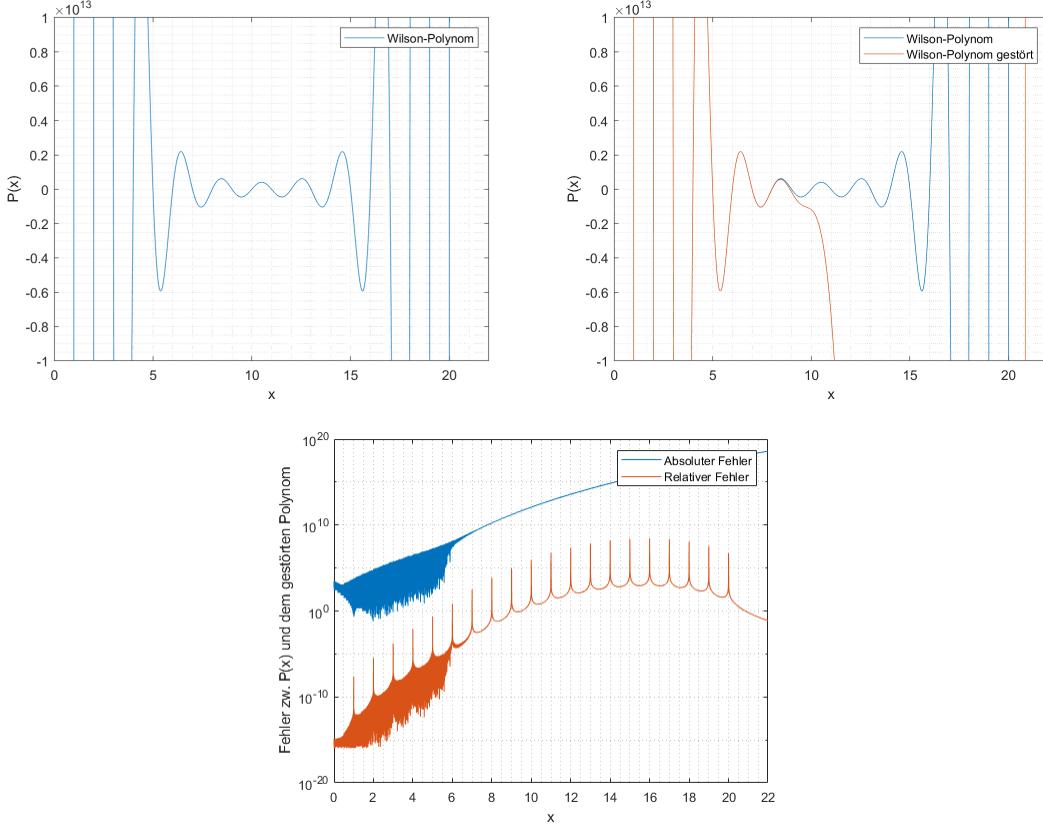


Abbildung 2.6: Das Wilson-Polynom und die gestörte Version.

Betrachten wir nun als weitere Beispiele die relativen Fortpflanzungsfehler für die grundlegenden arithmetischen Operationen.

Fehlerfortpflanzung bei Summation

Für

$$f(x) = x + c \quad (c \in \mathbb{R})$$

haben wir für die Ableitung $f'(x) = 1$ und damit

$$\frac{|f(\tilde{x}) - f(x)|}{|f(x)|} \approx \frac{|x|}{|x+c|} \cdot \frac{|\tilde{x}-x|}{|x|}$$

bzw.

$$K = \frac{|x|}{|x+c|}$$

Wenn x und die Konstante c gleiches Vorzeichen haben, gilt $K \leq 1$, dann haben wir also ein gut konditioniertes Problem. Was passiert aber, wenn x und c entgegengesetzte Vorzeichen haben und betragsmäßig fast gleich gross sind? Dann wird $|x+c|$ sehr klein und somit K sehr gross, die Addition (bzw. Subtraktion) ist dann schlecht konditioniert. Dieses Phänomen nennt man auch **Auslöschung**. Es tritt immer dann auf, wenn ungefähr gleich grosse fehlerbehaftete Zahlen voneinander abgezogen werden und das Resultat anschliessend normalisiert wird.

Beispiel 2.6 [7]:

- Für die beiden reellen Zahlen $r = \frac{3}{5}$ und $s = \frac{4}{7}$ mit den normalisierten gerundeten Repräsentationen mit fünfstelliger Mantisse, also $\tilde{r} = (0.10011)_2$ und $\tilde{s} = (0.10010)_2$, berechnen wir die Differenz $r - s = \frac{1}{35}$ näherungsweise als

$$0.10011 \cdot 2^0 - 0.10010 \cdot 2^0 = 0.00001 \cdot 2^0 = 0.10000 \cdot 2^{-4} = \frac{1}{32}.$$

Für den relativen Fehler erhalten wir

$$\frac{\frac{1}{32} - \frac{1}{35}}{\frac{1}{35}} = 0.0938 \approx 9.4\%$$

was viel ist (zum Vergleich, dies ist rund dreimal grösser als die Maschinengenauigkeit $2^{-5} = 0.0313$). Für die Berechnung mit dreistelliger Mantisse erhalten wir

$$0.101 - 0.101 = 0$$

und damit einen Fehler von 100%.

Beispiel 2.7 [6]:

- Gegeben seien die drei Werte

$$\begin{aligned} x_1 &= 123.454 \cdot 10^9 \\ x_2 &= 123.446 \cdot 10^9 \\ x_3 &= 123.435 \cdot 10^9 \end{aligned}$$

Legt man eine 5-stellige dezimale Gleitpunktarithmetik zugrunde, so wird durch Rundung

$$\begin{aligned} \tilde{x}_1 &= 0.12345 \cdot 10^{12} \\ \tilde{x}_2 &= 0.12345 \cdot 10^{12} \\ \tilde{x}_3 &= 0.12344 \cdot 10^{12} \end{aligned}$$

und man erhält statt

$$\begin{aligned} x_1 - x_2 &= x_1 + (-x_2) = 8 \cdot 10^6 \\ x_1 - x_3 &= x_1 + (-x_3) = 19 \cdot 10^6 \end{aligned}$$

die fehlerhaften Werte

$$\begin{aligned} \tilde{x}_1 - \tilde{x}_2 &= 0 \\ \tilde{x}_1 - \tilde{x}_3 &= 10 \cdot 10^6 \end{aligned}$$

Aufgabe 2.6:

1. Untersuchen Sie, ob die Multiplikation und die Division zweier Zahlen gut oder schlecht konditionierte Funktionsauswertungen sind.

Kapitel 3

Numerische Lösung von Nullstellenproblemen

In diesem Kapitel behandeln wir Verfahren zur näherungsweisen Lösung von nichtlinearen Gleichungen mit einer Unbekannten (die Lösung linearer Gleichungen einer Variablen ist trivial). Wie wir sehen werden, ist die Lösung von nichtlinearen Gleichungen mit einer Unbekannten äquivalent zur Bestimmung der Nullstellen einer Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ mit $f(x) = 0$, deshalb der Titel.

Lernziele:

- Sie können die Begriffe Fixpunktgleichung, Fixpunktiteration sowie anziehender bzw. abstoßender Fixpunkt definieren.
- Sie können zu einer konkreten Aufgabenstellung die Fixpunktgleichung aufstellen und die entsprechende Iteration durchführen.
- Sie können dabei auftretende Fehler mittels des Banachschen Fixpunktsatzes quantifizieren.
- Sie können das Newtonverfahren, das vereinfachte Newtonverfahren sowie das Sekantenverfahren anwenden.
- Sie verstehen der Begriff der Konvergenzordnung.

Bemerkung: Die im Kapitel 2 verwendete Normalisierung $x = \pm 0.m_1m_2m_3\dots m_n \cdot B^e$ haben wir im Zusammenhang mit der Theorie der Rechnerarithmetik und der maschinendarstellbaren Zahlen zu verschiedenen Basen eingeführt. In den Ingenieurwissenschaften werden numerische Resultate aber meist als Dezimalzahlen in der Potenzschreibweise dargestellt mit vier Nachkommastellen, wobei die erste Ziffer vor dem Komma *ungleich 0* sein muss (für $x \neq 0$). Sofern wir im weiteren mit numerischen Resultaten arbeiten und es nicht ausdrücklich anders verlangt ist, werden wir also im Dezimalsystem i.d.R. mit der Darstellung $x = \pm m_1.m_2m_3m_4m_5 \cdot 10^{\pm e}$ mit $m_1 \neq 0$ (für $x \neq 0$) arbeiten.

3.1 Zur historischen Entwicklung

Die Fragestellung der Lösung nichtlinearer Gleichungen begleitet die (numerische) Mathematik seit ihren Anfängen. Die Babylonier (und vermutlich bereits die Ägypter) beschäftigten sich in ihrer auf die Geometrie fokussierten Mathematik unter anderem mit der Frage, wie gross die Seitenlängen x eines Quadrates mit der gegebenen Fläche A sind, also mit der Lösung der nichtlinearen Gleichung $x^2 = A$ (vgl. die Lösung in Abb. 1.1 für $A = 2$). Eng damit verwandt ist natürlich die Fragestellung des Flächeninhaltes eines rechtwinkligen Dreiecks. Der nach dem griechischen Philosophen Pythagoras von Samos (um 570-510 v.Chr.) benannte Satz $a^2 + b^2 = c^2$ war den Babylonier bereits rund 1000 Jahre früher bekannt. Die Übersetzung einer babylonischen Tontafel (ca. 1900-1600 v.Chr.) im

Britischen Museum lautet¹:

4 is the length and 5 the diagonal. What is the breadth?

Its size is not known.

4 times 4 is 16.

5 times 5 is 25.

You take 16 from 25 and there remains 9.

What times what shall I take in order to get 9?

3 times 3 is 9.

3 is the breadth.

Der Griechen Heron von Alexandria (1. Jhr. n.Chr.) beschrieb in seinem Werk *Metrika* (Buch der Messung) das nach ihm benannte Näherungsverfahren von Heron zur iterativen Berechnung einer (beliebigen) Quadratwurzel $x = \sqrt{A}$ für $A > 0$ mit der Iterationsvorschrift (für einen Startwert $x_0 \neq 0$):

$$x_{n+1} = \frac{x_n + \frac{A}{x_n}}{2}$$

Im Mittelalter konzentrierte man sich auf die Nullstellensuche von Polynomen. Der italienische Mathematiker Girolamo Cardano (1501-1576) veröffentlichte als erster Lösungsformeln (die Cardanischen Formeln) für kubische Gleichungen und zusätzlich Lösungen für Gleichungen vierten Grades. Bei seinen Berechnungen stieß er auf die komplexen Zahlen und zeigte (entgegen der damaligen Lehrmeinung), dass auch mit negativen Zahlen gerechnet werden kann.

Isaac Newton beschrieb im Zeitraum 1664 bis 1671 einen neuen Algorithmus zur Nullstellenbestimmung von Polynomen dritten Grades. Sein Landsmann und Mathematiker Thomas Simpson (1710-1761) formulierte dieses Verfahren unter Benutzung der Ableitung in der Iterationsvorschrift

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

was wir heute als Newton-Verfahren bezeichnen (vgl. Kap. 3.4). Tatsächlich ist das Newton-Verfahren äquivalent zum Heron-Verfahren für die Bestimmung der Nullstellen der Funktion $f(x) = x^2 - A$. Generell lässt sich das Newton-Verfahren natürlich (unter gewissen Einschränkungen bzgl. der Konvergenz) für beliebige stetig differenzierbare Funktionen $f(x)$ einsetzen, nicht nur für Polynome.

Wahrscheinlich im Zusammenhang mit dem Beweis des Zwischenwertsatzes der Analysis konstruierte der böhmische Priester und Mathematiker Bernard Bonzano (1781-1848) um 1817 das Bisektionsverfahren², welches es durch fortlaufende Intervallhalbierung zuverlässig (aber langsam) erlaubt, eine Nullstelle einer stetigen Funktion zu finden (ohne Benutzung der Ableitung wie im Newton-Verfahren). Der polnische Mathematiker Stefan Banach (1892-1945) formulierte 1922 den Banachschen Fixpunktsatz zur Theorie der Fixpunktiterationen (siehe Kap. 3.3), die zur Lösung von Nullstellenproblemen in einem weit gefassten Bereich von einfachen Funktionen bis hin zu linearen oder nichtlinearen Gleichungssystemen und Differentialgleichungen reicht.

Modernere Verfahren zur Nullstellenbestimmung sind meist Kombinationen der hier bereits erwähnten und in den folgenden Unterkapiteln detaillierter vorgestellten Verfahren.

3.2 Problemstellung

Gegeben sei eine stetige Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$. Gesucht sei ein Näherungswert für die (bzw. für eine) Nullstelle \bar{x} von f . Natürlich ist eine Gleichung der Form $g(x) = h(x)$ äquivalent zu $f(x) \equiv g(x) - h(x) = 0$. Geometrisch bedeutet das, dass $f(x)$ an der Stelle \bar{x} die x-Achse schneidet.

Aufgabe 3.1:

- Die nichtlineare Gleichung $x = \cos(x)$ lässt sich als Nullstellenproblem von $f(x) \equiv x - \cos(x) = 0$ interpretieren. Lösen Sie für $x \in [0, 1]$ auf graphischem Weg einmal die Gleichung $x = \cos(x)$ und dann die Gleichung $f(x) = x - \cos(x) = 0$.

¹John J. O'Connor, Edmund F. Robertson: Pythagoras's theorem in Babylonian mathematics. In: MacTutor History of Mathematics archive (englisch) unter <http://www-history.mcs.st-andrews.ac.uk/Indexes/HistoryTopics.html>

²Edwards, C. H. (1979). Bolzano, Cauchy, and Continuity. The Historical Development of the Calculus (pp. 308, 309). New York, NY: Springer New York.

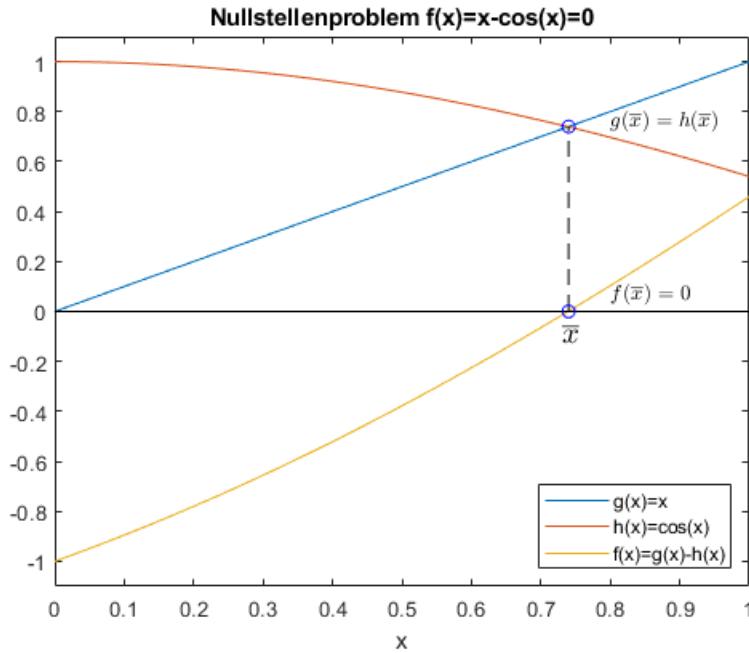


Abbildung 3.1: Graphische Lösung zu Aufgabe 3.1

Folgende Fragen sollten aber erst geklärt werden, bevor ein solches Problem gelöst werden kann:

1. Gibt es überhaupt eine Nullstelle von $f(x)$, und wenn ja, in welchem Bereich?
2. Gibt es mehrere Nullstellen? Wenn ja, welche davon sollten mit dem Rechner gefunden werden?

Wir betrachten dazu im folgenden Abschnitt die Fixpunktiteration mit dem Banachschen Fixpunktsatz.

3.3 Fixpunktiteration

Die Fixpunktiteration ist eine relativ einfache Methode zur Bestimmung von Nullstellen. Sie beruht auf der Idee, dass für nichtlineare Gleichungen der Form $f(x) = F(x) - x$ die Bedingung $f(\bar{x}) = 0$ genau dann erfüllt ist, wenn $F(\bar{x}) = \bar{x}$.

Definition 3.1: Fixpunktgleichung / Fixpunkt [1]

- Eine Gleichung der Form $F(x) = x$ heisst **Fixpunktgleichung**.
- Ihre Lösungen \bar{x} , für die $F(\bar{x}) = \bar{x}$ erfüllt ist, heissen **Fixpunkte** (da die Funktion F die Punkte \bar{x} auf sich selbst abbildet).

Anstelle eines Nullstellenproblems kann man also ein dazu äquivalentes Fixpunktproblem betrachten. Dazu muss aber $f(x) = 0$ in die Fixpunktform $F(x) = x$ gebracht werden, wozu es viele Möglichkeiten gibt. Bei dieser Überführung muss unbedingt auf Äquivalenz geachtet werden, d.h. die Lösungsmenge darf nicht verändert werden.

Beispiel 3.1:

- Die Gleichung $p(x) = x^3 - x + 0.3$ soll in Fixpunktform gebracht werden.
Lösung: Die einfachste Möglichkeit ist $p(x) = 0 \iff F(x) \equiv x^3 + 0.3 = x$
Aber auch $F(x) \equiv \sqrt[3]{x - 0.3} = x$ ist möglich.
- Die Gleichung $x = \cos(x)$, die wir weiter oben graphisch gelöst haben, ist bereits in der Fixpunktform.

Definition 3.2: Fixpunktiteration [1]

- Gegeben sei $F : [a, b] \rightarrow \mathbb{R}$, mit $x_0 \in [a, b]$. Die rekursive Folge

$$x_{n+1} \equiv F(x_n), \quad n = 0, 1, 2, \dots$$

heisst Fixpunktiteration von F zum Startwert x_0 .

Die 'Hoffnung' ist, dass die erzeugte Folge gegen einen Fixpunkt von F konvergiert. Fixpunktiterationen sind leicht durchzuführen und jeder Iterationsschritt benötigt nur eine Funktionsauswertung. Aus der generellen Form $F(x) = x$ folgt aber auch direkt, dass sich graphisch die Lösung ergibt als die Schnittpunkte zwischen den beiden Funktionen $y = F(x)$ und $y = x$. Allerdings können sich zwei Fixpunktiterationen zum gleichen Nullstellenproblem bzgl. ihrem Konvergenzverhalten unterscheiden.

Beispiel 3.2:

- Berechnen Sie Nullstellen von $p(x) = x^3 - x + 0.3$ mittels Fixpunktiteration.

Lösung: Die Fixpunktiteration lautet $x_{n+1} = F(x_n) = x_n^3 + 0.3$. Wir wissen bereits aus der letzten Aufgabe, wo wir die Nullstellen zu vermuten haben, also wählen wir Startwerte aus der Umgebung, z.B. -1, 0, 1. Wir erhalten die folgende Tabelle (aus [1]):

n	x_n	x_n	x_n
0	-1	0	1
1	-0.7	0.3	1.3
2	-0.043	0.327	2.497
3	0.299920493	0.334965783	15.86881747
4	0.3269785388	0.3375838562	3996.375585
5	0.3349588990	0.3384720217	:
6	0.3375815390	0.3387764750	:
7	0.3384712295	0.3388812067	:
8	0.3387762027	0.3389172778	:
9	0.3388811129	0.3389297064	:
10	0.3389172455	0.3389339894	:

Während mit den beiden Startwerten -1 und 0 die Fixpunktiteration gegen 0.3389... konvergiert, divergiert sie für den Startwert 1. Auch für andere Startwerte würde man feststellen, dass die Folgen entweder gegen 0.3389... konvergieren oder dann divergieren. Die Nullstelle bzw. der Fixpunkt $x = 0.3389$ scheint die Iterationsfolgen anzuziehen, die beiden anderen Nullstellen aber nicht. Daher können sie mit dieser Iteration nicht angenähert werden.

Die obere Figur in Abbildung 3.2 zeigt die Fixpunktiteration in der Nähe des Fixpunktes $x = 0.3389$. Man sieht, dass die Folge schnell konvergiert. Was führt nun dazu, dass die Folge für die beiden anderen Fixpunkte nicht konvergiert?

Die untere Figur in Abbildung 3.2 zeigt alle drei Schnittpunkte von $y = F(x)$ und $y = x$. Die Vermutung liegt nahe, dass die Steigung der Funktion $y = F(x)$ verglichen mit derjenigen von $y = x$ an der Stelle der Fixpunkte \bar{x} eine Rolle spielt. Dort, wo die Steigung von $F(x)$ kleiner ist als diejenige von $y = x$ (welche die Steigung 1 hat), scheint die Fixpunktiteration zu funktionieren, es muss also gelten $F'(\bar{x}) < 1$ (wie es in der Umgebung von \bar{x}_2 der Fall ist). Die Folge konvergiert schneller je kleiner $F'(\bar{x})$. Umgekehrt gilt, die Fixpunktiteration divergiert für $F'(\bar{x}) > 1$, wie es der Fall für die beiden anderen Fixpunkte \bar{x}_1 und \bar{x}_3 ist. Diese sind nicht mit dieser Fixpunktiteration bestimmbar.

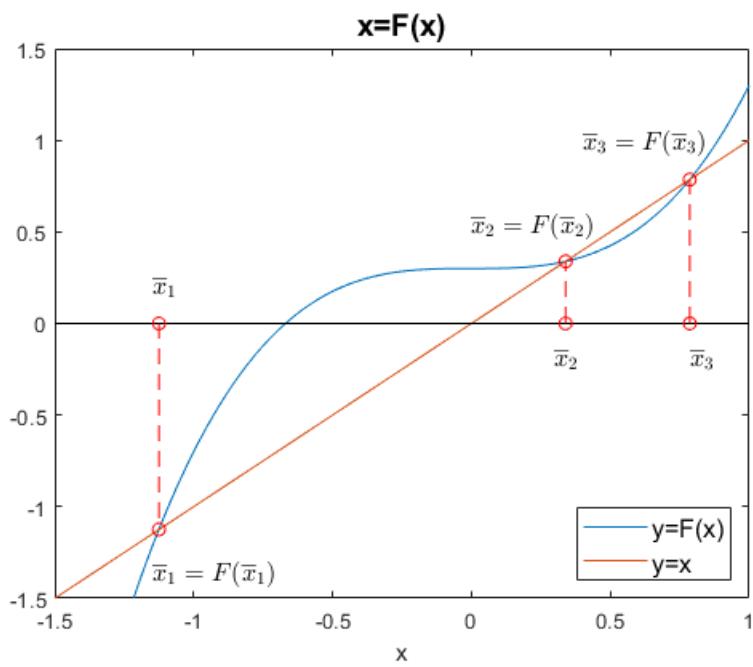
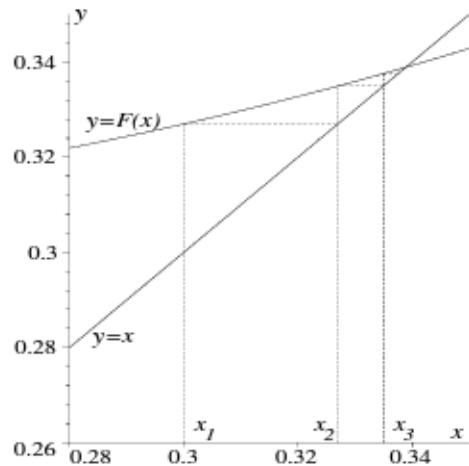


Abbildung 3.2: Oben: Fixpunktiteration zu $x_{n+1} = F(x_n) = x_n^3 + 0.3$ in der Umgebung des Fixpunktes \bar{x}_2 (aus [1]). Unten: Alle drei Fixpunkte von $F(x)$.

Wir halten also fest:

Satz 3.1 zur Fixpunktiteration [1]:

- Sei $F : [a, b] \rightarrow \mathbb{R}$ mit stetiger Ableitung F' und $\bar{x} \in [a, b]$ ein Fixpunkt von F . Dann gilt für die Fixpunktiteration $x_{n+1} = F(x_n)$:
 - Ist $|F'(\bar{x})| < 1$, so konvergiert x_n gegen \bar{x} , falls der Startwert x_0 nahe genug bei \bar{x} liegt. Der Punkt \bar{x} heisst dann **anziehender Fixpunkt**.
 - Ist $|F'(\bar{x})| > 1$, so konvergiert x_n für keinen Startwert $x_0 \neq \bar{x}$. Der Punkt \bar{x} heisst dann **abstossender Fixpunkt**.

Aufgabe 3.2 [1]:

- Überprüfen Sie anhand des obigen Satzes für das Polynom $p(x) = x^3 - x + 0.3$, welche der drei Fixpunkte $\bar{x}_1 = -1.125$, $\bar{x}_2 = 0.3389$, $\bar{x}_3 = 0.7864$ der zugehörigen Fixpunktiteration

$$x_{n+1} = F(x_n) = x_n^3 + 0.3$$

abstossend oder anziehend sind.

- Prüfen Sie, ob der Fixpunkt $\bar{x}_3 = 0.7864$ für die alternative Fixpunktiteration $x_{n+1} = F(x_n) = \sqrt[3]{x_n - 0.3}$ anziehend oder abstossend ist.
- Bestimmen Sie anhand einer Fixpunktiteration die Lösung(en) von $x = \cos(x)$.

Was uns nun interessiert ist, welche Startwerte für eine Fixpunktiteration geeignet sind und was für Fehler wir für die n -te Fixpunktiteration erwarten müssen. Dazu dient uns der

Satz 3.2: Banachscher Fixpunktsatz [1]

- Sei $F : [a, b] \rightarrow [a, b]$ (d.h. F bildet $[a, b]$ auf sich selber ab) und es existiere eine Konstante α mit $0 < \alpha < 1$ und

$$|F(x) - F(y)| \leq \alpha |x - y| \quad \text{für alle } x, y \in [a, b]$$

(d.h. F ist ‘‘Lipschitz-stetig’’ und ‘‘kontraktiv’’, α nennt man auch Lipschitz-Konstante). Dann gilt:

- F hat genau einen Fixpunkt \bar{x} in $[a, b]$
- Die Fixpunktiteration $x_{n+1} = F(x_n)$ konvergiert gegen \bar{x} für alle Startwerte $x_0 \in [a, b]$
- Es gelten die Fehlerabschätzungen

$$\begin{aligned} |x_n - \bar{x}| &\leq \frac{\alpha^n}{1 - \alpha} |x_1 - x_0| && \text{a-priori Abschätzung} \\ |x_n - \bar{x}| &\leq \frac{\alpha}{1 - \alpha} |x_n - x_{n-1}| && \text{a-posteriori Abschätzung} \end{aligned}$$

Bemerkungen:

- Aus $|F(x) - F(y)| \leq \alpha |x - y|$ für alle $x, y \in [a, b]$ folgt

$$\frac{|F(x) - F(y)|}{|x - y|} \leq \alpha,$$

wobei die linke Seite sämtliche möglichen Steigungen der Sekanten durch die beiden Punkte $(x, F(x))$ und $(y, F(y))$ für alle $x, y \in [a, b]$ darstellt. Aus diesem Grund kann man α als die grösstmögliche Steigung von $F(x)$ auf dem Intervall $[a, b]$ interpretieren, bzw.

$$\alpha = \max_{x_0 \in [a, b]} |F'(x_0)|$$

- Wählt man das Intervall $[a, b]$ sehr nahe um einen anziehenden Fixpunkt \bar{x} , so ist also $\alpha \approx |F'(\bar{x})|$.
- In der Praxis gestaltet es sich meist schwierig, ein Intervall $[a, b]$ zu finden, dass unter F auf sich selbst abgebildet wird. Hat man ein solches Intervall gefunden, dann sind die Fehlerabschätzungen aber recht nützlich. Wir werden diesen Satz nochmals im Zusammenhang mit der iterativen Lösung von linearen Gleichungssystemen in Kap. 4 aufgreifen.

Beispiel 3.3:

- Gesucht ist ein Intervall $[a, b]$ und eine Konstante $\alpha < 1$, so dass der Banachsche Fixpunktsatz auf die Fixpunktiteration $x_{n+1} = F(x_n) = x_n^3 + 0.3$ anwendbar ist.

Lösung: Wir wissen bereits, dass die Fixpunktiteration in der Nähe von $\bar{x} = 0.3389$ konvergiert. Also suchen wir in der Nähe davon ein geeignetes Intervall. Wir versuchen es zum Beispiel mit $[a, b] = [0, 0.5]$. Für jedes x in diesem Intervall gilt $F(x) = x^3 + 0.3 \geq 0.3$ und der maximale Funktionswert ist $F(0.5) = 0.125 + 0.3 = 0.425 \leq 0.5$. Also bildet F das Intervall $[0, 0.5]$ tatsächlich auf $[0, 0.5]$ ab, die erste Bedingung ist also erfüllt. Jetzt untersuchen wir, ob es eine Konstante $\alpha < 1$ gibt, so dass $|F(x) - F(y)| \leq \alpha |x - y|$ für alle $x, y \in [0, 0.5]$ gilt. Aus der obigen Bemerkung wissen wir dass

$$\alpha = \max_{x_0 \in [a, b]} |F'(x_0)|$$

Also berechnen wir die Ableitung $F'(x)$ auf dem Intervall $[0, 0.5]$ und finden, dass der maximale Wert der Ableitung $|F'(x)| = 3x^2$ wegen ihrem monoton steigenden Verhalten bei $x = 0.5$ erreicht wird und dass $|F'(x)| = 3x^2 = 3 * 0.5^2 = 0.75 < 1$. Also setzen wir $\alpha = 0.75$.

Aufgabe 3.3 [1]:

1. Schätzen sie jetzt für das obige Beispiel mit der a-priori Abschätzung ab, wie viele Iterationen ausreichen sollten, um ausgehend von $x_0 = 0$ einen absoluten Fehler von max. 10^{-4} zu erhalten. Wenden Sie dann die a-posteriori Abschätzung an, um den absoluten Fehler zu erhalten.
2. Finden Sie mit Hilfe des Banachschen Fixpunktsatzes den Fixpunkt \bar{x}_2 für die Fixpunktiteration $x_{n+1} = F(x_n) = \sqrt[3]{x_n - 0.3}$ und den Startwert $x_0 = 0.7$.
3. Welche der beiden Fixpunktiterationen $x_{n+1} = F(x_n) = x_n^3 + 0.3$, $x_0 = 0$ und $x_{n+1} = F(x_n) = \sqrt[3]{x_n - 0.3}$, $x_0 = 0.7$ wird nach Ihrer Erwartung schneller konvergieren?

3.4 Das Newton-Verfahren

In diesem Abschnitt werden wir ein weiteres Verfahren zur Lösung nichtlinearer Gleichungssysteme betrachten, das bereits aus der Analysis bekannte Newton-Verfahren. Im Vergleich zu den bisher betrachteten Verfahren konvergiert dieses meist deutlich schneller. Wie wir im nächsten Abschnitt sehen werden, ist es quadratisch konvergent. Im Gegensatz zum Bisektions-Verfahren oder der Fixpunktiteration wird hier nicht allerdings nur die Funktion f selbst sondern auch ihre Ableitung benötigt. Wir setzen also voraus, dass f stetig differenzierbar ist.

Die Idee des Newton-Verfahrens ist wie folgt: Berechne die Tangente $g(x)$ von f im Punkt x_n , d.h. die Gerade

$$g(x) = f(x_n) + f'(x_n)(x - x_n).$$

Die Nullstelle von g sei x_{n+1} , dann gilt also

$$g(x_{n+1}) = 0 = f(x_n) + f'(x_n)(x_{n+1} - x_n).$$

Auflösen nach x_{n+1} liefert

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (n = 0, 1, 2, 3, \dots).$$

Das gilt natürlich nur, wenn $f'(x_n) \neq 0$ erfüllt ist. Die Idee ist in Abb.3.3 graphisch dargestellt. Den Startwert sollte man in der Nähe der Nullstelle wählen, um eine schnelle Konvergenz zu erreichen. Die Konvergenz der Folge

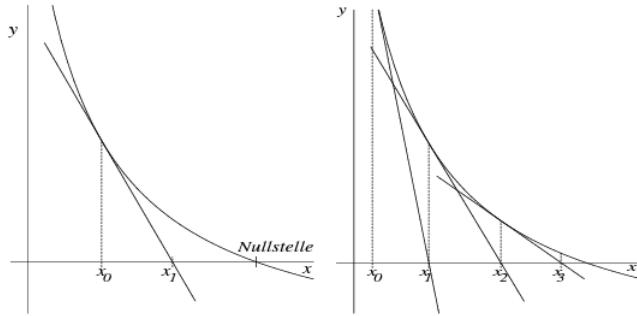


Abbildung 3.3: Newton-Verfahren (aus [1]).

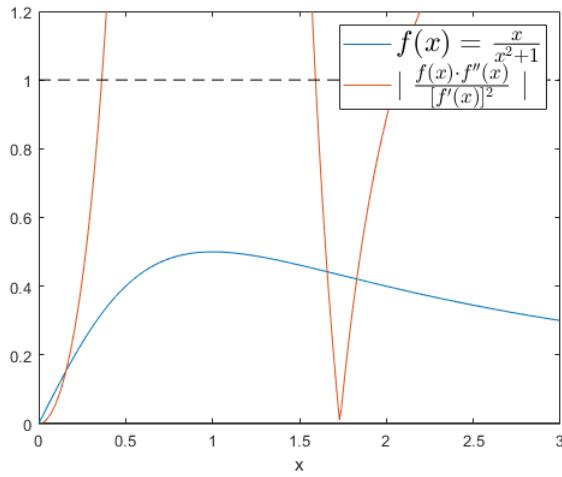


Abbildung 3.4: Für die dargestellte Funktion $f(x) = x/(x^2 + 1)$ konvergiert das Newton-Verfahren für Startwerte $x_0 \geq 1$ nicht.

(x_0, x_1, x_2, \dots) ist sicher gegeben, wenn im Intervall $[a, b]$, in dem alle Näherungswerte (und die Nullstelle selbst) liegen sollen, die Bedingung

$$\left| \frac{f(x) \cdot f''(x)}{[f'(x)]^2} \right| < 1$$

erfüllt ist (hinreichende Konvergenzbedingung). In der Regel überprüft man diese Bedingung zumindest für den Startwert x_0 . Ungeeignet sind Startwerte, in deren unmittelbarer Umgebung die Kurventangente nahezu parallel zur x -Achse verläuft.

Betrachten wir z.B. die Funktion

$$f(x) = \frac{x}{x^2 + 1}$$

in Abb. 3.4. Für Startwerte $x_0 < 1$ konvergiert das Newton-Verfahren gegen die Nullstelle $\bar{x} = 0$, für $x_0 \geq 1$ divergiert es. Nur im Intervall $[0, 0.3625]$ ist die hinreichende Konvergenzbedingung für alle x_i erfüllt.

Aufgabe 3.4 [1]:

- Bestimmen Sie die Nullstellen von $f(x) = x^2 - 2 = 0$ näherungsweise mit dem Newton-Verfahren und dem Startwert $x_0 = 2$. Vergleichen Sie ihren Wert nach $n+1 = 4$ Iterationsschritten mit dem exakten Wert von $\sqrt{2}$. Auf wie vielen Nachkommastellen stimmt die Iterationslösung überein? Für welchen Startwert konvergiert die Folge nicht.?
- Bestimmen Sie das Iterationsverfahren für $f(x) = x^2 - a = 0$ als Berechnungsmöglichkeit für \sqrt{a} und vergleichen Sie das Resultat mit dem in Kap. 3.1 vorgestellten Heron-Verfahren.

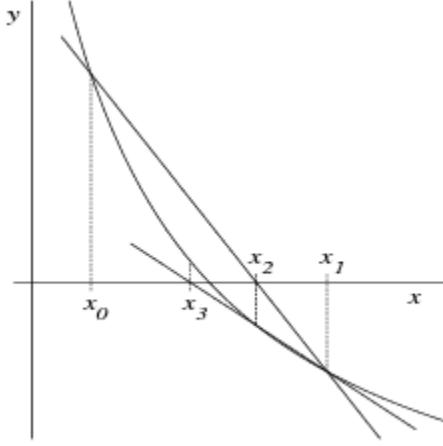


Abbildung 3.5: Sekantenverfahren (aus [1]).

Das Newton-Verfahren ist ein häufig verwendetes und schnelles Verfahren, um Nullstellen zu bestimmen. Es hat aber den Nachteil, dass man in jedem Schritt wieder eine Ableitung berechnen muss. Um das zu umgehen, kann man zu zwei vereinfachten Verfahren greifen, dem vereinfachten Newton-Verfahren und dem Sekantenverfahren.

3.4.1 Vereinfachtes Newton-Verfahren

Statt in jedem Schritt $f'(x_n)$ auszurechnen, kann man immer wieder $f'(x_0)$ verwenden. Damit ergibt sich die Rekursionsformel:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)} \quad (n = 0, 1, 2, 3, \dots).$$

Natürlich wird man erwarten, dass dieses Verfahren weniger gut funktioniert als das originale Newton-Verfahren. Tatsächlich konvergiert es langsamer.

3.4.2 Sekantenverfahren

Hier wird nicht der Schnittpunkt der Tangenten mit der x -Achse berechnet, sondern der Schnittpunkt von Sekanten ('Schneidenden') durch jeweils zwei Punkte $(x_0, f(x_0))$ und $(x_1, f(x_1))$ mit der x -Achse. Statt der Ableitung $f'(x_0)$ wird in der Iterationsformel dann die Steigung

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

der Sekanten eingesetzt und man erhält

$$x_2 = x_1 - \frac{f(x_1)}{\frac{f(x_1) - f(x_0)}{x_1 - x_0}} = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} \cdot f(x_1)$$

und analog die Iterationsformel

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \cdot f(x_n) \quad (n = 1, 2, 3, \dots).$$

Das Sekantenverfahren ist veranschaulicht in Abbildung 3.5. Es benötigt zwei Startwerte x_0 , x_1 und konvergiert langsamer, dafür benötigt es keine Ableitungen.

3.5 Konvergenzgeschwindigkeit

Wie wir bereits angesprochen haben, unterscheiden sich die Nullstellenverfahren in ihrer Konvergenzgeschwindigkeit. Diese lässt sich durch den Begriff der Konvergenzordnung miteinander vergleichen.

Definition 3.3: Konvergenzordnung [1]

- Sei (x_n) eine gegen \bar{x} konvergierende Folge. Dann hat das Verfahren die **Konvergenzordnung** $q \geq 1$ wenn es eine Konstante $c > 0$ gibt mit

$$|x_{n+1} - \bar{x}| \leq c \cdot |x_n - \bar{x}|^q$$

für alle n . Falls $q = 1$ verlangt man noch $c < 1$. Im Fall $q = 1$ spricht man von linearer, im Fall $q = 2$ von quadratischer Konvergenz.

Beispiel 3.4:

- Sei $c = 1$ und $|x_0 - \bar{x}| \leq 0.1$. Es gilt dann also z.B. für quadratische Konvergenz nach jeder Iteration, dass der Fehler quadratisch abnimmt:

$$\begin{aligned} |x_1 - \bar{x}| &\leq |x_0 - \bar{x}|^2 \leq 0.1^2 = 10^{-2} \\ |x_2 - \bar{x}| &\leq |x_1 - \bar{x}|^2 \leq (10^{-2})^2 = 10^{-4} \\ |x_3 - \bar{x}| &\leq |x_2 - \bar{x}|^2 \leq (10^{-4})^2 = 10^{-8} \\ &\vdots \end{aligned}$$

Bemerkungen:

- Es gilt: für einfache Nullstellen konvergiert das Newton-Verfahren quadratisch, das vereinfachte Newton-Verfahren linear, und für das Sekantenverfahren gilt $q = (1 + \sqrt{5})/2 = 1.618\dots$

3.6 Fehlerabschätzung

Wir haben beim Banachschen Fixpunktsatz (Kap. 3.3) bereits eine Art der Fehlerabschätzung kennengelernt, benötigen dort aber die Konstante α . In der Praxis gibt es einfachere Methoden, um abzuschätzen, wie weit eine Näherung x_n nach der n -ten Iteration von der exakten Nullstelle entfernt ist.

Zur Lösung dient der folgende Satz aus der Analysis:

Satz 3.3: Nullstellensatz von Bolzano

- Sei $f : [a, b] \rightarrow \mathbb{R}$ stetig mit $f(a) \leq 0 \leq f(b)$ oder $f(a) \geq 0 \geq f(b)$. Dann muss f in $[a, b]$ eine Nullstelle besitzen.

Wenn man also auf dem Intervall $[a, b]$ einen Vorzeichenwechsel von f feststellt, d.h.

$$f(a) \cdot f(b) < 0,$$

dann besitzt f in diesem Intervall mindestens eine Nullstelle. Im folgenden beschreiben wir ein Verfahren, dass diesen Umstand benutzt.

Eine einfache Möglichkeit ist es also, die Funktion in der Umgebung der Näherung auszuwerten und zu überprüfen, ob ein Vorzeichenwechsel stattfindet. Daraus lässt sich gemäß dem Nullstellensatz schliessen, dass eine Nullstelle innerhalb des betrachteten Intervalls liegen muss und man kann abschätzen, wie weit die Näherung x_n von der tatsächlichen Nullstelle entfernt ist. Dieses Verfahren ist auf jedes iterative Verfahren zur Nullstellenbestimmung einer Funktion anwendbar, sofern die Nullstelle ungerade Ordnung hat (d.h. sie ist ein Schnittpunkt und nicht ein Berührungs punkt des Funktionsgraphen mit der x -Achse).

Sei x_n also ein iterativ bestimmter Näherungswert einer exakten Nullstelle ξ (ungerader Ordnung) der stetigen Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ und es gelte für eine vorgegebene Fehlerschranke / Fehlertoleranz $\epsilon > 0$

$$f(x_n - \epsilon) \cdot f(x_n + \epsilon) < 0,$$

dann muss gemäß dem Nullstellensatz im offenen Intervall $(x_n - \epsilon, x_n + \epsilon)$ eine Nullstelle ξ liegen und es gilt die Fehlerabschätzung (vgl. Abb. 3.6)

$$|x_n - \xi| < \epsilon$$

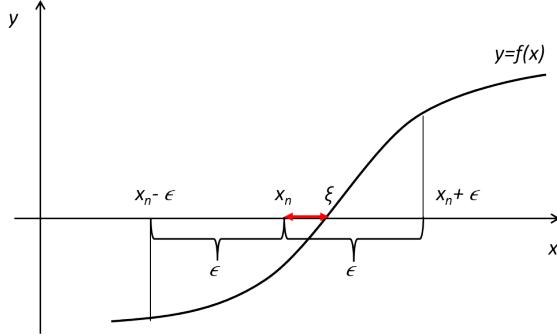


Abbildung 3.6: $f(x_n - \epsilon) \cdot f(x_n + \epsilon) < 0 \Rightarrow |x_n - \xi| < \epsilon$ (nach [6]).

Beispiel 3.5 [1]:

- Es soll für $f(x) = x^2 - 2 = 0$ der Fehler für die Näherung x_3 der Nullstelle mit dem Newton-Verfahren berechnet werden.

Lösung: Es ist leicht zu sehen dass $f(x_3 - 10^{-5}) < 0$ und $f(x_3 + 10^{-5}) > 0$. Gemäss dem Nullstellensatz gibt es also eine Nullstelle $x \in [x_3 - 10^{-5}, x_3 + 10^{-5}]$ für die der absolute Fehler $|x - x_3| \leq 10^{-5}$ ist. Tatsächlich gilt $|\sqrt{2} - x_3| \approx 2.1 \cdot 10^{-6}$

Um auch den Fall möglicher Berührungs punkten mit der x -Achse oder schlecht konditionierte Probleme abzudecken, empfiehlt es sich sich, in einem Programm zusätzliche Abbruchkriterien einzubauen, da ansonsten die Iteration vielleicht in eine Endlos-Schleife mündet. Einfachstes Mittel, ist eine Obergrenze N_{max} für die Anzahl Iterationsschritte anzugeben. Notwendige (aber nicht hinreichende) Kriterien, um eine Nullstelle zu erkennen, sind für ein vorgegebenes $\epsilon > 0$ beispielsweise, dass der Funktionswert nach der n -ten Iteration kleiner wird als ϵ , also $|f(x_n)| < \epsilon$, oder auch, dass die Differenz zwischen zwei aufeinanderfolgenden Werten unterhalb einer vorgegebenen Schwelle sinkt, also $|x_{n+1} - x_n| < \epsilon$. Diese Abbruchkriterien liefern aber keine Garantie, dass wir tatsächlich nahe genug bei einer Nullstelle sind.

Aufgaben 3.5 [1]:

- A Bestimmen Sie alle Lösungen der Gleichung $2 \sin x = x$ bis auf einen nachgewiesenen absoluten Fehler von max. 10^{-3} .
- B Das Bauer-Ziege-Wiese-Problem: Ein Bauer besitzt eine kreisrunde Wiese vom Radius R . Am Rand dieser Wiese bindet er eine Ziege an mit einer Leine der Länge r , und zwar so, dass die Ziege genau die Hälfte der Wiese abgrasen kann (s. Bild 2.4). Wie groß ist r ?

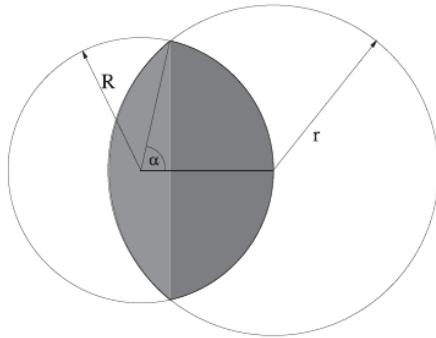


Bild 2.5

Mit dem Kosinussatz erhält man $r = R \sqrt{2(1 - \cos \alpha)}$. Das Problem führt auf folgende Gleichung für den Winkel α (im Bogenmaß):

$$\frac{\pi}{2 \cos \alpha} + \alpha - \pi - \tan \alpha = 0.$$

Offensichtlich kann diese Gleichung nicht durch geschicktes Umformen nach α aufgelöst werden. Die Hilfe numerischer Methoden ist daher nötig. Bestimmen Sie ein Intervall, in dem sich die gesuchte Lösung befindet und bestimmen Sie die Lösung mit einem Verfahren Ihrer Wahl bis auf einen gesicherten absoluten Fehler von 0.0001.

- C Wenden Sie das Newton-Verfahren, das vereinfachte Newton-Verfahren und das Sekantenverfahren zur näherungsweisen Bestimmung der Nullstelle von $f(x) = x^2 - 2$ an.

Aufgabe 3.6:

- Optional: Implementieren Sie das Sekanten-Verfahren in Python. Wo würden Sie beim Versuch, das Newton-Verfahren zu implementieren, momentan noch auf Schwierigkeiten stossen?

Kapitel 4

Numerische Lösung linearer Gleichungssysteme

In diesem Kapitel behandeln wir die Lösung linearer Gleichungssysteme, die in vielen Anwendungen in der Numerik, Physik, Technik, Betriebswirtschaftslehre etc. auftreten. Beispiele sind das Newton-Verfahren für *nichtlineare* Gleichungssysteme, wo bei jedem Schritt lineare Gleichungssysteme auftreten; die Methode der kleinsten Quadrate von Gauss in der Ausgleichsrechnung; die numerische Lösung von Randwertproblemen bei gewöhnlichen und partiellen Differentialgleichungen mit Hilfe von Differenzenverfahren; bei der Interpolation mittels Splines; die Behandlung von Eigenwertproblemen in der mathematischen Physik; in der Elektrotechnik die Berechnung von Netzwerken (Ströme zu vorgegebenen Spannungen und Widerständen); in der Betriebswirtschaftslehre bei der linearen Programmierung uvm. In der Theorie ist das Problem der Auflösung linearer Gleichungssysteme vollständig gelöst, in der Praxis geht es um deren effiziente Berechnung.

Lernziele:

- Sie können lineare Gleichungssysteme selbst aufstellen.
- Sie können den Gauss-Algorithmus mit und ohne Pivotisierung sowie die *LR*-Zerlegung auf konkrete Problemstellungen anwenden.
- Sie kennen die *QR*-Zerlegung und können Sie anwenden.
- Sie können die Fehler für gestörte lineare Gleichungssysteme berechnen.
- Sie können das Jacobi- sowie das Gauss-Seidel-Verfahren anwenden und in Python implementieren.
- Sie beherrschen die zugehörigen Fehlerabschätzungen.
- Sie können Eigenwerte und Eigenvektoren von Matrizen berechnen.

4.1 Zur historischen Entwicklung

Auch lineare Gleichungssystem beschäftigten Mathematiker schon vor Tausenden von Jahren. Eine Aufgabe, die rund 4000 Jahre alt ist und aus Mesopotamien stammt, lautet: "Ein Viertel der Breite zur Länge addiert ergibt 7 Handbreiten, Länge und Breite addiert macht 10 Handbreiten". Natürlich beschäftigten sich auch die Ägypter mit ähnlichen Problemen, wie z.B. der folgenden Aufgabe aus dem 'Papyrus Moskau'¹ ca. 2000 v.Chr.: "Berechne die Länge und Breite eines Rechteckes der Fläche 12, wenn die Breite $3/4$ der Länge ist". In der heutigen Schreibweise würden wir das erste Beispiel als System zweier Gleichungen mit zwei Unbekannten formulieren (mit x als Breite

¹welcher in Moskau aufbewahrt wird, daher der Name

und y als Länge²):

$$\begin{aligned}\frac{1}{4}x + y &= 7 \\ x + y &= 10\end{aligned}$$

bzw. im Matrizenkalkül

$$\begin{pmatrix} \frac{1}{4} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 7 \\ 10 \end{pmatrix}$$

Die Babylonier oder die Ägypter kannten kein Matrizenkalkül. Die Chinesen kamen dem zwischen 200 bis 100 v.Chr. schon bedeutend näher, wie im chinesischen Mathematikbuch Jiu Zhang Suanshu (dt. 'Neun Kapitel der Rechenkunst' od. 'Neun Bücher arithmetischer Technik') aus dieser Zeit festgehalten ist, welches die chinesische Mathematik und diejenige der umliegenden Länder bis ins 17. Jhr. prägte. So wurde darin bereits das Verfahren beschrieben, welches wir heute als Gauss-Algorithmus kennen³.

Die erste systematische Untersuchung von linearen Gleichungssystemen wird Gottfried Wilhelm Leibniz (1646-1716) zugeschrieben⁴. Er führte die Formeln für Determinanten für 2×2 und 3×3 Gleichungssysteme ein. Gabriel Cramer (1704-1752) entwickelte die nach ihm benannte allgemeine Lösungsformel für Systeme von n Gleichungen mit n Unbekannten. Seine Regel benötigt allerdings einen enormen Rechenaufwand von rund $n(n+1)!$ Gleitkommaoperationen. Für $n = 10$ benötigt man bereits fast 400 Mio. Punktoperationen und für $n = 20$ bereits 10^{21} . Deshalb ist die Cramersche Regel in der Praxis völlig unbrauchbar (dies gilt bereits für $n = 3$).

Der deutsche Mathematiker, Physiker, Astronom und Geodät Carl Friedrich Gauss (1777-1855) betrachtete lineare Gleichungssysteme im Zusammenhang mit astronomischen Problemen. So gelang es ihm, den Zwergplaneten Ceres im Asteroidengürtel zwischen Mars und Jupiter, der 1801 entdeckt und gleich darauf wieder verloren gegangen war, aufgrund seiner Berechnungen basierend auf der Methode der kleinsten Quadrate wieder zu finden. 1811 entwickelte er den nach ihm benannten Gauss-Algorithmus (vgl. Kap. 4.3), eines der heutigen Standardverfahren zur Lösung von linearen Gleichungssystemen. Der Gauss-Algorithmus benötigt für die Lösung eines $n \times n$ Gleichungssystems lediglich $\frac{2}{3}n^3 + \frac{5}{2}n^2 - \frac{13}{6}n$ Punktoperationen (vgl. Kap. 4.6.2), d.h. für $n = 20$ also nur rund 6000 im Gegensatz zu 10^{21} bei der Cramerschen Regel.

Ausgehend von den Untersuchungen linearer Gleichungssysteme entwickelte sich daraus das Gebiet der linearen Algebra, unter anderem basierend auf den Werken von William Rowan Hamilton (1805-1865; Vektoren, Quaternionen), Hermann Grassmann (1809-1877; endlichdimensionale Vektorräume), Arthur Cayley (1821-1895; Matrizen als algebraische Objekte), Camille Jordan (1838-1922; Jordansche Normalform), Ferdinand Georg Frobenius (1849-1917; Gruppentheorie), Maxime Bôcher (1867-1918; *Introduction to higher algebra*), Herbert Westren Trumbull (1885-1961) und Alexander Aitken (1895-1967) mit *Introduction to the Theory of Canonical Matrices* sowie Leon Mirsky (1918-1983) mit *An introduction to linear algebra*.

Beispiele aus der Praxis mit grossen linearen Gleichungssystemen [6]

- Parabolantenne der Firma Krupp mit 100 m Durchmesser am oberen Rand (vgl. Abb. 4.1^{5,6}).

Es handelt sich dabei um einen räumlichen Verbund aus Stäben und Balken, die geometrisch ein Rotationsparaboloid bilden. Die Berechnung muss so erfolgen, dass bei Verformung durch Neigung und Eigengewicht wegen der Richtgenauigkeit der Antenne immer wieder ein Rotationsparaboloid entsteht. Es sind jeweils ca. 5000 Gleichungen mit 5000 Unbekannten zu lösen. Nur der Empfänger muss dann jeweils in den neuen Brennpunkt nachgeführt werden. Für jede neue Einstellung beträgt die mittlere Abweichung vom idealen Paraboloid weniger als 0.6 mm (2012).

²Die Bezeichnung unbekannter Größen durch Buchstaben x, y, z stammt übrigens vom französischen Mathematiker und Philosoph René Descartes (1596-1650)

³siehe z.B. MacTutor unter http://www-history.mcs.st-and.ac.uk/HistTopics/Matrices_and_determinants.html

⁴siehe <http://www.math.kit.edu/iag2/~globke/media/geschichtela.pdf>

⁵By Dr. Schorsch (photo taken by Dr. Schorsch) |GFDL (<http://www.gnu.org/copyleft/fdl.html>), CC-BY-SA-3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>) or CC-BY-SA-3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>], via Wikimedia Commons

⁶„Radiotelescope effelsberg full“ von Hotstepper13 - Eigenes Werk. Lizenziert unter Creative Commons Attribution 3.0 über Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Radiotelescope_effelsberg_full.jpg#mediaviewer/File:Radiotelescope_effelsberg_full.jpg

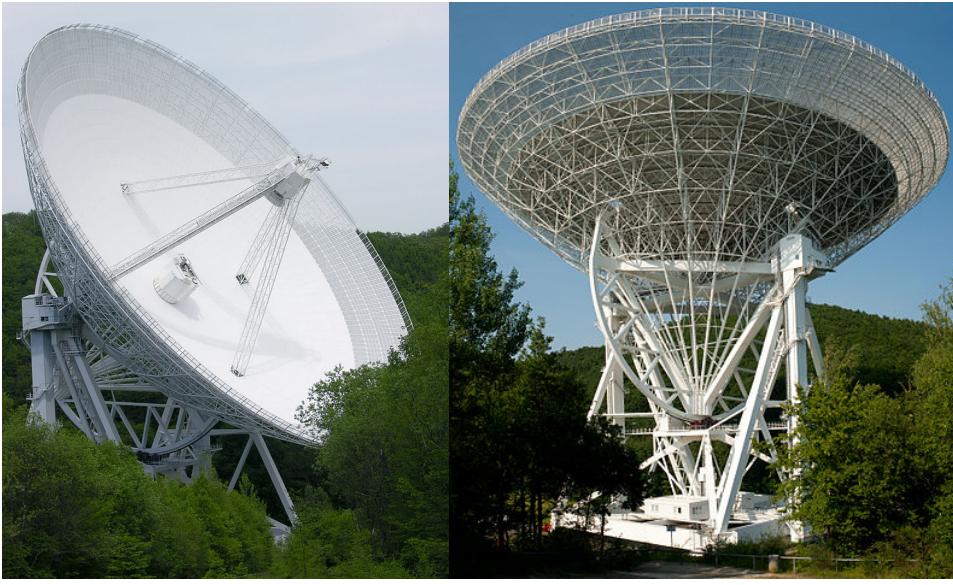


Abbildung 4.1: 100 m Radioteleskop in der Eiffel (Wikimedia Commons).

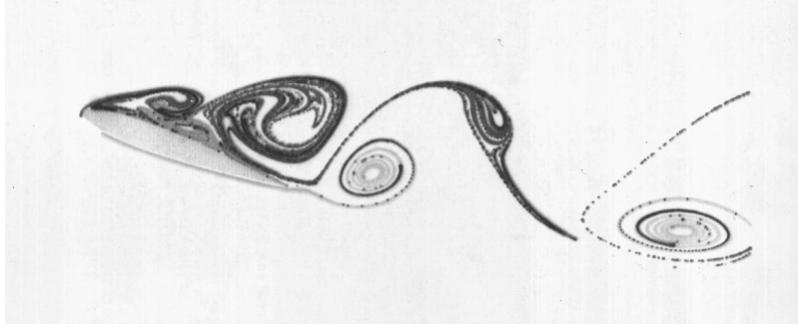


Abbildung 4.2: Simulaton einer ablösenden Strömung [6].

2. Beispiel des Aerodynamischen Instituts der RWTH Aachen. Numerische Simulation einer ablösenden Strömung um Tragflügelprofile, gerechnet mit den Navier-Stokes- Gleichungen: Wenn 3-dimensional gerechnet wird und ein $(31 \times 31 \times 51)$ -Gitter mit je 4 Gleichungen verwendet wird, so erhält man nichtlineare Systeme aus 196 044 Gleichungen mit 196 044 Unbekannten, die iterativ (etwa mit 5 Iterationen) gelöst werden. Rechnet man bis zum Wirbelablösen 10 000 Zeitschritte, so ergeben sich $5 \times 10 000 = 50 000$ lineare Gleichungssysteme aus rund ca. 200 000 Gleichungen, die zu lösen sind.
3. Ein Finite-Element-Beispiel aus dem Institut für Bildsame Formgebung der RWTH Aachen: Bei der Simulation des Fließpress-Verfahrens zur Herstellung eines Zahnrades mit zwölf Zähnen wird unter Ausnutzung der Symmetriebedingungen mit dem Modell eines halben Zahnes gerechnet. In diesem Beispiel wird dazu ein Netz mit 2911 Knoten erstellt. Man erhält unter Berücksichtigung aller Randbedingungen insgesamt 7560 nichtlineare Gleichungen, die iterativ gelöst werden. Dabei tritt eine Bandmatrix auf.
4. Der PageRank-Algorithmus, welcher auf die Gründer von Google, Sergey Brin und Lawrence Page, zurückgeht⁷, erlaubt die Klassifizierung einer Menge von verlinkten Dokumenten, z.B. der Seiten des Internets, nach ihrer "Wichtigkeit", bzw. dem *rank*. Die zugrunde liegende Idee⁸ ist, dass eine Seite umso wichtiger ist, je mehr Links von anderen wichtigen Seiten auf sie zeigen. Zur Bestimmung der Wichtigkeit wird die PageRank- bzw. Google-Matrix benötigt. Diese Matrix repräsentiert einen gerichteten Graphen, wobei die Knoten des Graphen den Web-Seiten entsprechen und die Kanten den Links dazwischen.

⁷Sergei Brin, Lawrence Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Computer Networks and ISDN Systems, Band 30, 1998, S. 107-117

⁸nach UZH, MAS410, Geometrie und Lineare Algebra

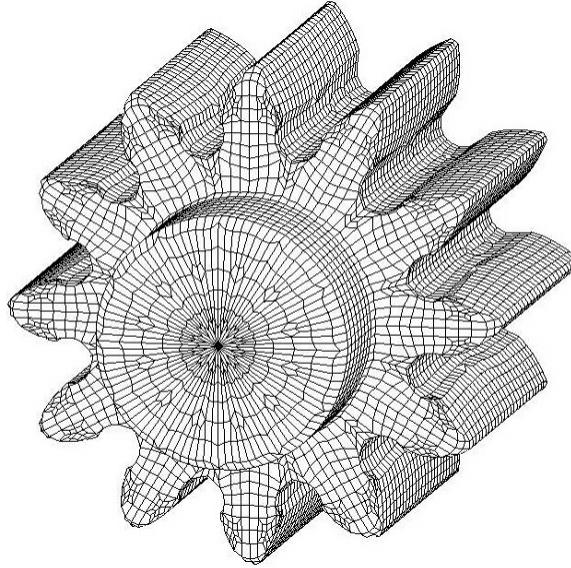


Abbildung 4.3: Fliesspressverfahren zur Herstellung eines Zahnrades [6].

Beispiel: ein einfaches Web mit 4 Seiten ist in Abb. 4.4 dargestellt. Ein Pfeil von Seite i zur Seite j entspricht einem Link. Die Bedeutung der Web-Seiten wird durch den Vektor

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

angegeben, wobei $x_i \in \mathbb{R}$ die Wichtigkeit der Seite i angibt. Für die Seite 1 ergibt sich z.B.

$$x_1 = 0 \cdot x_1 + 0 \cdot x_2 + 1 \cdot x_3 + \frac{1}{2} \cdot x_4.$$

Die Gewichte berechnen sich daraus, dass die Seite 1 keinen Link auf sich selber ($0 \cdot x_1$) oder von Seite 2 ($0 \cdot x_2$) hat, jedoch je einen Link von Seite 3 und Seite 4. Da Seite 3 insgesamt nur einen ausgehenden Link aufweist, erhält dieser für Seite 1 das volle Gewicht ($1 \cdot x_3$). Da Seite 4 aber 2 ausgehende Links aufweist, erhält der Link auf Seite 1 nur das Gewicht $\frac{1}{2}$ (also $\frac{1}{2} \cdot x_4$). Für alle vier Seiten erhält man so das lineare Gleichungssystem:

$$\begin{aligned} x_1 &= 0x_1 + 0x_2 + 1x_3 + \frac{1}{2}x_4 \\ x_2 &= \frac{1}{3}x_1 + 0x_2 + 0x_3 + 0x_4 \\ x_3 &= \frac{1}{3}x_1 + \frac{1}{2}x_2 + 0x_3 + \frac{1}{2}x_4 \\ x_4 &= \frac{1}{3}x_1 + \frac{1}{2}x_2 + 0x_3 + 0x_4 \end{aligned}$$

Oder in Matrix-Schreibweise:

$$\mathbf{x} = \mathbf{P} \cdot \mathbf{x}, \quad \text{mit } \mathbf{P} = \begin{pmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

Also ist \mathbf{x} ein Eigenvektor von \mathbf{P} zum Eigenwert 1. Dies ist zudem eine Fixpunktgleichung und kann gemäss Kap. 3.4 iterativ gelöst werden. Mit dem Startvektor $\mathbf{x}_0 = (1, 1, 1, 1)^T$ erhalten wir mittels der Fixpunktiteration

$$\mathbf{x}_{i+1} = \mathbf{P} \mathbf{x}_i$$

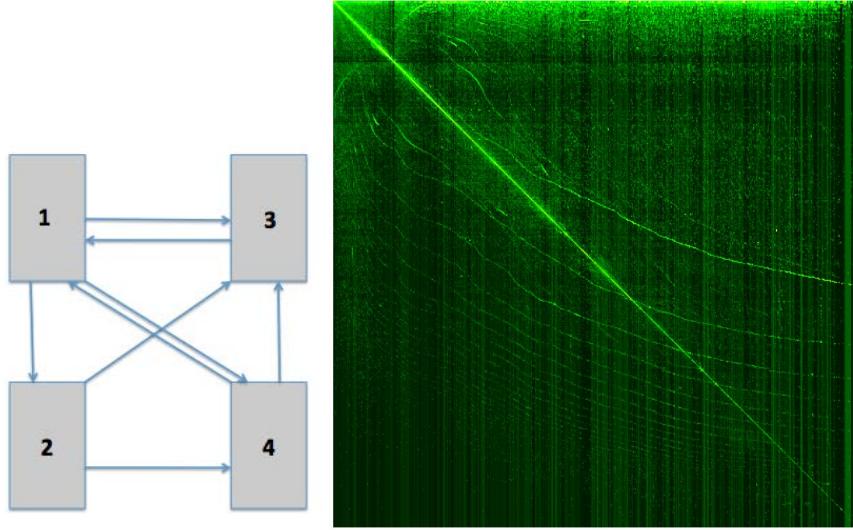


Abbildung 4.4: Links: Einfaches Web mit $n = 4$ Seiten und seinen Links⁸. Rechts: Google Matrix des Netzwerks der Cambridge Universität aus dem Jahr 2006 mit $n = 212710$ (<http://arxiv.org/abs/1106.6215>, GFDL, Wikimedia Commons)

die Näherungslösung

$$\mathbf{x}_1 = \mathbf{P}\mathbf{x}_0 = \begin{pmatrix} 1.5000 \\ 0.3333 \\ 1.3333 \\ 0.8333 \end{pmatrix}, \quad \mathbf{x}_2 = \mathbf{P}\mathbf{x}_1 = \begin{pmatrix} 1.75 \\ 0.5 \\ 1.0833 \\ 0.6667 \end{pmatrix}, \dots, \mathbf{x}_{16} = \mathbf{P}\mathbf{x}_{15} = \begin{pmatrix} 1.5484 \\ 0.5161 \\ 1.1613 \\ 0.7742 \end{pmatrix}$$

Also hat die Seite 1 die höchste Wichtigkeit, Seite 3 die zweithöchste, Seite 4 die dritthöchste, und Seite 2 die vierthöchste bzw. die niedrigste Wichtigkeit. Unter Verwendung der Einheitsmatrix \mathbf{I} lässt sich \mathbf{x} auch mit dem aus der linearen Algebra bereits bekannten und in Kap. 4.3 nochmals detailliert beschriebenen Gauss-Algorithmus als eine Lösung der homogenen Gleichung

$$(\mathbf{P} - \mathbf{I})\mathbf{x} = \mathbf{0}$$

bestimmen.

Um auch zufälliges Hüpfen zwischen den Seiten (ohne Benützung von Links) abbilden zu können, wird die Matrix \mathbf{P} noch modifiziert mit einer Matrix \mathbf{S} , deren Elemente alle den Wert $\frac{1}{n}$ haben bei einem Web mit n Seiten. Die **Google-Matrix** \mathbf{G} erhält man als Überlagerung der beiden Matrizen:

$$\mathbf{G} = \alpha\mathbf{P} + (1 - \alpha)\mathbf{S}.$$

Dabei ist $0 \leq \alpha \leq 1$ ein Faktor, der das zufällige Hüpfen modelliert (für $\alpha = 1$ findet kein zufälliges Hüpfen statt, für $\alpha = 0$ findet ausschließlich zufälliges Hüpfen statt). Die Erfinder des PageRank-Algorithmus wählten $\alpha = 0.85$.

4.2 Problemstellung

Gesucht ist eine Lösung zu einem linearen Gleichungssystem mit n Gleichungen und n Unbekannten der Form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n &= b_n \end{aligned} \tag{4.1}$$

Üblicherweise schreibt man solche Gleichungssysteme in Matrix-Form als

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.2}$$

mit

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \in \mathbb{R}^{n \times n}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n \text{ und } \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \in \mathbb{R}^n.$$

Bezüglich der Notation werden in diesem Skript Matrizen mit fettgedruckten Grossbuchstaben und Vektoren mit fettgedruckten Kleinbuchstaben hervorgehoben. Im obigen Gleichungssystem sind \mathbf{A} und \mathbf{b} gegeben, \mathbf{x} ist gesucht. Gewisse Eigenschaften der Matrix $\mathbf{A} = (a_{ij})$ entscheiden darüber, was für ein Verfahren zur Lösung des Gleichungssystems (4.2) sinnvoll eingesetzt werden kann. Da die Anzahl n der Gleichungen in (4.1) der Anzahl Unbekannten x_1, \dots, x_n entspricht, ist \mathbf{A} in (4.2) eine quadratische Matrix der Dimension $n \times n$.

Für quadratische Matrizen \mathbf{A} wissen wir aus der linearen Algebra, dass genau dann eine eindeutige Lösung existiert, wenn die Determinante $\det(\mathbf{A})$ nicht verschwindet (gleichbedeutend mit \mathbf{A} ist invertierbar bzw. \mathbf{A} ist regulär), d.h. wenn eine Matrix \mathbf{A}^{-1} existiert, so dass $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{I}$, wobei \mathbf{I} die $n \times n$ Einheitsmatrix ist (die Einträge auf der Diagonalen sind 1, alle anderen Einträge sind 0).

Bei der numerischen Lösung von Systemen der Art (4.2) unterscheidet man zwischen

- direkten Verfahren

Mit einem direkten Verfahren erhält man mit einer endlichen Zahl von Rechenschritten die exakte Lösung (wenn man Rundungsfehler vernachlässigt)

- iterativen Verfahren

Hier wird eine Folge von Vektoren erzeugt, die gegen die Lösung von (4.2) konvergiert.

Wir beginnen mit den direkten Verfahren. Hierfür benötigen wir die Definition der oberen bzw. unteren Dreiecksform.

Definition 4.1: Untere Dreiecksmatrix / Obere Dreiecksmatrix [6]

- Eine $n \times n$ Matrix $\mathbf{L} = (l_{ij})$ heisst **untere Dreiecksmatrix**, wenn $l_{ij} = 0$ für $j > i$ gilt; sie heisst **normierte untere Dreiecksmatrix**, wenn außerdem $l_{ii} = 1$ für alle i gilt.
- Eine $n \times n$ Matrix $\mathbf{R} = (r_{ij})$ heisst **obere Dreiecksmatrix**, wenn $r_{ij} = 0$ für $i > j$ gilt; sie heisst **normierte obere Dreiecksmatrix**, wenn außerdem $r_{ii} = 1$ für alle i gilt.

Beispiel 4.1

- Untere normierte Dreiecksmatrix:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn-1} & 1 \end{pmatrix}$$

- Obere Dreiecksmatrix:

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ 0 & 0 & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & r_{nn} \end{pmatrix}$$

4.3 Der Gauss-Algorithmus

Das Eliminationsverfahren nach Gauss (der ‘‘Gauss-Algorithmus’’) ist ein anschauliches Verfahren, das zudem gut implementiert werden kann. Es beruht auf der Tatsache, dass ein lineares Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ leicht lösbar ist, falls die Matrix \mathbf{A} als obere Dreiecksmatrix vorliegt, also alle Elemente unterhalb der Diagonalen verschwinden.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix} \quad (4.3)$$

In diesem Fall kann man für $\mathbf{A}\mathbf{x} = \mathbf{b}$ mittels der folgenden rekursiven Vorschrift, dem sogenannten Rückwärtseinsetzen, die Komponenten von \mathbf{x} berechnen:

$$x_n = \frac{b_n}{a_{nn}}, \quad x_{n-1} = \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}, \quad \dots, \quad x_1 = \frac{b_1 - a_{12}x_2 - \dots - a_{1n}x_n}{a_{11}} \quad (4.4)$$

oder, kompakt geschrieben,

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}}, \quad i = n, n-1, \dots, 1. \quad (4.5)$$

Die Idee des Gauss-Algorithmus ist nun, ein beliebiges Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ umzuformen in ein äquivalentes Gleichungssystem $\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$, so dass die Matrix $\tilde{\mathbf{A}}$ als obere Dreiecksmatrix vorliegt.

Bei dieser Transformation sind folgende Umformungen zugelassen:

- $z_j := z_j - \lambda z_i$ mit $i < j$ ($\lambda \in \mathbb{R}$), wobei z_i die i -te Zeile des Gleichungssystems bezeichnet
- $z_i \rightarrow z_j$: Vertauschen der i -ten und j -ten Zeile im System

Es sind also nur Zeilenvertauschungen und die Subtraktion eines Vielfachens einer Zeile von einer darunter stehenden Zeile erlaubt. Mit diesen beiden Umformungen kann jede Matrix \mathbf{A} in die obere Dreiecksform (4.3) gebracht werden (natürlich müssen für die Lösung von $\mathbf{A}\mathbf{x} = \mathbf{b}$ die Umformungen auch auf \mathbf{b} angewandt werden).

Beispiel 4.2

- Es soll folgendes System $\mathbf{A}\mathbf{x} = \mathbf{b}$ gelöst werden, wobei

$$\mathbf{A} = \begin{pmatrix} 1 & 5 & 6 \\ 7 & 9 & 6 \\ 2 & 3 & 4 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 29 \\ 43 \\ 20 \end{pmatrix}$$

Um die Matrix auf die obere Dreiecksgestalt zu bringen, müssen wir die Einträge 7, 3, und 2 unterhalb der Diagonalen eliminieren. Wir subtrahieren das 7-fache der ersten Zeile von der zweiten Zeile ($z_2 \equiv z_2 - 7z_1$) und erhalten

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 5 & 6 \\ 0 & -26 & -36 \\ 2 & 3 & 4 \end{pmatrix} \text{ und } \mathbf{b}_1 = \begin{pmatrix} 29 \\ -160 \\ 20 \end{pmatrix}.$$

Jetzt subtrahieren 2-mal die erste Zeile von der letzten ($z_3 \equiv z_3 - 2z_1$) und erhalten

$$\mathbf{A}_2 = \begin{pmatrix} 1 & 5 & 6 \\ 0 & -26 & -36 \\ 0 & -7 & -8 \end{pmatrix} \text{ und } \mathbf{b}_2 = \begin{pmatrix} 29 \\ -160 \\ -38 \end{pmatrix}.$$

Im letzten Schritt subtrahieren wir $7/26$ -mal die zweite Zeile von der dritten ($z_3 \equiv z_3 - \frac{7}{26}z_2$):

$$\mathbf{A}_3 = \begin{pmatrix} 1 & 5 & 6 \\ 0 & -26 & -36 \\ 0 & 0 & \frac{22}{13} \end{pmatrix} \text{ und } \mathbf{b}_3 = \begin{pmatrix} 29 \\ -160 \\ \frac{66}{13} \end{pmatrix}.$$

Somit erhalten wir über Rückwärtseinsetzen gemäss (4.4) die gesuchten Komponenten von x :

$$x_3 = \frac{\frac{66}{13}}{\frac{22}{13}} = 3, x_2 = \frac{-160 - 3 \cdot (-36)}{-26} = 2 \text{ und } x_1 = \frac{29 - 2 \cdot 5 - 3 \cdot 6}{1} = 1.$$

Für die Programmierung des Gauss-Algorithmus sollte nun folgendermassen vorgegangen werden:

- Zuerst erzeugt man Nullen in der ersten Spalte unterhalb von a_{11} mit der Operation $z_j := z_j - \frac{a_{j1}}{a_{11}} z_i$ mit $j = 2, \dots, n$.
Dies geht nur, falls $a_{11} \neq 0$. Ist $a_{11} = 0$, so vertauschen wir die erste Zeile mit der i-ten Zeile, wobei $a_{i1} \neq 0$ sein muss. Falls alle Zeilen der Matrix in der ersten Zeile eine Null besitzen funktioniert die Vertauschung nicht. Dann ist allerdings auch die Matrix nicht regulär und die Lösungsmenge kann leer sein oder auch unendlich viele Elemente enthalten.
- Dieser Schritt wird nun wiederholt, in dem man mit der zweiten Spalte fortfährt und unterhalb der Diagonalen Nullen erzeugt.

Schliesslich erhält man den

Gauss-Algorithmus zur Transformation von $\mathbf{A}x = \mathbf{b}$ auf ein oberes Dreiecksystem [1]:

- für $i = 1, \dots, n$:
 - erzeuge Nullen unterhalb des Diagonalelementes in der i-ten Spalte
 - Falls nötig und möglich, sorge durch Zeilenvertauschung für $a_{ii} \neq 0$:
falls $a_{ii} \neq 0$: tue nichts
 - falls $a_{ii} = 0$:
 - falls $a_{ji} = 0$ für alle $j = i + 1, \dots, n$:
 A ist nicht regulär; stop;
 - wenn $a_{ji} \neq 0$ für ein $j = i + 1, \dots, n$:
 - sei $j \geq i + 1$ der kleinste Index mit $a_{ji} \neq 0$
 - $z_i \longleftrightarrow z_j$
 - Eliminationsschritt:
für $j = i + 1, \dots, n$ eliminiere das Element a_{ji} durch:

$$z_j := z_j - \frac{a_{ji}}{a_{ii}} \cdot z_i$$

Aufgabe 4.1 [1]

- Bringen Sie das Gleichungssystem $\mathbf{A}x = \mathbf{b}$ auf die obere Dreiecksform und lösen Sie nach x auf, wobei

$$\mathbf{A} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -3 & -2 \\ 5 & 1 & 4 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 0 \\ 5 \\ 3 \end{pmatrix}$$

Lösung:

$$i = 1, j = 2 \Rightarrow z_2 \equiv z_2 - \frac{1}{(-1)} z_1 \Rightarrow (\mathbf{A}_1 | \mathbf{b}_1) = \left(\begin{array}{ccc|c} -1 & 1 & 1 & 0 \\ 0 & -2 & -1 & 5 \\ 5 & 1 & 4 & 3 \end{array} \right)$$

$$i = 1, j = 3 \Rightarrow z_3 \equiv z_3 - \frac{5}{(-1)} z_1 \Rightarrow (\mathbf{A}_2 | \mathbf{b}_2) = \left(\begin{array}{ccc|c} -1 & 1 & 1 & 0 \\ 0 & -2 & -1 & 5 \\ 0 & 6 & 9 & 3 \end{array} \right)$$

$$i = 2, j = 3 \Rightarrow z_3 \equiv z_3 - \frac{6}{(-2)} z_2 \Rightarrow (\mathbf{A}_3 | \mathbf{b}_3) = \left(\begin{array}{ccc|c} -1 & 1 & 1 & 0 \\ 0 & -2 & -1 & 5 \\ 0 & 0 & 6 & 18 \end{array} \right)$$

Rückeinsetzen gemäss (4.4) liefert:

$$x_3 = \frac{18}{6} = 3, x_2 = \frac{5 - (-1) \cdot 3}{(-2)} = -4, x_1 = \frac{0 - 1 \cdot (-4) - 1 \cdot 3}{(-1)} = -1$$

Natürlich muss man, wenn man $\mathbf{Ax} = \mathbf{b}$ schon mal gelöst hat und nun die Lösung von $\mathbf{Ax} = \mathbf{c}$ für einen neuen Vektor \mathbf{c} bestimmen will, die Matrix \mathbf{A} nicht nochmal auf die obere Dreiecksform bringen, sondern wendet die Zeilenumformungen einfach auf den neuen Vektor \mathbf{c} an mit anschliessendem Rückwärtseinsetzen.

Aufgabe 4.3 [1]

- Es soll das Gleichungssystem $\mathbf{Ax} = \mathbf{c}$ mit der Matrix \mathbf{A} aus der vorherigen Aufgabe gelöst werden für $\mathbf{c} = (13, -32, 22)^T$

Lösung:

$$z_2 \equiv z_2 - \frac{1}{(-1)} z_1 \Rightarrow \mathbf{c}_1 = \begin{pmatrix} 13 \\ -19 \\ 22 \end{pmatrix}$$

$$z_3 \equiv z_3 - \frac{5}{(-1)} z_1 \Rightarrow \mathbf{c}_2 = \begin{pmatrix} 13 \\ -19 \\ 87 \end{pmatrix}$$

$$z_3 \equiv z_3 - \frac{6}{(-2)} z_2 \Rightarrow \mathbf{c}_3 = \begin{pmatrix} 13 \\ -19 \\ 30 \end{pmatrix}$$

Rückeinsetzen liefert die Lösung $\mathbf{x} = (-1, 7, 5)^T$

Eine zusätzliche Anwendung des Gauss-Algorithmus ist die Determinantenbestimmung. Wenn wir mit $\tilde{\mathbf{A}}$ die obere Dreiecksmatrix von \mathbf{A} bezeichnen, dann gilt die Beziehung

$$\det(\mathbf{A}) = (-1)^l \cdot \det(\tilde{\mathbf{A}}) = (-1)^l \prod_{i=1}^n \tilde{a}_{ii}$$

wobei \tilde{a}_{ii} die Diagonalelemente von $\tilde{\mathbf{A}}$ sind und l die Anzahl der im Laufe des Gauss-Algorithmus vorgenommenen Zeilenumtauschungen.

Aufgabe 4.2 [1]

- Bestimmen Sie die Determinante der Matrix \mathbf{A} aus Aufgabe 4.1 mittels des Gauss-Algorithmus.

Lösung: die obere Dreiecksform $\tilde{\mathbf{A}}$ von \mathbf{A} haben wir bereits berechnet:

$$\tilde{\mathbf{A}} = \mathbf{A}_3 = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & -1 \\ 0 & 0 & 6 \end{pmatrix}$$

Dabei wurde keine Zeilenumtauschung durchgeführt, d.h. die Determinante von \mathbf{A} ist das Produkt der Diagonalelemente von $\tilde{\mathbf{A}}$:

$$\det(\mathbf{A}) = (-1) \cdot (-2) \cdot 6 = 12$$

Aufgabe 4.3 [1]

- Optional: Implementieren Sie den Gauss-Algorithmus in Python und bestimmen Sie damit die Lösungen für die untenstehenden Gleichungssysteme sowie die Determinanten der Matrizen $\mathbf{A}_1 - \mathbf{A}_4$. Wer die Aufgabe lieber von Hand löst, kann dies unter Angabe aller Zwischenschritte tun.

$$\mathbf{A}_1 \mathbf{x} = \begin{pmatrix} 4 & -1 & -5 \\ -12 & 4 & 17 \\ 32 & -10 & -41 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} -5 \\ 19 \\ -39 \end{pmatrix} \text{ bzw. } = \begin{pmatrix} 6 \\ -12 \\ 48 \end{pmatrix}$$

$$\mathbf{A}_2 \mathbf{x} = \begin{pmatrix} 2 & 7 & 3 \\ -4 & -10 & 0 \\ 12 & 34 & 9 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 25 \\ -24 \\ 107 \end{pmatrix} \text{ bzw. } = \begin{pmatrix} 5 \\ -22 \\ 42 \end{pmatrix}$$

$$\mathbf{A}_3 \mathbf{x} = \begin{pmatrix} -2 & 5 & 4 \\ -14 & 38 & 22 \\ 6 & -9 & -27 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 1 \\ 40 \\ 75 \end{pmatrix} \text{ bzw. } = \begin{pmatrix} 16 \\ 82 \\ -120 \end{pmatrix}$$

$$\mathbf{A}_4 \mathbf{x} = \begin{pmatrix} -1 & 2 & 3 & 2 & 5 & 4 & 3 & -1 \\ 3 & 4 & 2 & 1 & 0 & 2 & 3 & 8 \\ 2 & 7 & 5 & -1 & 2 & 1 & 3 & 5 \\ 3 & 1 & 2 & 6 & -3 & 7 & 2 & -2 \\ 5 & 2 & 0 & 8 & 7 & 6 & 1 & 3 \\ -1 & 3 & 2 & 3 & 5 & 3 & 1 & 4 \\ 8 & 7 & 3 & 6 & 4 & 9 & 7 & 9 \\ -3 & 14 & -2 & 1 & 0 & -2 & 10 & 5 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} -11 \\ 103 \\ 53 \\ -20 \\ 95 \\ 78 \\ 131 \\ -26 \end{pmatrix}$$

4.4 Fehlerfortpflanzung beim Gauss-Algorithmus und Pivotisierung

Im vorherigen Abschnitt haben wir Zeilen nur vertauscht, falls ein Diagonalelement im Laufe der Berechnungen Null wurde. Man kann aber Zeilenvertauschungen aber auch dazu verwenden, um Fehler z.B. durch Gleitpunktoperationen, zu minimieren.

In jedem Eliminationsschritt wurden die Zeilen bisher mit $\lambda = \frac{a_{ji}}{a_{ii}}$ multipliziert, d.h. der absolute Fehler vergrößerte sich um den Faktor $|\lambda|$ (siehe Kap. 2). Wünschenswert wäre es also, wenn $|\lambda| = |\frac{a_{ji}}{a_{ii}}| < 1$. Dies lässt sich einfach dadurch erreichen, dass man vor dem Eliminationsschritt überprüft, welches Element in der Spalte betragsmäßig am grössten ist und die Zeile vertauscht, so dass dieses grösste Element zum Diagonalelement wird. Dieses Vorgehen wird Spaltenpivotisierung genannt.

Gauss-Algorithmus zur Transformation von $Ax = b$ mit Spaltenpivotisierung [1]:

- für $i = 1, \dots, n$:

erzeuge Nullen unterhalb des Diagonalelementes in der i-ten Spalte

– Suche das betragsgrösste Element unterhalb der Diagonalen in der i-ten Spalte:

Wähle k so, dass $|a_{ki}| = \max\{|a_{ji}| \mid j = i, \dots, n\}$

$\begin{cases} \text{falls } a_{ki} = 0 : A \text{ ist nicht regulär; stop;} \\ \text{falls } a_{ki} \neq 0 : z_k \longleftrightarrow z_i; \end{cases}$

– Eliminationsschritt:

für $j = i + 1, \dots, n$ eliminiere das Element a_{ji} durch:

$$z_j := z_j - \frac{a_{ji}}{a_{ii}} \cdot z_i$$

Beispiel 4.4 [1]

- Die Matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -1 \\ 4 & -2 & 6 \\ 3 & 1 & 0 \end{pmatrix}$$

soll mittels Spaltenpivotisierung auf die (rechts-) obere Dreiecksform gebracht werden.

Lösung:

$$\begin{aligned} \mathbf{A} := \begin{pmatrix} 1 & 2 & -1 \\ 4 & -2 & 6 \\ 3 & 1 & 0 \end{pmatrix} &\xrightarrow{z_1 \leftrightarrow z_2} \begin{pmatrix} 4 & -2 & 6 \\ 1 & 2 & -1 \\ 3 & 1 & 0 \end{pmatrix} \xrightarrow{z_2 := z_2 - 0.25 z_1} \begin{pmatrix} 4 & -2 & 6 \\ 0 & 2.5 & -2.5 \\ 3 & 1 & 0 \end{pmatrix} \\ &\xrightarrow{z_3 := z_3 - 0.75 z_1} \begin{pmatrix} 4 & -2 & 6 \\ 0 & 2.5 & -2.5 \\ 0 & 2.5 & -4.5 \end{pmatrix} \xrightarrow{z_3 := z_3 - z_2} \begin{pmatrix} 4 & -2 & 6 \\ 0 & 2.5 & -2.5 \\ 0 & 0 & -2.5 \end{pmatrix} \end{aligned}$$

Bemerkung: in dieser Lösung aus [1] hat es einen Fehler ... wo?

4.5 Dreieckszerlegung von Matrizen

4.5.1 Die LR-Zerlegung

In der obigen Version des Gauß-Verfahrens haben wir die Matrix \mathbf{A} auf obere Dreiecksform gebracht und zugleich alle dafür notwendigen Operationen auch auf den Vektor \mathbf{b} angewendet. Es gibt alternative Möglichkeiten, lineare Gleichungssysteme zu lösen, bei denen der Vektor \mathbf{b} unverändert bleibt. Wir werden nun ein Verfahren kennenlernen, bei dem die Matrix \mathbf{A} in ein Produkt von zwei Matrizen \mathbf{L} und \mathbf{R} zerlegt wird, also $\mathbf{A} = \mathbf{LR}$, wobei \mathbf{R} eine obere Dreiecksmatrix und \mathbf{L} eine untere normierte Dreiecksmatrix ist:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn-1} & 1 \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ 0 & 0 & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & r_{nn} \end{pmatrix} \quad (4.6)$$

Die Zerlegung $\mathbf{A} = \mathbf{LR}$ wird als **LR-Faktorisierung** oder **LR-Zerlegung** bezeichnet. Das ursprüngliche Gleichungssystem

$$\mathbf{Ax} = \mathbf{b}$$

lautet dann

$$\mathbf{LRx} = \mathbf{b} \iff \mathbf{Ly} = \mathbf{b} \text{ und } \mathbf{Rx} = \mathbf{y}$$

und lässt sich wie folgt in zwei Schritten lösen:

1. Zunächst löst man das Gleichungssystem $\mathbf{Ly} = \mathbf{b}$. Dies kann, ganz analog zum Rückwärtseinsetzen (4.4) durch Vorwärtseinsetzen geschehen:

$$y_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij} y_j}{l_{ii}}, \quad i = 1, 2, \dots, n. \quad (4.7)$$

2. Anschliessend löst man durch Rückwärtseinsetzen das Gleichungssystem $\mathbf{Rx} = \mathbf{y}$. Dann gilt

$$\mathbf{Ax} = \mathbf{LRx} = \mathbf{Ly} = \mathbf{b} \quad (4.8)$$

womit das System $\mathbf{Ax} = \mathbf{b}$ gelöst ist.

Wir haben beim Gauss-Algorithmus bereits gesehen, dass sich eine beliebige Matrix durch Zeilenumformungen in eine obere Dreiecksform transformieren lässt. Im Folgenden gehen wir davon aus, dass Zeilenvertauschungen nicht notwendig sind. Der Gauss-Algorithmus lässt sich dann so erweitern, dass damit eine **LR-Zerlegung** einer invertierbaren Matrix \mathbf{A} möglich ist. Tatsächlich gilt:

- \mathbf{R} ist gerade die durch den Gauss-Algorithmus auf die obere Dreiecksform gebrachte Matrix $\tilde{\mathbf{A}}$
- Die Elemente l_{ji} von \mathbf{L} entsprechen gerade den berechneten Faktoren λ aus den Eliminationsschritten $z_j := z_j - \lambda_{ji} z_i$, also $l_{ji} = \lambda_{ji}$

Beispiel 4.5

- Wir berechnen für die Matrix A aus Aufgabe 4.1 die normierte untere Dreiecksmatrix L und die obere Dreiecksmatrix R , so dass $A = LR$.

Lösung: Wir hatten

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

und

$$i = 1, j = 2 \Rightarrow z_2 \equiv z_2 - \underbrace{\frac{1}{(-1)} z_1}_{l_{21}} \Rightarrow A_1 = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & -1 \\ 5 & 1 & 4 \end{pmatrix}$$

$$i = 1, j = 3 \Rightarrow z_3 \equiv z_3 - \underbrace{\frac{5}{(-1)} z_1}_{l_{31}} \Rightarrow A_2 = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & -1 \\ 0 & 6 & 9 \end{pmatrix}$$

$$i = 2, j = 3 \Rightarrow z_3 \equiv z_3 - \underbrace{\frac{6}{(-2)} z_2}_{l_{32}} \Rightarrow A_3 = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & -1 \\ 0 & 0 & 6 \end{pmatrix} = R$$

Das heisst, wir können

$$R = A_3 = \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & -1 \\ 0 & 0 & 6 \end{pmatrix}$$

setzen und für die Elemente von L erhalten wir aus den 3 Eliminationsschritten die drei Elemente

$$\begin{aligned} l_{21} &= \frac{1}{(-1)} = -1 \\ l_{31} &= \frac{5}{(-1)} = -5 \\ l_{32} &= \frac{6}{(-2)} = -3 \end{aligned}$$

und damit

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -5 & -3 & 1 \end{pmatrix}$$

Die Probe ergibt wie gewünscht

$$LR = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -5 & -3 & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 1 \\ 0 & -2 & -1 \\ 0 & 0 & 6 \end{pmatrix} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -3 & -2 \\ 5 & 1 & 4 \end{pmatrix} = A$$

Es gilt der folgende

Satz 4.1: LR -Zerlegung [1]

Zu jeder regulären $n \times n$ Matrix A , für die der Gauss-Algorithmus ohne Zeilenumtauschung durchführbar ist, gibt es $n \times n$ Matrizen L und R mit den folgenden Eigenschaften:

- L ist eine normierte untere Dreiecksmatrix (also mit $l_{ii} = 1$ für $i = 1, \dots, n$)
- R ist eine obere Dreiecksmatrix mit $r_{ii} \neq 0$ für $i = 1, \dots, n$
- $A = L \cdot R$ ist die LR -Zerlegung von A .

Aufwand: Die Berechnung der LR -Zerlegung mit dem Gauss-Algorithmus benötigt ca. $\frac{2}{3}n^3$ Punktoperationen

Bemerkungen:

1. Die direkte Lösung von $\mathbf{A}\mathbf{x} = \mathbf{b}$ durch die Berechnung der inversen \mathbf{A}^{-1} ist nicht praktikabel, da dies die Lösung von n linearen Gleichungssystemen erfordern würde und damit erheblich aufwendiger wäre.
2. Ein mit der **LR -Zerlegung** (in Englisch **LU -decomposition**) verwandter Algorithmus wird auch teilweise angewendet als Benchmark für die Rechengeschwindigkeit. Aus Wikipedia: “ *LU reduction is an algorithm related to LU decomposition. This term is usually used in the context of super computing and highly parallel computing. In this context it is used as a benchmarking algorithm, i.e. to provide a comparative measurement of speed for different computers. LU reduction is a special parallelized version of an LU decomposition algorithm, an example can be found in (Guitart 2001). The parallelized version usually distributes the work for a matrix row to a single processor and synchronizes the result with the whole matrix (Escribano 2000)*”.
3. Unter anderem ist die **LR -Zerlegung** eine geschickte Variante, die Zwischenresultate des Gauss-Algorithmus zu speichern.

Aufgabe 4.4 [1]:

- Finden Sie für die Matrix A des linearen Gleichungssystems $\mathbf{A}\mathbf{x} = \mathbf{b}$ mit

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -1 \\ 4 & -2 & 6 \\ 3 & 1 & 0 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 9 \\ -4 \\ 9 \end{pmatrix}$$

die **LR -Zerlegung**. Benutzen Sie dafür die folgenden Operationen der Gauss-Elimination:

$$(\mathbf{A} \mid \mathbf{b}) = \left(\begin{array}{ccc|c} 1 & 2 & -1 & 9 \\ 4 & -2 & 6 & -4 \\ 3 & 1 & 0 & 9 \end{array} \right) \xrightarrow{z_2 := z_2 - 4z_1} \left(\begin{array}{ccc|c} 1 & 2 & -1 & 9 \\ 0 & -10 & 10 & -40 \\ 3 & 1 & 0 & 9 \end{array} \right)$$

$$\xrightarrow{z_3 := z_3 - 3z_1} \left(\begin{array}{ccc|c} 1 & 2 & -1 & 9 \\ 0 & -10 & 10 & -40 \\ 0 & -5 & 3 & -18 \end{array} \right) \xrightarrow{z_3 := z_3 - 0.5z_2} \left(\begin{array}{ccc|c} 1 & 2 & -1 & 9 \\ 0 & -10 & 10 & -40 \\ 0 & 0 & -2 & 2 \end{array} \right)$$

Berechnen Sie die Lösung \mathbf{x} zuerst mittels Rückwärtseinsetzen direkt aus der obigen Dreiecksform und dem \mathbf{b} Vektor. Lösen Sie anschliessend die beiden linearen Systeme $\mathbf{Ly} = \mathbf{b}$ und $\mathbf{Rx} = \mathbf{y}$ und vergleichen Sie.

- Optional: Erweitern Sie ihr unter Aufgabe 4.3 erstelltes Programm zum Gauss-Algorithmus, so dass es gleichzeitig auch die **LR -Zerlegung** von \mathbf{A} berechnet. Berechnen Sie damit die **LR -Zerlegung** für die Matrixen aus Aufgabe 4.3.

4.5.1.1 Die **LR -Zerlegung mit Zeilenumtauschung**

Sind Zeilenumtauschungen nötig, so lässt sich in der Regel keine **LR -Zerlegung** erhalten. Allerdings lässt sich die Vertauschung der i -ten und j -ten Zeile in \mathbf{A} durch eine Multiplikation von links mit einer $n \times n$ Matrix \mathbf{P}_k der Form

$$\mathbf{P}_k = \left(\begin{array}{ccccc|c} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline & & 0 & & 1 & \\ \hline & & & 1 & & \\ & & & & \ddots & \\ & & & & 1 & \\ \hline & & 1 & & 0 & \\ \hline & & & & & 1 \\ & & & & & & \ddots \\ & & & & & & & 1 \end{array} \right) \quad \begin{array}{l} i - \text{te Zeile} \\ \\ j - \text{te Zeile} \end{array}$$

erreichen. \mathbf{P}_k erhält man aus der Einheitsmatrix \mathbf{I}_n durch Vertauschung der i -ten und j -ten Zeile. Es gilt dann also $p_{ii} = p_{jj} = 0$ bzw. $p_{ij} = p_{ji} = 1$. Der ganzzahlige Index $k = 1, 2, \dots$ dient hier nur dazu, mehrere solcher Matrizen voneinander unterscheiden zu können, denn bei mehreren Zeilenvertauschungen können die dafür benötigten Matrizen $\mathbf{P}_1, \dots, \mathbf{P}_l$ zu einer einzigen Matrix $\mathbf{P} = \mathbf{P}_l \cdot \dots \cdot \mathbf{P}_1$ aufmultipliziert werden (bei linksseitiger Multiplikation). Die Matrix \mathbf{P} nennt sich die Permutationsmatrix, sie ist immer regulär und es gilt $\mathbf{P}^{-1} = \mathbf{P}$.

Beispiel 4.6

- Die Vertauschung der 2. und 4. Zeile bei der Matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \text{ führt zu } \mathbf{A}^* = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \\ 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \end{pmatrix},$$

welches sich auch durch die Multiplikation von links

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \\ 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

ausdrücken lässt, also $\mathbf{P}_1 \cdot \mathbf{A} = \mathbf{A}^*$.

- Die zusätzliche Vertauschung der 1. und 3. Zeile, also

$$\mathbf{A}^* = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \\ 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \end{pmatrix} \text{ geht über in } \mathbf{A}^{**} = \begin{pmatrix} 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix},$$

lässt sich darstellen durch die Multiplikation von links mit $\mathbf{P}_2 \cdot \mathbf{A}^* = \mathbf{A}^{**}$:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \\ 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \end{pmatrix} = \begin{pmatrix} 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

- Die beiden Zeilenvertauschungen können zusammengefasst werden durch Multiplikation von $\mathbf{P} = \mathbf{P}_2 \cdot \mathbf{P}_1$, also $\mathbf{P} \cdot \mathbf{A} = \mathbf{A}^{**}$ wobei

$$\mathbf{P} = \mathbf{P}_2 \cdot \mathbf{P}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Mit dieser Permutationsmatrix erhält man dann als **RL-Zerlegung**

$$\mathbf{PA} = \mathbf{LR}$$

und das lineare Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ lässt sich schreiben als $\mathbf{PAx} = \mathbf{Pb}$ bzw. $\mathbf{Lrx} = \mathbf{Pb}$ und in den zwei Schritten lösen:

$$\begin{aligned} \mathbf{Ly} &= \mathbf{Pb} \Rightarrow \mathbf{y} = \dots \\ \mathbf{Rx} &= \mathbf{y} \Rightarrow \mathbf{x} = \dots \end{aligned}$$

Wird die Zerlegung mittels Gauss-Algorithmus mit Spaltenpivotisierung (vgl. Kap. 4.4) durchgeführt, muss man also bei jeder Zeilenvertauschung die dazugehörige Permutationsmatrix berechnen. Zusätzlich muss jede Zeilenvertauschung fortlaufend auch in \mathbf{L} für die Elemente in den jeweiligen Zeilen aber nur unterhalb der Diagonalen durchgeführt werden (die Diagonalelemente $l_{ii} = 1$ verändern ihre Position nicht). So erhält man schliesslich \mathbf{L} , \mathbf{R} und \mathbf{P} . Dieses Verfahren nennt man auch **LR-Zerlegung mit Spalten- bzw. Kolonnenmaximumstrategie**.

Beispiel 4.7

- Gegeben ist das Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ mit

$$\mathbf{A} = \begin{pmatrix} 3 & 9 & 12 & 12 \\ -2 & -5 & 7 & 2 \\ 6 & 12 & 18 & 6 \\ 3 & 7 & 38 & 14 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 51 \\ 2 \\ 54 \\ 79 \end{pmatrix}$$

Berechen Sie die \mathbf{LR} -Zerlegung von \mathbf{A} mit Spaltenmaximumstrategie und bestimmen Sie anhand von \mathbf{L} , \mathbf{R} und \mathbf{P} die Lösung \mathbf{x} .

- Lösung:

- Zeilenvertauschung von 1. Zeile mit 3. Zeile in \mathbf{A} , so dass mit $a_{31} = 6$ das betragsmäßig grösste Element auf der Diagonale liegt. \mathbf{P}_1 bildet diese Zeilenvertauschung ab. Da die Elemente in \mathbf{L} unterhalb der Diagonalen noch unbestimmt sind, hat eine Zeilenvertauschung noch keinen Einfluss.

$$\mathbf{A}^* = \begin{pmatrix} 6 & 12 & 18 & 6 \\ -2 & -5 & 7 & 2 \\ 3 & 9 & 12 & 12 \\ 3 & 7 & 38 & 14 \end{pmatrix}, \quad \mathbf{P}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ ? & 1 & 0 & 0 \\ ? & ? & 1 & 0 \\ ? & ? & ? & 1 \end{pmatrix},$$

$$i = 1, j = 2 \Rightarrow z_2 \equiv z_2 - \frac{(-2)}{6}z_1 \Rightarrow \mathbf{A}_1^* = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & -1 & 13 & 4 \\ 3 & 9 & 12 & 12 \\ 3 & 7 & 38 & 14 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ ? & ? & 1 & 0 \\ ? & ? & ? & 1 \end{pmatrix}$$

$$i = 1, j = 3 \Rightarrow z_3 \equiv z_3 - \frac{3}{6}z_1 \Rightarrow \mathbf{A}_2^* = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & -1 & 13 & 4 \\ 0 & 3 & 3 & 9 \\ 3 & 7 & 38 & 14 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ \frac{1}{2} & ? & 1 & 0 \\ ? & ? & ? & 1 \end{pmatrix}$$

$$i = 1, j = 4 \Rightarrow z_4 \equiv z_4 - \frac{3}{6}z_1 \Rightarrow \mathbf{A}_3^* = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & -1 & 13 & 4 \\ 0 & 3 & 3 & 9 \\ 0 & 1 & 29 & 11 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ \frac{1}{2} & ? & 1 & 0 \\ \frac{1}{2} & ? & ? & 1 \end{pmatrix}$$

- Zeilenvertauschung von 2. Zeile mit 3. Zeile in \mathbf{A}_3^* , auch für die Elemente in der ersten Spalte von \mathbf{L} .

$$\mathbf{A}^{**} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & -1 & 13 & 4 \\ 0 & 1 & 29 & 11 \end{pmatrix}, \quad \mathbf{P}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ -\frac{1}{3} & ? & 1 & 0 \\ \frac{1}{2} & ? & ? & 1 \end{pmatrix}$$

$$i = 2, j = 3 \Rightarrow z_3 \equiv z_3 - \frac{(-1)}{3}z_2 \Rightarrow \mathbf{A}_1^{**} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & 0 & 14 & 7 \\ 0 & 1 & 29 & 11 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & 1 & 0 \\ \frac{1}{2} & ? & ? & 1 \end{pmatrix}$$

$$i = 2, j = 4 \Rightarrow z_4 \equiv z_4 - \frac{1}{3}z_2 \Rightarrow \mathbf{A}_2^{**} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & 0 & 14 & 7 \\ 0 & 0 & 28 & 8 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & 1 & 0 \\ \frac{1}{2} & \frac{1}{3} & ? & 1 \end{pmatrix}$$

- Zeilenvertauschung von 4. Zeile mit 3. Zeile in \mathbf{A}_2^{**} , auch für die Elemente in der ersten und zweiten Spalte von \mathbf{L} :

$$\mathbf{A}^{***} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & 0 & 28 & 8 \\ 0 & 0 & 14 & 7 \end{pmatrix}, \quad \mathbf{P}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ -\frac{2}{3} & \frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & ? & 1 \end{pmatrix}$$

$$i = 3, j = 4 \Rightarrow z_4 \equiv z_4 - \frac{1}{2}z_3 \Rightarrow \mathbf{A}_1^{***} = \mathbf{R} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & 0 & 28 & 8 \\ 0 & 0 & 0 & 3 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{2} & 1 \end{pmatrix}$$

1. Als Resultat erhalten wir damit:

$$\mathbf{R} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & 0 & 28 & 8 \\ 0 & 0 & 0 & 3 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{2} & 1 \end{pmatrix}, \quad \mathbf{P} = \mathbf{P}_3 \cdot \mathbf{P}_2 \cdot \mathbf{P}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

und es gilt wie gewünscht

$$\mathbf{LR} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 3 & 9 & 12 & 12 \\ 3 & 7 & 38 & 14 \\ -2 & -5 & 7 & 2 \end{pmatrix} = \mathbf{PA}$$

2. Für die zu lösenden Gleichungssysteme

$$\begin{aligned} \mathbf{Ly} &= \mathbf{Pb} \\ \mathbf{Rx} &= \mathbf{y} \end{aligned}$$

erhalten wir den Vektor \mathbf{y} durch Vorwärtseinsetzen:

$$\mathbf{Ly} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \mathbf{Pb} = \begin{pmatrix} 54 \\ 51 \\ 79 \\ 2 \end{pmatrix} \Rightarrow \mathbf{y} = \begin{pmatrix} 54 \\ 24 \\ 44 \\ 6 \end{pmatrix}$$

und die eigentlich gesuchte Lösung \mathbf{x} durch Rückwärtseinsetzen:

$$\mathbf{Rx} = \begin{pmatrix} 6 & 12 & 18 & 6 \\ 0 & 3 & 3 & 9 \\ 0 & 0 & 28 & 8 \\ 0 & 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \mathbf{y} = \begin{pmatrix} 54 \\ 24 \\ 44 \\ 6 \end{pmatrix} \Rightarrow \mathbf{x} = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

4.5.2 Die QR-Zerlegung⁹

Neben der \mathbf{LR} -Zerlegung wollen wir nun eine weitere wichtige Zerlegung kennenlernen, die sogenannte \mathbf{QR} -Zerlegung. Sie ist u.a. wichtig für die Bestimmung von Eigenwerten von Matrizen (vgl. Kap. 4.8). \mathbf{R} ist dabei weiterhin eine rechtsobere Dreiecksmatrix, aber \mathbf{Q} ist nun keine Dreiecksmatrix mehr, besitzt aber die nützliche Eigenschaft der Orthogonalität.

Definition 4.2: Orthogonalmatrix / QR-Zerlegung [1]

- Eine Matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ heisst **orthogonal**, wenn $\mathbf{Q}^T \cdot \mathbf{Q} = \mathbf{I}_n$ ist. Dabei ist \mathbf{I}_n die $n \times n$ Einheitsmatrix. Man sagt auch kurz, \mathbf{Q} ist eine **Orthogonalmatrix**.
- Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Eine **QR-Zerlegung** von \mathbf{A} ist eine Darstellung von \mathbf{A} als Produkt einer orthogonalen $n \times n$ Matrix \mathbf{Q} und einer rechtsoberen $n \times n$ Dreiecksmatrix \mathbf{R} :

$$\mathbf{A} = \mathbf{QR}$$

⁹Kapitel hauptsächlich übernommen aus [1]

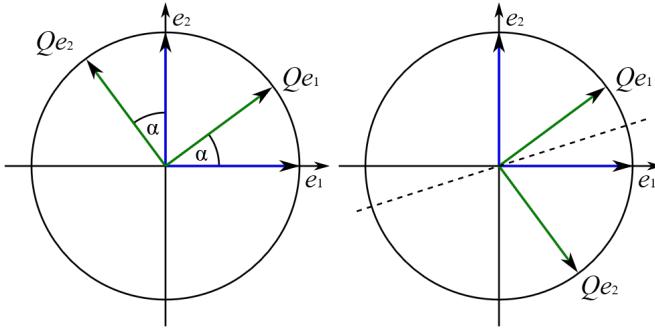


Abbildung 4.5: Durch Multiplikation mit einer orthogonalen Matrix Q können Vektoren gedreht (links) oder gespiegelt (rechts) werden. Die Länge der Vektoren und der Winkel zwischen den Vektoren bleiben dabei erhalten. (Quartl - Eigenes Werk, CC BY-SA 3.0, link)

Bemerkungen:

1. Eine orthogonale Matrix Q ist regulär mit $Q^{-1} = Q^T$:

$$Q^T \cdot Q = I_n \iff Q^T \cdot \underbrace{Q \cdot Q^{-1}}_{I_n} = \underbrace{I_n \cdot Q^{-1}}_{Q^{-1}} \iff Q^T = Q^{-1}$$

Wir können die Inverse also direkt aus der transponierten Matrix berechnen.

2. Die Spalten und Zeilen einer solchen Orthogonalmatrix stehen, wenn man sie als Vektoren interpretiert, also paarweise "senkrecht" zueinander und haben die Länge 1, d.h. sie sind orthonormal bzgl. des Standardskalarprodukts.
3. Orthogonale Matrizen beschreiben Drehungen und Spiegelungen oder Kombinationen daraus.
4. Die effiziente Berechnung der QR -Zerlegung einer $n \times n$ Matrix A benötigt mit etwa $\frac{5}{3}n^3$ Punktoperationen rund doppelt so viele wie für die LR -Zerlegung, kann dafür aber numerisch stabiler sein.

Beispiele 4.8

1. $Q = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ ist eine Orthogonalmatrix

Beweis:

$$Q^T \cdot Q = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2. $Q = \frac{1}{5} \begin{pmatrix} 3 & 4 \\ -4 & 3 \end{pmatrix}$ ist eine Orthogonalmatrix

Beweis:

$$Q^T \cdot Q = \frac{1}{5} \begin{pmatrix} 3 & -4 \\ 4 & 3 \end{pmatrix} \cdot \frac{1}{5} \begin{pmatrix} 3 & 4 \\ -4 & 3 \end{pmatrix} = \frac{1}{25} \begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

3. $Q = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$ ist eine Orthogonalmatrix und dreht einen Vektor um den Winkel α relativ zum Ursprung.

Wie können wir nun die QR -Zerlegung einer Matrix A berechnen? Die Idee ist, die benötigten Eliminations schritte durch einfache orthogonale Matrizen Q_i zu beschreiben und die gesuchte Matrix Q als Produkt der Q_i darzustellen. Die Q_i sind dabei die im Folgenden definierten Householder¹⁰-Matrizen.

¹⁰nach Alston S. Householder, US-amerikanischer Mathematiker (1904-1993)

Definition 4.3: Householder-Matrizen [1]

- Sei $\mathbf{u} \in \mathbb{R}^n$ ein Vektor der Länge 1, also $|\mathbf{u}| = \sqrt{u_1^2 + u_2^2 + \dots + u_n^2} = 1$. Die orthogonale $n \times n$ Matrix $\mathbf{H} := \mathbf{I}_n - 2\mathbf{u}\mathbf{u}^T$ heisst **Householder-Matrix**.
- Neben der Orthogonalität gilt weiter $\mathbf{H}^T = \mathbf{H}$, d.h. \mathbf{H} ist **symmetrisch**.

Bemerkungen:

1. $\mathbf{u}\mathbf{u}^T$ ist ebenfalls eine $n \times n$ Matrix (hier wird ja eine Spalte mit einer Zeile multipliziert)
2. Die durch Householder-Matrizen beschriebenen Abbildungen sind geometrisch gesehen Spiegelungen an einer zu \mathbf{u} senkrechten Hyperebene (d.h. einer Geraden für $n = 2$, einer Ebene für $n = 3$).
3. Da \mathbf{H} symmetrisch ist ($\mathbf{H}^T = \mathbf{H}$) und orthogonal ($\mathbf{H}^T = \mathbf{H}^{-1}$) gilt also

$$\mathbf{H} = \mathbf{H}^T = \mathbf{H}^{-1}$$

d.h. \mathbf{H} ist gleich wie seine Inverse und

$$\mathbf{H} \cdot \mathbf{H} = \mathbf{I}_n$$

Beispiel 4.9

- Berechnen Sie die Householder-Matrix zum Vektor

$$\mathbf{u} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

- Lösung:

Wir normieren zuerst den Vektor auf die Länge 1 und erhalten

$$\tilde{\mathbf{u}} = \frac{\mathbf{u}}{|\mathbf{u}|} = \frac{1}{\sqrt{1^2 + 2^2 + 3^2}} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \frac{1}{\sqrt{14}} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Damit wird

$$\begin{aligned} \mathbf{H} &= \mathbf{I}_n - 2\tilde{\mathbf{u}}\tilde{\mathbf{u}}^T \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \cdot \frac{1}{14} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{7} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix} \\ &= -\frac{1}{7} \begin{pmatrix} -6 & 2 & 3 \\ 2 & -3 & 6 \\ 3 & 6 & 2 \end{pmatrix} \end{aligned}$$

Wir wollen nun Householder-Matrizen benutzen, um eine $n \times n$ Matrix \mathbf{A} schrittweise auf eine rechts-obere Dreiecksform zu bringen, ganz ähnlich wie bei beim Gauss-Algorithmus, wo wir die benötigten Eliminationsschritte in die Elemente der Matrix \mathbf{L} eingeschrieben hatten. Mit jeweils einer Householder-Matrix werden wir jeweils eine Spalte von \mathbf{A} unterhalb der Diagonalen ausräumen (d.h. deren Elemente zu Null machen).

1. Betrachten wir den ersten Schritt gneauer, d.h. es geht nun um die erste Spalte von \mathbf{A} und die gesuchte Transformation \mathbf{H}_1 mit

$$\mathbf{H}_1 \cdot \mathbf{A} = \mathbf{H}_1 \cdot \underbrace{\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}}_{=\mathbf{A}} = \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix}$$

Wir finden das gesuchte \mathbf{H}_1 wie folgt: sei \mathbf{a}_1 die erste Spalte von \mathbf{A} und \mathbf{e}_1 der erste Einheitsvektor, also

$$\mathbf{a}_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \\ a_{51} \end{pmatrix} \quad \text{und} \quad \mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Dann definieren wir:

$$\begin{aligned} \mathbf{v}_1 &:= \mathbf{a}_1 + \text{sign}(a_{11}) \cdot |\mathbf{a}_1| \cdot \mathbf{e}_1 \\ \mathbf{u}_1 &:= \frac{1}{|\mathbf{v}_1|} \mathbf{v}_1 \\ \mathbf{H}_1 &:= \mathbf{I}_n - 2\mathbf{u}_1 \mathbf{u}_1^T \end{aligned}$$

Für die Funktion $\text{sign}()$ gilt hier

$$\text{sign}(x) = \begin{cases} +1 & \text{für } x \geq 0 \\ -1 & \text{für } x < 0 \end{cases}$$

Nun kann man nachrechnen (vgl. Bsp. 4.10), dass mit dieser Definition, die Matrixmultiplikation

$$\mathbf{H}_1 \cdot \mathbf{A}$$

tatsächlich das gewünschte Resultat bringt und die erste Spalte unterhalb der Diagonalen von \mathbf{A} zu Null wird.
Wir setzen nun

$$\mathbf{Q}_1 := \mathbf{H}_1$$

Diese Grundidee wird nun mehrfach hintereinander auf die Spalten von \mathbf{A} angewendet, um \mathbf{A} schrittweise auf die rechtsobere Dreiecksform zu bringen. So erhalten wir nach und nach die Faktoren \mathbf{Q}_i die wir brauchen, um die gesuchte Orthogonalmatrix

$$\mathbf{Q} = (\mathbf{Q}_4 \cdot \mathbf{Q}_3 \cdot \mathbf{Q}_2 \cdot \mathbf{Q}_1)^{-1}$$

zu berechnen, während \mathbf{A} schrittweise in die gesuchte rechtsobere Dreiecksmatrix \mathbf{R} transformiert wird.

2. Betrachten wir nun den zweiten Schritt. Es wird von $\mathbf{H}_1 \cdot \mathbf{A}$ die erste Zeile und die erste Spalte gestrichen und die so entstehende $(n-1) \times (n-1)$ Matrix \mathbf{A}_2 weiterbehandelt:

$$\mathbf{H}_1 \cdot \mathbf{A} = \left(\begin{array}{c|ccccc} * & * & * & * & * \\ \hline 0 & & & & & \\ 0 & & & & & \\ 0 & & & & & \\ 0 & & & & & \end{array} \right) \quad \mathbf{A}_2$$

Nun kann \mathbf{H}_2 analog zum ersten Schritt bestimmt werden, so dass

$$\mathbf{H}_2 \cdot \mathbf{A}_2 = \mathbf{H}_2 \cdot \underbrace{\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}}_{=\mathbf{A}_2} = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}$$

wobei

$$\begin{aligned}\mathbf{v}_2 &:= \tilde{\mathbf{a}}_1 + \text{sign}(\tilde{a}_{11}) \cdot |\tilde{\mathbf{a}}_1| \cdot \tilde{\mathbf{e}}_1 \\ \mathbf{u}_2 &:= \frac{1}{|\mathbf{v}_2|} \mathbf{v}_2 \\ \mathbf{H}_2 &:= \mathbf{I}_{n-1} - 2\mathbf{u}_2\mathbf{u}_2^T\end{aligned}$$

Aufgepasst, hier ist $\tilde{\mathbf{a}}_1$ nun die erste Spalte von \mathbf{A}_2 , \tilde{a}_{11} analog das erste Element von \mathbf{A}_2 und $\tilde{\mathbf{e}}_1 = (1, 0, 0, 0)^T$ der um ein Element verkürzte Einheitsvektor.

Da \mathbf{H}_2 nun aber eine $(n-1) \times (n-1)$ Matrix ist, definieren wir \mathbf{Q}_2 , welches eine $n \times n$ Matrix sein muss, als

$$\mathbf{Q}_2 = \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ \hline 0 & & & & \\ 0 & & \mathbf{H}_2 & & \\ 0 & & & & \\ 0 & & & & \end{array} \right)$$

Damit erhalten wir

$$\begin{aligned}\mathbf{Q}_2 \cdot \mathbf{Q}_1 \cdot \mathbf{A} &= \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ \hline 0 & & & & \\ 0 & & \mathbf{H}_2 & & \\ 0 & & & & \\ 0 & & & & \end{array} \right) \cdot \left(\begin{array}{c|cccc} * & * & * & * & * \\ \hline 0 & & & & \\ 0 & & \mathbf{A}_2 & & \\ 0 & & & & \\ 0 & & & & \end{array} \right) \\ &= \left(\begin{array}{c|cccc} * & * & * & * & * \\ \hline 0 & & & & \\ 0 & & \mathbf{H}_2 \cdot \mathbf{A}_2 & & \\ 0 & & & & \\ 0 & & & & \end{array} \right) \\ &= \left(\begin{array}{c|ccccc} * & * & * & * & * \\ \hline 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{array} \right)\end{aligned}$$

3. Analog definieren wir im dritten Schritt \mathbf{A}_3 mit

$$\mathbf{Q}_2 \cdot \mathbf{Q}_1 \cdot \mathbf{A} = \left(\begin{array}{c|ccccc} * & * & * & * & * & * \\ \hline 0 & * & * & * & * & * \\ 0 & 0 & & & & \\ \hline 0 & 0 & & & & \\ 0 & 0 & & & & \\ 0 & 0 & & & & \end{array} \right) \mathbf{A}_3$$

und berechen \mathbf{H}_3 . Wir definieren

$$\mathbf{Q}_3 = \left(\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & & & \\ \hline 0 & 0 & & & \\ 0 & 0 & & & \end{array} \right) \mathbf{H}_3$$

und erhalten

$$\mathbf{Q}_3 \cdot \mathbf{Q}_2 \cdot \mathbf{Q}_1 \cdot \mathbf{A} = \left(\begin{array}{c|ccccc} * & * & * & * & * & * \\ \hline 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{array} \right)$$

4. Im letzten Schritt ergibt sich schliesslich die gesuchte rechtsoberre Matrix \mathbf{R} mit

$$\underbrace{\mathbf{Q}_4 \cdot \mathbf{Q}_3 \cdot \mathbf{Q}_2 \cdot \mathbf{Q}_1}_{=\mathbf{Q}^{-1}} \cdot \mathbf{A} = \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{pmatrix} =: \mathbf{R}$$

sowie die für $\mathbf{A} = \mathbf{Q} \cdot \mathbf{R}$ gesuchte Orthogonalmatrix \mathbf{Q} als

$$\mathbf{Q} := (\mathbf{Q}_4 \cdot \mathbf{Q}_3 \cdot \mathbf{Q}_2 \cdot \mathbf{Q}_1)^{-1} = \mathbf{Q}_1^{-1} \cdot \mathbf{Q}_2^{-1} \cdot \mathbf{Q}_3^{-1} \cdot \mathbf{Q}_4^{-1} = \mathbf{Q}_1^T \cdot \mathbf{Q}_2^T \cdot \mathbf{Q}_3^T \cdot \mathbf{Q}_4^T$$

da die \mathbf{Q}_i orthogonal sind. Es müssen also keine Inversen berechnet werden.

Zusammenfassend lässt sich die Grundidee als Algorithmus formulieren:

Algorithmus zur QR -Zerlegung [1]:

- $\mathbf{R} := \mathbf{A}$
- $\mathbf{Q} := \mathbf{I}_n$
- Für $i = 1, \dots, n-1$:
 - erzeuge Nullen in \mathbf{R} in der i-ten Spalte unterhalb der Diagonalen:
 - bestimme die $(n-i+1) \times (n-i+1)$ Householder-Matrix \mathbf{H}_i
 - erweitere \mathbf{H}_i durch einen \mathbf{I}_{i-1} Block links oben zur $n \times n$ Matrix \mathbf{Q}_i
 - $\mathbf{R} := \mathbf{Q}_i \cdot \mathbf{R}$
 - $\mathbf{Q} := \mathbf{Q} \cdot \mathbf{Q}_i^T$
- Ausgabe: \mathbf{Q}, \mathbf{R}

Auch wenn sich dieser Algorithmus so implementieren lässt, ist er nicht effizient. Die spezielle Struktur der auftretenden Householder-Matrizen erlaubt Einsparungen an Rechenarbeit. So müssen die \mathbf{Q}_i nicht berechnet werden, es kann direkt mit den kleineren \mathbf{H}_i gearbeitet werden, von denen es ausreicht, die zugrunde liegenden Vektoren \mathbf{u}_i zu kennen. Trotzdem wollen wir den Algorithmus wie oben beschrieben einmal zur besseren Verständlichkeit einmal konkret durchrechnen. Für spätere konkrete Berechnungen, z.B. die Bestimmung von Eigenwerten (vgl. Kap. 4.8) werden wir ansonsten die effizientere Implementierung in Python benutzen.

Beispiel 4.10 [1]

- Gegeben ist das Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ mit

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -1 \\ 4 & -2 & 6 \\ 3 & 1 & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 9 \\ -4 \\ 9 \end{pmatrix}$$

Berechen Sie die QR -Zerlegung von \mathbf{A} und bestimmen Sie anhand von \mathbf{Q} und \mathbf{R} die Lösung \mathbf{x} (in Aufgabe 4.4 hatten wir dieses Gleichungssystem mit der LR -Zerlegung gelöst).

- Lösung:

Wir benutzen die erste Spalte \mathbf{a}_1 von \mathbf{A} und berechnen die folgenden Größen

$$\begin{aligned} \mathbf{v}_1 &:= \mathbf{a}_1 + \text{sign}(a_{11}) \cdot |a_{11}| \cdot \mathbf{e}_1 \\ \mathbf{u}_1 &:= \frac{1}{|\mathbf{v}_1|} \mathbf{v}_1 \\ \mathbf{H}_1 &:= \mathbf{I}_n - 2\mathbf{u}_1 \mathbf{u}_1^T \end{aligned}$$

also mit

$$\begin{aligned}
\mathbf{a}_1 &= \begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix} \\
\mathbf{v}_1 &= \begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix} + 1 \cdot \sqrt{1^2 + 4^2 + 3^2} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 6.099 \\ 4 \\ 3 \end{pmatrix} \\
\mathbf{u}_1 &= \frac{1}{7.8866} \begin{pmatrix} 6.099 \\ 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 0.7733 \\ 0.5072 \\ 0.3804 \end{pmatrix} \\
\mathbf{H}_1 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} 0.7733 \\ 0.5072 \\ 0.3804 \end{pmatrix} \begin{pmatrix} 0.7733 & 0.5072 & 0.3804 \end{pmatrix} \\
&= \begin{pmatrix} -0.1961 & -0.7845 & -0.5883 \\ -0.7845 & 0.4855 & -0.3859 \\ -0.5883 & -0.3859 & 0.7106 \end{pmatrix} =: \mathbf{Q}_1
\end{aligned}$$

Es folgt

$$\mathbf{Q}_1 \mathbf{A} = \begin{pmatrix} -5.0990 & 0.5883 & -4.5107 \\ 0 & -2.9258 & 3.6976 \\ 0 & 0.3056 & -1.7268 \end{pmatrix}$$

Im zweiten Schritt definieren wir \mathbf{A}_2 als den 2×2 Block rechts unten in $\mathbf{Q}_1 \mathbf{A}$, also

$$\mathbf{A}_2 = \begin{pmatrix} -2.9258 & 3.6976 \\ 0.3056 & -1.7268 \end{pmatrix}$$

und berechnen wieder die gleichen Größen, jetzt bezogen auf \mathbf{A}_2

$$\begin{aligned}
\tilde{\mathbf{a}}_1 &= \begin{pmatrix} -2.9258 \\ 0.3056 \end{pmatrix} \\
\mathbf{v}_2 &= \tilde{\mathbf{a}}_1 + \text{sign}(\tilde{a}_{11}) \cdot |\tilde{\mathbf{a}}_1| \cdot \tilde{\mathbf{e}}_1 = \begin{pmatrix} -2.9258 \\ 0.3056 \end{pmatrix} + (-1) \cdot \sqrt{(-2.9258)^2 + 0.3056^2} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} -5.8676 \\ 0.3056 \end{pmatrix} \\
\mathbf{u}_2 &= \frac{1}{|\mathbf{v}_2|} \mathbf{v}_2 = \frac{1}{5.8755} \begin{pmatrix} -5.8676 \\ 0.3056 \end{pmatrix} \\
&= \begin{pmatrix} -0.9986 \\ 0.0520 \end{pmatrix} \\
\mathbf{H}_2 &= \mathbf{I}_2 - 2\mathbf{u}_2\mathbf{u}_2^T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} -0.9986 \\ 0.0520 \end{pmatrix} \begin{pmatrix} -0.9986 & 0.0520 \end{pmatrix} \\
&= \begin{pmatrix} -0.9946 & 0.1039 \\ 0.1039 & 0.9946 \end{pmatrix} \\
\mathbf{Q}_2 &= \left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & \mathbf{H}_2 & \\ 0 & & \end{array} \right) = \left(\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & -0.9946 & 0.1039 \\ 0 & 0.1039 & 0.9946 \end{array} \right)
\end{aligned}$$

Damit haben wir das Ende der Schleife erreicht und erhalten als Ergebnis

$$\begin{aligned}
\mathbf{Q} &= \mathbf{Q}_1^T \cdot \mathbf{Q}_2^T = \begin{pmatrix} -0.1961 & 0.7191 & -0.6667 \\ -0.7845 & -0.5230 & -0.3333 \\ -0.5883 & 0.4576 & 0.6667 \end{pmatrix} \\
\mathbf{R} &= \mathbf{Q}_2 \mathbf{Q}_1 \mathbf{A} = \begin{pmatrix} -5.0990 & 0.5883 & -4.5107 \\ 0 & 2.9417 & -3.8570 \\ 0 & 0 & -1.3333 \end{pmatrix}
\end{aligned}$$

Wie gewünscht gilt

$$QR = \begin{pmatrix} 1 & 2 & -1 \\ 4 & -2 & 6 \\ 3 & 1 & 0 \end{pmatrix} = A$$

Zum Abschluss müssen wir noch das Gleichungssystem $Ax = b$ lösen:

$$Ax = b \iff QRx = b \iff Rx = Q^T b$$

also

$$\underbrace{\begin{pmatrix} -5.0990 & 0.5883 & -4.5107 \\ 0 & 2.9417 & -3.8570 \\ 0 & 0 & -1.3333 \end{pmatrix}}_R \underbrace{x}_{\mathbf{x}} = \underbrace{\begin{pmatrix} -0.1961 & -0.7845 & -0.5883 \\ 0.7191 & -0.5230 & 0.4576 \\ -0.6667 & -0.3333 & 0.6667 \end{pmatrix}}_{Q^T} \underbrace{\begin{pmatrix} 9 \\ -4 \\ 9 \end{pmatrix}}_b = \begin{pmatrix} -3.9223 \\ 12.6822 \\ 1.3333 \end{pmatrix}$$

Rückwärtseinsetzen liefert die Lösung $\mathbf{x} = (2, 3, -1)^T$.

4.6 Fehlerrechnung und Aufwandabschätzung

4.6.1 Fehlerrechnung bei linearen Gleichungssystemen

Wie bereits in Kapitel 2 ausgeführt, können Computer nicht alle reellen Zahlen darstellen, weswegen alle Zahlen intern gerundet werden, damit sie in die endliche Menge der maschinendarstellbaren Zahlen passen. Hierdurch entstehen Rundungsfehler. Selbst wenn sowohl die Eingabewerte als auch das Ergebnis eines Algorithmus maschinendarstellbare Zahlen sind, können solche Fehler auftreten, denn auch die (möglicherweise nicht darstellbaren) Zwischenergebnisse eines Algorithmus werden gerundet. Aufgrund von diesen Fehlern aber auch wegen Eingabe- bzw. Messfehlern in den vorliegenden Daten oder Fehlern aus vorhergehenden numerischen Rechnungen, wird durch einen Algorithmus üblicherweise nicht die exakte Lösung \mathbf{x} des linearen Gleichungssystems

$$Ax = b$$

berechnet, sondern eine Näherungslösung $\tilde{\mathbf{x}}$. Um dies formal zu fassen, führt man ein "benachbartes" oder "gestörtes" Gleichungssystems

$$A\tilde{\mathbf{x}} = \tilde{\mathbf{b}} = \mathbf{b} + \Delta\mathbf{b}$$

ein, für das $\tilde{\mathbf{x}}$ gerade die exakte Lösung ist. Dabei ist $\Delta\mathbf{b}$ das *Residuum* oder der *Defekt* der Näherungslösung $\tilde{\mathbf{x}}$. Den Vektor $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$ nennen wir den Fehler der Näherungslösung $\tilde{\mathbf{x}}$. Da Rundung und andere Fehlerquellen i.A. nur kleine Fehler bewirken, ist es gerechtfertigt anzunehmen, dass der noch zu definierende 'Betrag' $\|\Delta\mathbf{b}\|$ 'klein' ist.

Das Ziel dieses Abschnittes ist es nun, aus der Grösse des Residuum $\|\Delta\mathbf{b}\|$ auf die Grösse des Fehlers $\|\Delta\mathbf{x}\|$ zu schließen. Insbesondere wollen wir untersuchen, wie sensibel die Grösse $\|\tilde{\mathbf{x}}\|$ von $\|\Delta\mathbf{b}\|$ abhängt, d.h. ob kleine Residuen $\|\Delta\mathbf{b}\|$ große Fehler in $\|\tilde{\mathbf{x}}\|$ hervorrufen können. Diese Analyse ist unabhängig von dem verwendeten Lösungsverfahren, da wir hier nur das Gleichungssystem selbst und kein explizites Verfahren betrachten. Um diese Analyse durchzuführen, brauchen wir das Konzept der Norm.

Definition 4.4: Vektornorm [1]

Eine Abbildung $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ heisst Vektornorm, wenn die folgenden Bedingungen für alle $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$ erfüllt sind:

- $\|\mathbf{x}\| \geq 0$ und $\|\mathbf{x}\| = 0 \iff \mathbf{x} = 0$
- $\|\lambda\mathbf{x}\| = |\lambda| \|\mathbf{x}\|$
- $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ "Dreiecksungleichung"

Die drei gebräuchlichsten Vektornormen sind die folgenden:

Definition 4.5: Vektornormen / Matrixnormen [1]

- Für Vektoren $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ gibt es die folgenden Vektornormen:

$$\begin{aligned} \text{1-Norm, Summennorm} &: \|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i| \\ \text{2-Norm, euklidische Norm} &: \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2} \\ \infty\text{-Norm, Maximumnorm} &: \|\mathbf{x}\|_\infty = \max_{i=1,\dots,n} |x_i| \end{aligned}$$

- Für eine $n \times n$ Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ sind mit den Vektornormen die folgenden Matrixnormen verbunden, welche die Eigenschaften der Definition 4.4 ebenfalls erfüllen:

$$\begin{aligned} \text{1-Norm, Spaltensummennorm} &: \|\mathbf{A}\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^n |a_{ij}| \\ \text{2-Norm, Spektralnorm} &: \|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^T \mathbf{A})} \\ \infty\text{-Norm, Zeilensummennorm} &: \|\mathbf{A}\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^n |a_{ij}| \end{aligned}$$

Bemerkungen:

- Die euklidische Norm entspricht dem herkömmlichen Verständnis der Länge eines Vektors, die beiden anderen Vektornormen sind aber im Zusammenhang mit Matrixoperationen einfacher berechenbar.
 - Die Vektor- und zugehörige Matrixnorm sind verträglich bzw. kompatibel zueinander, da für alle $\mathbf{A} \in \mathbb{R}^{n \times n}$ und $\mathbf{x} \in \mathbb{R}^n$ gilt
- $$\|\mathbf{Ax}\|_i \leq \|\mathbf{A}\|_i \cdot \|\mathbf{x}\|_i \quad (i = 1, 2, \infty)$$
- Die Spektralnorm verwendet den Spektralradius $\rho()$, welchen wir in Def. 4.18 genauer kennenlernen werden. Wir fokussieren hier auf die 1- und ∞ -Norm.

Beispiel 4.11 [1]:

- Berechnen Sie die 1-, 2-, und ∞ -Norm des Vektors $\begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}$ sowie die 1- und ∞ -Norm von $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & -2 \\ 7 & -3 & 5 \end{pmatrix}$.

Lösung:

$$\begin{aligned}\left\| \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix} \right\|_1 &= 1 + 2 + 3 = 6, \quad \left\| \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix} \right\|_2 = \sqrt{1 + 2^2 + 3^2} = \sqrt{14}, \\ \left\| \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix} \right\|_\infty &= \max\{1, 2, 3\} = 3 \\ \left\| \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & -2 \\ 7 & -3 & 5 \end{pmatrix} \right\|_1 &= \max\{1 + 3 + 7, 2 + 4 + 3, 3 + 2 + 5\} = 11 \\ \left\| \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & -2 \\ 7 & -3 & 5 \end{pmatrix} \right\|_\infty &= \max\{1 + 2 + 3, 3 + 4 + 2, 7 + 3 + 5\} = 15.\end{aligned}$$

Für die Fehlerabschätzung von $\tilde{\mathbf{x}}$ und $\tilde{\mathbf{b}}$ gilt der folgende

Satz 4.2: Abschätzung für fehlerbehaftete Vektoren [1]

- Sei $\|\cdot\|$ eine Norm, $\mathbf{A} \in \mathbb{R}^{n \times n}$ eine reguläre $n \times n$ Matrix und $\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{b}, \tilde{\mathbf{b}} \in \mathbb{R}^n$ mit $\mathbf{A}\mathbf{x} = \mathbf{b}$ und $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$. Dann gilt für den absoluten und den relativen Fehler in \mathbf{x} :
 - $\|\mathbf{x} - \tilde{\mathbf{x}}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{b} - \tilde{\mathbf{b}}\|$
 - $\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \cdot \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|}{\|\mathbf{b}\|}$ falls $\|\mathbf{b}\| \neq 0$
- Die Zahl $\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$ nennt man Konditionszahl der Matrix \mathbf{A} bzgl. der verwendeten Norm.

Bemerkungen:

- Für Matrizen, deren Kondition $\text{cond}(\mathbf{A})$ groß ist, können sich kleine Fehler im Vektor \mathbf{b} (bzw. Rundungsfehler im Verfahren) zu großen Fehlern im Ergebnis \mathbf{x} verstärken. Man spricht in diesem Fall von schlecht konditionierten Matrizen.

Beispiel 4.12 [1]:

- Untersuchen Sie die Fehlerfortpflanzung im linearen Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ mit

$$\mathbf{A} = \begin{pmatrix} 2 & 4 \\ 4 & 8.1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1.5 \end{pmatrix}$$

für den Fall, dass die rechte Seite von $\tilde{\mathbf{b}}$ in jeder Komponente um maximal 0.1 von \mathbf{b} abweicht.

- Lösung: Wir betrachten das System $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, wobei $\tilde{\mathbf{b}}$ maximal um 0.1 von jeder Komponente von \mathbf{b} abweicht. Zuerst müssen wir eine der möglichen Norm wählen. Hierfür ist die ∞ -Norm besonders geeignet, da wir schreiben können $\|\tilde{\mathbf{b}} - \mathbf{b}\|_\infty \leq 0.1$. Zusätzlich haben wir $\|\mathbf{A}\|_\infty = 12.1$ und mit

$$\mathbf{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} = \frac{1}{2 \cdot 8.1 - 4 \cdot 4} \begin{pmatrix} 8.1 & -4 \\ -4 & 2 \end{pmatrix}$$

erhalten wir $\|\mathbf{A}^{-1}\|_\infty = \frac{12.1}{0.2} = 60.5$ und für die Konditionszahl $\text{cond}(\mathbf{A})$ erhalten wir in der ∞ -Norm $\text{cond}(\mathbf{A})_\infty = \|\mathbf{A}\|_\infty \|\mathbf{A}^{-1}\|_\infty = 12.1 \cdot 60.5 = 732.05$. Mit dem obigen Satz gilt also

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty \leq \|\mathbf{A}^{-1}\|_\infty \cdot \|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty \leq 60.5 \cdot 0.1 = 6.05$$

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty}{\|\mathbf{x}\|_\infty} \leq \text{cond}(\mathbf{A})_\infty \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty}{\|\mathbf{b}\|_\infty} \leq 732 \cdot \frac{0.1}{1.5} = 48.8$$

Wie ist dies nun zu interpretieren? Die Lösung $\tilde{\mathbf{x}}$ des gestörten Systems $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ wird also von der Lösung \mathbf{x} des exakten Systems $\mathbf{Ax} = \mathbf{b}$ in jeder Komponente um maximal 6.05 abweichen (absoluter Fehler), und der relative Fehler wird maximal 48.8 betragen. Testen wir das an einem konkreten Beispiel.

Beispiel 4.13 [1]:

- Betrachten Sie obiges Beispiel und nehmen sie für die gestörte rechte Seite $\tilde{\mathbf{b}} = \begin{pmatrix} 0.9 \\ 1.6 \end{pmatrix}$. Berechnen sie die Lösungen von $\mathbf{Ax} = \mathbf{b}$ und $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$. Berechnen Sie anschliessend den absoluten Fehler $\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty$ und den relativen Fehler $\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty}{\|\mathbf{x}\|_\infty}$. Vergleichen Sie mit $\|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty$ und $\frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty}{\|\mathbf{b}\|_\infty}$.
- Lösung: Wir erhalten $\mathbf{x} = \begin{pmatrix} 10.5 \\ -5 \end{pmatrix}$ und $\tilde{\mathbf{x}} = \begin{pmatrix} 4.45 \\ -2 \end{pmatrix}$. Mit $\|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty = 0.1$ und $\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty = 6.05$ sehen wir, dass der absolute Fehler um den maximal möglichen Faktor 60.5 verstärkt worden ist. Mit $\frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty}{\|\mathbf{b}\|_\infty} = \frac{0.1}{1.5} = 0.0667$ und $\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty}{\|\mathbf{x}\|_\infty} = \frac{6.05}{10.5} = 0.5762$ wurde der relative Fehler um einen Faktor 8.6 verstärkt, weniger als der maximal mögliche Faktor von 732.

Wir waren bisher davon ausgegangen, dass die Matrix \mathbf{A} selbst exakt ist. Wie verhält sich die Fehlerabschätzung nun unter der Annahme, dass auch noch \mathbf{A} fehlerbehaftet ist, wir es also mit einem Gleichungssystem

$$\tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}} = \tilde{\mathbf{b}}$$

zu tun haben? Dafür gilt die folgende Fehlerabschätzung

Satz 4.3: Abschätzung für fehlerbehaftete Matrix [1]

Sei $\|\cdot\|$ eine Norm, $\mathbf{A}, \tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$ reguläre $n \times n$ Matrizen und $\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{b}, \tilde{\mathbf{b}} \in \mathbb{R}^n$ mit $\mathbf{Ax} = \mathbf{b}$ und $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$. Falls

$$\text{cond}(\mathbf{A}) \cdot \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|}{\|\mathbf{A}\|} < 1$$

dann gilt:

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \frac{\text{cond}(\mathbf{A})}{1 - \text{cond}(\mathbf{A}) \cdot \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|}{\|\mathbf{A}\|}} \cdot \left(\frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|}{\|\mathbf{A}\|} + \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|}{\|\mathbf{b}\|} \right)$$

Bemerkung:

- Für den Fall, dass \mathbf{A} exakt gegeben ist, gilt $\frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|}{\|\mathbf{A}\|} = 0$ und der relative Fehler für \mathbf{x} aus Satz 4.3 reduziert sich auf den relativen Fehler in Satz 4.2.

Beispiel 4.14 [1]:

- Nehmen Sie noch einmal das Beispiel 4.12 und untersuchen Sie die Fehlerfortpflanzung unter der zusätzlichen Annahme, dass die Matrix \mathbf{A} um maximal 0.003 elementweise gestört ist.
- Lösung: Wir hatten bereits die folgenden Größen berechnet

$$\|\mathbf{A}\|_\infty = 12.1, \text{cond}(\mathbf{A}) = 732.05, \|\mathbf{b}\|_\infty = 1.5, \|\mathbf{b} - \tilde{\mathbf{b}}\|_\infty \leq 0.1$$

Wenn nun jedes Element von \mathbf{A} um maximal 0.003 gestört wird, summiert sich diese Störung in der ∞ -Norm auf und wir erhalten $\|\mathbf{A} - \tilde{\mathbf{A}}\|_\infty \leq 0.006$ und damit

$$\text{cond}(\mathbf{A}) \cdot \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_\infty}{\|\mathbf{A}\|_\infty} \leq 0.363 < 1.$$

Wir können also die Abschätzung aus Satz 4.3 anwenden und erhalten

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_\infty}{\|\mathbf{x}\|_\infty} \leq \frac{732.05}{1 - 0.363} \left(\frac{0.006}{12.1} + \frac{0.1}{1.5} \right) \leq 77.2$$

4.6.2 Aufwandabschätzung¹¹

Ein wichtiger Aspekt bei der Analyse numerischer Verfahren ist die Abschätzung, wie viel Aufwand diese Verfahren in der Regel benötigen, um zu dem gewünschten Ergebnis zu kommen. Dies hängt entscheidend von der Leistungsfähigkeit des verwendeten Computers ab. Deshalb wird nicht direkt die Zeit abgeschätzt, sondern vielmehr die Anzahl der Rechenoperationen, die ein Algorithmus benötigt. Da hierbei die Gleitkommaoperationen, also Addition, Multiplikation etc. von reellen Zahlen, die mit Abstand zeitintensivsten Operationen sind, beschränkt man sich in der Analyse üblicherweise auf diese.

Bisher haben wir nur direkte Verfahren angeschaut, welche nach einer endlichen Anzahl von Rechenschritten die 'exakte' Lösung liefern. Natürlich hängt hierbei die Anzahl Schritte von deren Dimension n der Matrix \mathbf{A} ab. Es genügt also, die Anzahl der dafür benötigten Gleitkommaoperationen in Abhängigkeit von n zu bestimmen. Dafür benötigt man die Gleichungen

$$\sum_{i=1}^n i = \frac{(n+1)n}{2} \text{ und } \sum_{i=1}^n i^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n.$$

Beispiel 4.15:

- Wie viele Gleitkommaoperationen benötigt das Rückwärtseinsetzen gemäss Gleichung 4.5?
- Lösung: Für jedes x_i haben wir

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}}, \quad i = n, n-1, \dots, 1.$$

Betrachten wir zuerst nur die Multiplikationen und Divisionen. Für $i = n$ haben wir eine Division, für $i = n-1$ haben wir eine Multiplikation und eine Division, etc. Für $i = 1$ schliesslich haben wir $n-1$ Multiplikationen und eine Division. Das ergibt für diese Operationen also

$$1 + 2 + 3 + \dots + n = \sum_{i=1}^n i = \frac{(n+1)n}{2}.$$

Nun schauen wir uns noch die Anzahl Additionen und Subtraktionen an. Für $i = n$ haben wir keine, für $i = n-1$ haben wir eine Subtraktion, etc., das ergibt dann

$$0 + 1 + 2 + \dots + n-1 = \sum_{i=1}^{n-1} i = \frac{(n-1+1)(n-1)}{2} = \frac{n(n-1)}{2}.$$

Für die Summe beider Operationstypen erhalten wir also

$$\frac{n^2}{2} + \frac{n}{2} + \frac{n^2}{2} - \frac{n}{2} = n^2.$$

Für das Vorwärtseinsetzen erhalten wir natürlich das gleiche Resultat.

Für die Gauss-Elimination erhält man nach einer ähnlichen Betrachtung die Anzahl Gleitkommaoperationen (ohne Pivotisierung) zu

$$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{13}{6}n.$$

Die Anzahl Operationen für die LR-Zerlegung ist identisch, wenn sie mit der Gauss-Elimination durchgeführt wurde. Für die QR-Zerlegung erhält man (ohne Beweis)

$$\frac{5}{3}n^3 + 3n^2 + \frac{7}{3}n - 7.$$

¹¹Kapitel hauptsächlich übernommen aus [3]

Für die vollständige Lösung eines linearen Gleichungssystems müssen nun die Operationen für Rückwärtseinsetzen (Gauss und QR-Zerlegung) bzw. Rückwärts- und Vorwärtseinsetzen (LR-Zerlegung) noch addiert werden. Für die Gauss-Elimination erhalten wir dann

$$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{13}{6}n + n^2 = \frac{2}{3}n^3 + \frac{5}{2}n^2 - \frac{13}{6}n,$$

für die LR-Zerlegung

$$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{13}{6}n + 2n^2 = \frac{2}{3}n^3 + \frac{7}{2}n^2 - \frac{13}{6}n,$$

und für die QR-Zerlegung

$$\frac{5}{3}n^3 + 3n^2 + \frac{7}{3}n - 7 + n^2 = \frac{5}{3}n^3 + 4n^2 + \frac{7}{3}n - 7$$

Operationen. Berücksichtigt man, dass für große n die “ n^3 -Terme” dominant werden, so ergibt sich, dass die QR-Zerlegung etwas mehr als doppelt so viel Zeit beansprucht wie die Gauß-Elimination bzw. die LR-Zerlegung.

Im Vergleich dazu müssen bei der Cramerschen Regel $n+1$ Determinanten und n Quotienten bestimmt werden, was für jede Determinante mit der Regel von Leibniz $(n-1) \cdot n!$ Multiplikationen und $n! - 1$ Additionen beinhaltet. Das ergibt also

$$(n+1)((n-1) \cdot n! + n! - 1) + n = n(n+1)! - 1$$

Punktoperationen.

Um einen Eindruck von den tatsächlichen Rechenzeiten zu bekommen, nehmen wir an, dass wir einen handelsüblichen PC verwenden, der mit einer 3.5GHz CPU mit 6 Kernen ausgestattet ist mit einer tatsächlichen Leistung von 100 GFLOPS (FLOPS = floating point operations per second), d.h. mit 10^{11} Gleitkommaoperationen pro Sekunde (zum Vergleich: die Zuse Z3 schaffte mit einer Taktrate von 5.3 Hz rund 1 FLOPS). Nehmen wir weiterhin an, dass wir Implementierungen der obigen Algorithmen haben, die diese Leistung optimal ausnutzen. Dann ergeben sich für $n \times n$ Gleichungssysteme die folgenden (ungefähren) Rechenzeiten

n	Gauss	LR-Zerlegung	QR-Zerlegung	Cramer
10^1	9 ns	9 ns	20 ns	4 ms
10^2	7 μ s	7 μ s	17 μ s	$3 \cdot 10^{143}$ y
10^3	7 ms	7 ms	17 ms	—
10^4	7 s	7 s	17 s	—
10^5	2 h	2 h	5 h	—
10^6	77 d	77 d	193 d	—
10^7	211 y	211 y	528 y	—

Wie man sieht, wächst die benötigte Zeit für den Gauß-Algorithmus, die LR-Zerlegung und die QR-Zerlegung um einen Faktor $10^3 = 1000$, wenn n um einen Faktor 10 erhöht wird. Für die Cramersche Regel benötigt man für $n = 10$ ’erst’ 4 Millisekunden, für $n = 20$ bereits rund 324 Jahre, für $n = 25$ bereits 3.2 Mia. Jahre und für $n = 100$ läppische $3 \cdot 10^{143}$ Jahre. Ein eindrückliches Beispiel, wie schnell die Fakultät wächst.

Ab $n > 10^5$ kommt aber auch für die anderen Algorithmen die Wartezeit in einen Bereich, der kaum mehr akzeptabel ist. Im nächsten Abschnitt werden wir deshalb die iterativen Verfahren kennen lernen, die zwar nicht mehr die ’exakte’ Lösung berechnen, dafür aber wesentlich schneller sind.

Zum Abschluss dieses Abschnitts wollen wir aber noch ein etwas gröberes Konzept zur Aufwandsabschätzung betrachten, welches für praktische Anwendungen häufig ausreicht. Man interessiert sich dabei nicht für die genaue Anzahl durchzuführender Operationen sondern nur für eine Abschätzung bei grossen Dimensionen, d.h. wie der Aufwand sich asymptotisch für $n \rightarrow \infty$ verhält. Dabei spricht man von der Ordnung eines Algorithmus:

Definition 4.6: Ordnung [3]

- Ein Algorithmus hat die Ordnung $O(n^q)$, wenn $q > 0$ die minimale Zahl ist, für die es eine Konstante $C > 0$ gibt, so dass der Algorithmus für alle $n \in N$ weniger als Cn^q Operationen benötigt.

Die Zahl q ist aus den obigen Aufwandsabschätzungen einfach abzulesen. Sie entspricht gerade der höchsten auftretenden Potenzen von n . Daraus folgt, dass Vor- und Rückwärtseinsetzen sind von der Ordnung $O(n^2)$, während das Gauß-Verfahren sowie die LR- und QR-Zerlegung von der Ordnung $O(n^3)$ sind.

4.7 Iterative Verfahren

Wir haben bereits gesehen, dass die bisher betrachteten direkten Verfahren die Ordnung $O(n^3)$ besitzen. Für große Gleichungssysteme, die in der Praxis durchaus auftreten, führt dies wie oben gesehen zu unakzeptabel hohen Rechenzeiten. Eine Klasse von Verfahren, die eine niedrigere Ordnung hat, sind die iterativen Verfahren. Allerdings zahlt man für den geringeren Aufwand einen Preis: Man kann bei diesen Verfahren nicht mehr erwarten, eine (bis auf Rundungsfehler) exakte Lösung zu erhalten, sondern muss von vornherein eine gewisse Ungenauigkeit im Ergebnis in Kauf nehmen.

Das Grundprinzip iterativer Verfahren funktioniert dabei wie folgt: Ausgehend von einem Startvektor $\mathbf{x}^{(0)}$ berechnet man mittels einer Rechenvorschrift

$$F : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

iterativ eine Folge von Vektoren

$$\mathbf{x}^{(k+1)} = F(\mathbf{x}^{(k)}) \text{ mit } k = 0, 1, 2, \dots$$

die für $k \rightarrow \infty$ gegen die Lösung \mathbf{x} des Gleichungssystems $\mathbf{A}\mathbf{x} = \mathbf{b}$ konvergieren. Wenn die gewünschte Genauigkeit erreicht ist, wird die Iteration abgebrochen und der letzte Wert $\mathbf{x}^{(i)}$ als Näherung des Ergebnisses verwendet.

- Bemerkung zur Notation: ein hochgestellter Index in Klammern $\mathbf{x}^{(k)}$ bezeichnet einen Vektor aus \mathbb{R}^n nach der k -ten Iteration. Die Elemente des Vektors $\mathbf{x}^{(k)}$ werden wie üblich mit einem tiefgestellten Index bezeichnet, z.B. ist also $x_i^{(k)}$ das i -te Element des Vektors, bzw.

$$\mathbf{x}^{(k)} = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_{n-1}^{(k)} \\ x_n^{(k)} \end{pmatrix}$$

Wir wollen nun versuchen, das obige Problem als Fixpunktiteration zu behandeln. Wir hatten in Kapitel 3 gesehen, dass die allgemeine Fixpunktgleichung die Form $F(x) = x$ hat, d.h. wir wollen die ursprüngliche Gleichung $\mathbf{A}\mathbf{x} = \mathbf{b}$ in eine ähnliche Form bringen. Dies gelingt uns, wenn wir die Matrix \mathbf{A} zerlegen können in eine Form

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{R},$$

wobei \mathbf{L} eine untere Dreiecksmatrix sein soll (mit $l_{ii} = 0$), \mathbf{D} eine Diagonalmatrix und \mathbf{R} eine obere Dreiecksmatrix (mit $r_{ii} = 0$). Die einfachste Form ist

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{R}$$

$$= \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

Achtung: diese Matrizen \mathbf{L} und \mathbf{R} hier sind nicht die gleichen wie die \mathbf{LR} -Zerlegung aus Kap. 4.5.1.

4.7.1 Das Jacobi-Verfahren

Wir können mit obiger Zerlegung dann die folgende Fixpunktiteration, die auch als Jacobi- oder Gesamtschritt-Verfahren bekannt ist, durchführen:

Definition 4.7: Jacobi- bzw. Gesamtschrittverfahren [1]

- Zu lösen sei $\mathbf{A}\mathbf{x} = \mathbf{b}$. Die Matrix $\mathbf{A} = (a_{ij})$ sei zerlegt in der Form

$$\mathbf{A} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn-1} & 0 \end{pmatrix}}_{=: \mathbf{L}} + \underbrace{\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix}}_{=: \mathbf{D}}$$

$$+ \underbrace{\begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n-1,n} \\ 0 & 0 & & & 0 \end{pmatrix}}_{=: \mathbf{R}}$$

Dann heisst die Fixpunktiteration

$$\begin{aligned} \mathbf{D}\mathbf{x}^{(k+1)} &= -(\mathbf{L} + \mathbf{R})\mathbf{x}^{(k)} + \mathbf{b} \text{ bzw.} \\ \mathbf{x}^{(k+1)} &= -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{R})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b} \end{aligned}$$

Gesamtschrittverfahren oder **Jacobi-Verfahren**.

Dabei ist für \mathbf{D} die inverse \mathbf{D}^{-1} sehr einfach zu berechnen, auf der Diagonalen von \mathbf{D}^{-1} stehen einfach die Kehrwerte der Diagonalen von \mathbf{D} , also

$$\mathbf{D} = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix} \Rightarrow \mathbf{D}^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{a_{33}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \frac{1}{a_{nn}} \end{pmatrix}$$

Die Herleitung des Jacobi-Verfahrens folgt direkt aus der Zerlegung von \mathbf{A} :

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ (\mathbf{L} + \mathbf{D} + \mathbf{R})\mathbf{x} &= \mathbf{b} \\ (\mathbf{L} + \mathbf{R})\mathbf{x} + \mathbf{D}\mathbf{x} &= \mathbf{b} \\ \mathbf{D}\mathbf{x} &= -(\mathbf{L} + \mathbf{R})\mathbf{x} + \mathbf{b} \\ \mathbf{x} &= -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{R})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b} \equiv F(\mathbf{x}) \end{aligned}$$

womit wir die Fixpunktgleichung bereits aufgestellt haben.

Beispiel 4.16 [1]:

- Wenden Sie das Jacobi-Verfahren auf das folgende System an:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \text{ mit } \mathbf{A} = \begin{pmatrix} 4 & -1 & 1 \\ -2 & 5 & 1 \\ 1 & -2 & 5 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 5 \\ 11 \\ 12 \end{pmatrix}$$

Lösung: Mit den Bezeichnungen aus (3.7) haben wir:

$$\mathbf{L} = \begin{pmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 1 & -2 & 0 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Die Iteration lautet somit:

$$\begin{aligned} \mathbf{x}^{(n+1)} &= -\mathbf{D}^{-1}((\mathbf{L} + \mathbf{R}) \mathbf{x}^{(n)} - \mathbf{b}) \\ &= -\begin{pmatrix} 0.25 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{pmatrix} \left(\begin{pmatrix} 0 & -1 & 1 \\ -2 & 0 & 1 \\ 1 & -2 & 0 \end{pmatrix} \mathbf{x}^{(n)} - \begin{pmatrix} 5 \\ 11 \\ 12 \end{pmatrix} \right) \\ &= \begin{pmatrix} 0 & 0.25 & -0.25 \\ 0.4 & 0 & -0.2 \\ -0.2 & 0.4 & 0 \end{pmatrix} \mathbf{x}^{(n)} + \begin{pmatrix} 1.25 \\ 2.2 \\ 2.4 \end{pmatrix} \end{aligned}$$

Wir wählen als Startvektor den Nullvektor und erhalten:

i	0	1	2	3	4	5
$\mathbf{x}^{(i)}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1.25 \\ 2.2 \\ 2.4 \end{pmatrix}$	$\begin{pmatrix} 1.2 \\ 2.22 \\ 3.03 \end{pmatrix}$	$\begin{pmatrix} 1.0475 \\ 2.074 \\ 3.048 \end{pmatrix}$	$\begin{pmatrix} 1.0065 \\ 2.0094 \\ 3.0201 \end{pmatrix}$	$\begin{pmatrix} 0.997325 \\ 1.99858 \\ 3.00246 \end{pmatrix}$

Es sieht so aus, als konvergiere diese Folge gegen $(1, 2, 3)^T$, was übrigens die Lösung des Systems $\mathbf{A} \mathbf{x} = \mathbf{b}$ darstellt. ■

Üblicherweise wird die Iteration programmiertechnisch nicht mit Matrixmultiplikation durchgeführt, sondern für jede Komponente des Vektors \mathbf{x} separat, für obiges Beispiel wäre das also

$$x_1^{(k+1)} = 0.25x_2^{(k)} - 0.25x_3^{(k)} + 1.25 \quad (4.9)$$

$$x_2^{(k+1)} = 0.4x_1^{(k)} - 0.2x_3^{(k)} + 2.2 \quad (4.10)$$

$$x_3^{(k+1)} = -0.2x_1^{(k)} + 0.4x_2^{(k)} + 2.4 \quad (4.11)$$

und in der allgemeinen Form (zur einfacheren Implementation) können wir das schreiben als

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right) \quad i = 1, \dots, n.$$

4.7.2 Das Gauss-Seidel-Verfahren

Wenn man nun davon ausgeht, dass nach der k -ten Iteration der Vektor $\mathbf{x}^{(k+1)}$ komponentenweise näher an der Lösung liegt als der Vektor vom vorherigen Iterationsschritt $\mathbf{x}^{(k)}$, dann ist es im obigen Beispiel vermutlich genauer, die gerade berechnete Komponente $x_1^{(k+1)}$ aus Gleichung (4.9) in die noch zu berechnende Komponente $x_2^{(k+1)}$ in Gleichung (4.10) einzusetzen. Analog setzt man anschliessend die Komponenten $x_1^{(k+1)}$ und $x_2^{(k+1)}$ in die Gleichung (4.11) ein, um $x_3^{(k+1)}$ zu erhalten. Dies führt dann auf die Iteration

$$x_1^{(k+1)} = 0.25x_2^{(k)} - 0.25x_3^{(k)} + 1.25$$

$$x_2^{(k+1)} = 0.4x_1^{(k+1)} - 0.2x_3^{(k)} + 2.2$$

$$x_3^{(k+1)} = -0.2x_1^{(k+1)} + 0.4x_2^{(k+1)} + 2.4$$

welche man in Matrix-Form schreiben kann als

$$\mathbf{x}^{(k+1)} = \begin{pmatrix} 0 & 0.25 & -0.25 \\ 0 & 0 & -0.2 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{x}^{(k)} + \begin{pmatrix} 0 & 0 & 0 \\ 0.4 & 0 & 0 \\ -0.2 & 0.4 & 0 \end{pmatrix} \mathbf{x}^{(k+1)} + \begin{pmatrix} 1.25 \\ 2.2 \\ 2.4 \end{pmatrix}.$$

Mit unseren Matrizen \mathbf{L} , \mathbf{D} , und \mathbf{R} wird das zu

$$\mathbf{x}^{(k+1)} = \mathbf{D}^{-1} (\mathbf{b} - \mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{R}\mathbf{x}^{(k)})$$

oder in der allgemeinen Form

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right) \quad i = 1, \dots, n. \quad (4.12)$$

Jetzt kann man das noch umformen, so dass alle Terme mit $\mathbf{x}^{(k+1)}$ auf der linken Seite erscheinen, und erhält damit das sogenannte Gauss-Seidel-Verfahren oder auch Einzelschrittverfahren.

Definition 4.8: Gauss-Seidel bzw. Einzelschrittverfahren [1]

- Zu lösen sei $\mathbf{Ax} = \mathbf{b}$. Die Matrix $\mathbf{A} = (a_{ij})$ sei zerlegt in der Form

$$\mathbf{A} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn-1} & 0 \end{pmatrix}}_{=: \mathbf{L}} + \underbrace{\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix}}_{=: \mathbf{D}}$$

$$+ \underbrace{\begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}}_{=: \mathbf{R}}$$

Dann heisst die Fixpunktiteration

$$\begin{aligned} (\mathbf{D} + \mathbf{L})\mathbf{x}^{(k+1)} &= -\mathbf{R}\mathbf{x}^{(k)} + \mathbf{b} \text{ bzw.} \\ \mathbf{x}^{(k+1)} &= -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{R}\mathbf{x}^{(k)} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b} \end{aligned}$$

Einzelschrittverfahren oder **Gauss-Seidel-Verfahren**.

Beispiel 4.17 [1]:

- Wenden Sie das Gauss-Seidel-Verfahren auf das System aus Beispiel 4.16 an, also:

$$\mathbf{Ax} = \mathbf{b} \text{ mit } \mathbf{A} = \begin{pmatrix} 4 & -1 & 1 \\ -2 & 5 & 1 \\ 1 & -2 & 5 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 5 \\ 11 \\ 12 \end{pmatrix}$$

- Lösung: Wir verwenden die allgemeine Gleichung für die Komponenten gemäss (4.12) und erhalten

$$\begin{aligned}
x_1^{(k+1)} &= \frac{1}{a_{11}} \left(b_1 - \sum_{j=1}^0 a_{1j} x_j^{(k+1)} - \sum_{j=2}^3 a_{1j} x_j^{(k)} \right) \\
&= \frac{1}{4} (5 - (-1)x_2^{(k)} - 1x_3^{(k)}) = 1.25 + 0.25x_2^{(k)} - 0.25x_3^{(k)} \\
x_2^{(k+1)} &= \frac{1}{a_{22}} \left(b_2 - \sum_{j=1}^1 a_{2j} x_j^{(k+1)} - \sum_{j=3}^3 a_{2j} x_j^{(k)} \right) \\
&= \frac{1}{5} (11 - (-2)x_1^{(k+1)} - 1x_3^{(k)}) = 2.2 + 0.4x_1^{(k+1)} - 0.2x_3^{(k)} \\
x_3^{(k+1)} &= \frac{1}{a_{33}} \left(b_3 - \sum_{j=1}^2 a_{3j} x_j^{(k+1)} - \sum_{j=4}^3 a_{3j} x_j^{(k)} \right) \\
&= \frac{1}{5} (12 - 1x_1^{(k+1)} - (-2)x_2^{(k+1)}) = 2.4 - 0.2x_1^{(k+1)} + 0.4x_2^{(k+1)}
\end{aligned}$$

was mit der weiter oben bereits hergeleiteten Iteration für dieses Beispiel natürlich übereinstimmt. Wir wählen den Null-Vektor als Startvektor, also $\mathbf{x}^{(0)} = (0, 0, 0)^T$ und setzen oben ein. Damit erhalten wir $x_1^{(1)} = 1.25$, $x_2^{(1)} = 2.2 + 0.4 \cdot 1.25 = 2.7$, $x_3^{(1)} = 2.4 - 0.2 \cdot 1.25 + 0.4 \cdot 2.7 = 3.23$, also $\mathbf{x}^{(1)} = (1.25, 2.7, 3.23)^T$. Weiteres Einsetzen liefert:

i	0	1	2	3	4
$\mathbf{x}^{(i)}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1.25 \\ 2.7 \\ 3.23 \end{pmatrix}$	$\begin{pmatrix} 1.1175 \\ 2.001 \\ 2.9769 \end{pmatrix}$	$\begin{pmatrix} 1.006025 \\ 2.00703 \\ 3.001607 \end{pmatrix}$	$\begin{pmatrix} 1.00135575 \\ 2.0002209 \\ 2.99981721 \end{pmatrix}$

Also können wir annehmen, dass diese Folge gegen $(1, 2, 3)^T$ konvergiert, und zwar schneller als mit dem Jacobi-Verfahren. Natürlich hätten wir die Iterationsgleichungen auch etwas übersichtlicher aus dem Zusammenhang

$$(\mathbf{D} + \mathbf{L})\mathbf{x}^{(k+1)} = -\mathbf{R}\mathbf{x}^{(k)} + \mathbf{b}$$

herleiten können:

$$\begin{pmatrix} 4 & 0 & 0 \\ -2 & 5 & 0 \\ 1 & -2 & 5 \end{pmatrix} \mathbf{x}^{(k+1)} = -\begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{x}^{(k)} + \begin{pmatrix} 5 \\ 11 \\ 12 \end{pmatrix}.$$

Komponentenweise folgt (ohne Berechnung der Inversen $(D + L)^{-1}$)

$$\begin{aligned}
4x_1^{(k+1)} &= -(-x_2^{(k)} + x_3^{(k)}) + 5 \\
-2x_1^{(k+1)} + 5x_2^{(k+1)} &= -x_3^{(k)} + 11 \\
x_1^{(k+1)} - 2x_2^{(k+1)} + 5x_3^{(k+1)} &= 12
\end{aligned}$$

bzw. wie erwartet

$$\begin{aligned}
x_1^{(k+1)} &= \frac{1}{4}x_2^{(k)} - \frac{1}{4}x_3^{(k)} + \frac{5}{4} \\
x_2^{(k+1)} &= \frac{2}{5}x_1^{(k+1)} - \frac{1}{5}x_3^{(k)} + \frac{11}{5} \\
x_3^{(k+1)} &= -\frac{1}{5}x_1^{(k+1)} + \frac{2}{5}x_2^{(k+1)} + \frac{12}{5}
\end{aligned}$$

4.7.3 Konvergenz

Wir haben in Kapitel 3.4 bereits Kriterien bezüglich der Konvergenz von Fixpunktiterationen kennengelernt. Diese können direkt auf die vektoriellen Fixpunktgleichungen des Jacobi- und des Gauss-Seidel-Verfahrens angewandt werden, es muss dabei nur eine Norm statt des Betragszeichens verwendet werden.

Definition 4.9: anziehender / abstossender Fixpunkt [1]

- Gegeben sei eine Fixpunktiteration

$$\mathbf{x}^{(n+1)} = \mathbf{B}\mathbf{x}^{(n)} + \mathbf{c} =: F(\mathbf{x}^{(n)})$$

wobei \mathbf{B} eine $n \times n$ Matrix ist und $\mathbf{c} \in \mathbb{R}^n$. Weiter sei $\|\cdot\|$ eine der in Kap. 4.6.1 eingeführten Normen und $\bar{\mathbf{x}} \in \mathbb{R}^n$ erfülle $\bar{\mathbf{x}} = \mathbf{B}\bar{\mathbf{x}} + \mathbf{c} = F(\bar{\mathbf{x}})$. Dann heisst

- $\bar{\mathbf{x}}$ anziehender Fixpunkt, falls $\|\mathbf{B}\| < 1$ gilt
- $\bar{\mathbf{x}}$ abstossender Fixpunkt, falls $\|\mathbf{B}\| > 1$ gilt.

Unter Anwendung des Banachschen Fixpunktsatzes haben wir dann die folgende Fehlerabschätzung zur Verfügung:

Satz 4.4: Abschätzungen [1]

- Gegeben sei wie in obiger Definition eine Fixpunktiteration

$$\mathbf{x}^{(n+1)} = \mathbf{B}\mathbf{x}^{(n)} + \mathbf{c} =: F(\mathbf{x}^{(n)})$$

und $\bar{\mathbf{x}} \in \mathbb{R}^n$ sei ein bezüglich der Norm $\|\cdot\|$ anziehender Fixpunkt. Dann konvergiert die Fixpunktiteration für alle Startvektoren $\mathbf{x}^{(0)} \in \mathbb{R}^n$ gegen $\bar{\mathbf{x}}$ und es gelten die Abschätzungen

$$\begin{aligned} \|\mathbf{x}^{(n)} - \bar{\mathbf{x}}\| &\leq \frac{\|\mathbf{B}\|^n}{1 - \|\mathbf{B}\|} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \quad \text{a-priori Abschätzung} \\ \|\mathbf{x}^{(n)} - \bar{\mathbf{x}}\| &\leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{B}\|} \|\mathbf{x}^{(n)} - \mathbf{x}^{(n-1)}\| \quad \text{a-posteriori Abschätzung} \end{aligned}$$

Bemerkung: Der Vergleich mit den Definitionen für das Gesamt- und Einzelschrittverfahren liefert die Matrix \mathbf{B} :

- für das Gesamtschrittverfahren (Jacobi) ist

$$\mathbf{B} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{R}),$$

- für das Einzelschrittverfahren (Gauss-Seidel) ist

$$\mathbf{B} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{R}.$$

Ausserdem gilt mit der folgenden Definition:

Definition 4.10: Diagonaldominanz [1]

- \mathbf{A} ist eine **diagonaldominante Matrix**, falls eines der beiden folgenden Kriterien gilt:

- für alle $i = 1, \dots, n$: $|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|$ (Zeilensummenkriterium)
- für alle $j = 1, \dots, n$: $|a_{jj}| > \sum_{i=1, i \neq j}^n |a_{ij}|$ (Spaltensummenkriterium)

Satz 4.5: Konvergenz [1]

- Falls A diagonaldominant ist, konvergiert das Gesamtschrittverfahren (Jacobi) und auch das Einzelschrittverfahren (Gauss-Seidel) für $Ax = b$.

Bemerkungen:

- Die Diagonaldominanz von A ist nur ein hinreichendes Kriterium für die Konvergenz der iterativen Verfahren. Es gibt nicht diagonaldominante Matrizen, für die die Verfahren dennoch konvergieren. Ein notwendiges und hinreichendes Kriterium für Konvergenz ist, dass der Spektralradius $\rho(B) < 1$ ist (vgl. Def. 4.18).

Aufgabe 4.5 [1]:

- Prüfen Sie, ob das Jacobi-Verfahren in Beispiel 4.16 konvergiert. Schätzen Sie den Fehler des Vektors $x^{(5)}$ ab. Wie viele Schritte sollten Sie rechnen, damit der berechnete Näherungsvektor in jeder Komponente um max. 10^{-4} von der exakten Lösung $x = (1, 2, 3)^T$ abweicht? Vergleichen Sie Ihre Fehlerabschätzung mit den wirklichen Gegebenheiten.
- Bearbeiten Sie die Aufgabenstellung nochmals, aber mit dem Gauss-Seidel-Verfahren und dem Näherungsvektor $x^{(4)}$ aus Beispiel 4.17.

4.8 Eigenwerte und Eigenvektoren

Wichtige Anwendungen aus der linearen Algebra basieren auf der Bestimmung von Eigenwerten und Eigenvektoren von Matrizen. Da Eigenwerte einer reellen Matrix auch komplex werden können, benötigen wir zuerst eine Einführung in die komplexen Zahlen (siehe Kap. 4.8.1), bevor wir auf Eigenwerte und Eigenvektoren (Kap. 4.8.2) eingehen können.

4.8.1 Einführung in die komplexen Zahlen

Im Laufe der Jahrtausende mussten die Zahlenbereiche laufend erweitert werden, um neue Problemstellungen mathematisch beschreiben zu können (vgl. Abb. 4.6). Auch in der Menge der reellen Zahlen \mathbb{R} sind einfache Gleichungen der Art

$$x^2 + 1 = 0$$

bereits nicht mehr lösbar. Es braucht also eine zusätzliche Erweiterung auf die Menge der sogen. komplexen Zahlen. Diese Zahlenmenge lässt sich beschreiben, indem der Zahlenstrahl der reellen Zahlen durch eine zusätzliche Achse zu einer Zahlenebene aufgespannt wird (vgl. Abb. 4.7).

4.8.1.1 Grundbegriffe

Die Menge der komplexen Zahlen \mathbb{C} erweitert die Menge der reellen Zahlen \mathbb{R} , so dass nun also auch Gleichungen der Art

$$x^2 + 1 = 0$$

lösbar werden. Dafür wird die imaginäre Einheit i mit der Eigenschaft

$$i^2 = -1$$

eingeführt¹². Damit ergibt sich die Lösung von $x^2 = -1 = i^2$ zu $x = \pm i$. Während reelle Zahlen auf einem Zahlenstrahl dargestellt werden können, werden komplexe Zahlen in der (zweidimensionalen) komplexen Zahlenebene (auch Gaußsche Zahlenebene genannt) dargestellt, wobei jedem Punkt

$$P = (x, y)$$

¹²In der Elektrotechnik wird die imaginäre Einheit auch mit j bezeichnet, um eine Verwechslung mit der Stromstärke auszuschliessen. Auch in Python ist die imaginäre Einheit j .

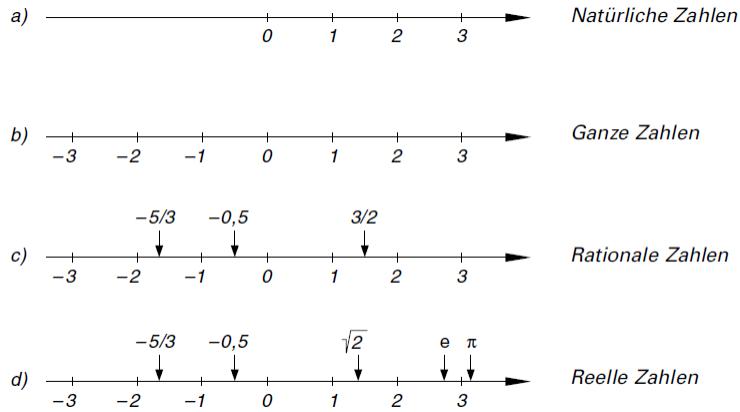


Abbildung 4.6: Zahlenbereiche von den natürlichen Zahlen \mathbb{N} (hier inkl. 0), zu den ganzen Zahlen \mathbb{Z} , den rationalen Zahlen \mathbb{Q} und den reellen Zahlen \mathbb{R} (aus [12]).

mit den kartesischen Koordinaten x und y in dieser Ebene eine komplexe Zahl z entsprechen soll (Abb. 4.7) mit der symbolischen Schreibweise

$$z = x + iy.$$

Durch das geordnete reelle Zahlenpaar (x, y) wird die komplexe Zahl eindeutig beschrieben und lässt sich als Bildpunkt in der Ebene darstellen

$$z = x + iy \longleftrightarrow P_z = (x, y)$$

Definition 4.11: Grundbegriffe Komplexe Zahlen [nach 12]

- Eine komplexe Zahl z ist ein geordnetes Paar (x, y) zweier reeller Zahlen x und y . Man schreibt symbolisch

$$z = x + iy$$

- Die imaginäre Einheit i ist definiert durch

$$i^2 = -1$$

- Die Menge der komplexen Zahlen wird mit \mathbb{C} bezeichnet:

$$\mathbb{C} = \{z \mid z = x + iy \text{ mit } x, y \in \mathbb{R}\}$$

Die Menge der reellen Zahlen ist also eine (echte) Teilmenge der komplexen Zahlen: $\mathbb{R} \subset \mathbb{C}$

- Die reellen Bestandteile x und y von z werden als Realteil und Imaginärteil bezeichnet:

- Realteil von z : $\operatorname{Re}(z) = x$
- Imaginärteil von z : $\operatorname{Im}(z) = y$

- Die komplexe Zahl $z = x + iy$ wird durch den Punkt $P_z = (x, y)$ in der kartesischen (x, y) -Ebene eindeutig dargestellt. Diese Ebene wird in diesem Zusammenhang häufig die komplexe oder auch Gaussche Zahlenebene genannt. Häufig wird in technischen Anwendungen eine komplexe Zahl durch ihren „Zeiger“ dargestellt, einem Pfeil vom Nullpunkt zum Bildpunkt P_z (vgl. Abb. 4.7).

- Die zu z konjugiert komplexe Zahl ist definiert als $z^* = x - iy$. Dies entspricht der an der x -Achse gespiegelten Zahl.

- Der Betrag einer komplexen Zahl ist definiert als $|z| = \sqrt{x^2 + y^2}$. Dies entspricht der Länge des Zeigers.

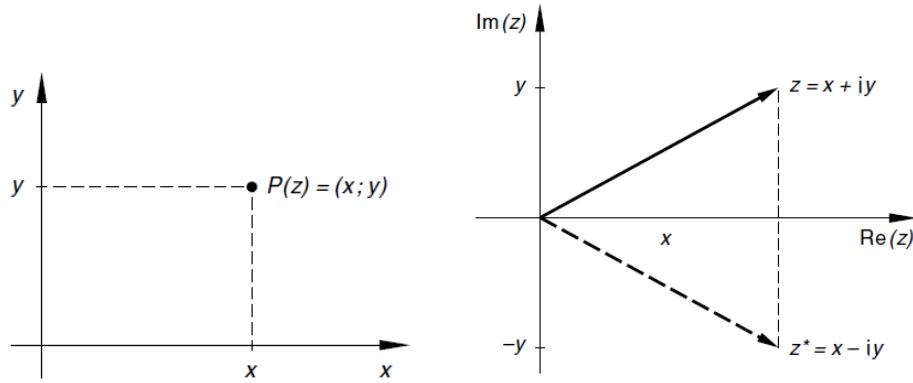


Abbildung 4.7: Darstellung einer komplexen Zahl $z = x + iy$ als Bildpunkt in der Ebene (links) und als sogn. Zeiger in der komplexen oder auch Gausschen Zahlenebene (rechts). Beide Abb. gemäss [12].

Aufgabe 4.6 [12]:

- Zeichnen Sie die folgenden komplexen Zahlen in der komplexen Zahlenebene mit ihren Zeigern ein:

$$\begin{aligned} z_1 &= 9 + 3i & z_2 &= 4 + 5i & z_3 &= -3 + 4j \\ z_4 &= -7 & z_5 &= -3 - 3i & z_6 &= 3 - 2j \\ z_7 &= -4i \end{aligned}$$

4.8.1.2 Darstellungsformen

Komplexe Zahlen können auf verschiedene Arten dargestellt werden. Man unterscheidet:

- Normalform (auch algebraische oder kartesische Form genannt):

$$z = x + iy$$

- Trigonometrische Form:

$$z = r(\cos \varphi + i \cdot \sin \varphi)$$

- Exponentialform:

$$z = r e^{i\varphi}$$

Die trigonometrische Form erhält man aus der Normalform durch die Einführung der sogn. Polarkoordinaten, nämlich dem Radius $r = |z| = \sqrt{x^2 + y^2}$ (entspricht dem Betrag von z bzw. der Länge des Zeigers) und dem Winkel φ (auch Argument od. Phase genannt) zwischen dem Zeiger und der x -Achse. Dabei gilt:

$$\begin{aligned} x &= r \cdot \cos \varphi \\ y &= r \cdot \sin \varphi \\ \Rightarrow z &= x + iy = r \cos \varphi + i \cdot r \sin \varphi = r(\cos \varphi + i \sin \varphi) \end{aligned}$$

Unter Verwendung der von Euler stammenden Formel

$$e^{i\varphi} = \cos \varphi + i \cdot \sin \varphi$$

folgt die Exponentialform

$$z = r e^{i\varphi}$$

Bemerkungen:

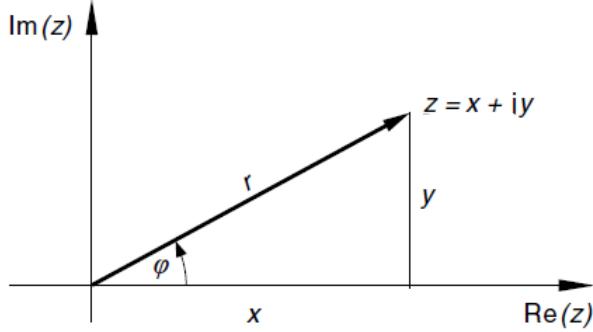


Abbildung 4.8: Darstellung einer komplexen Zahl in der Zahleebene in der Normalform und der Exponentialform (aus [12]).

- Die hier auftretende komplexe Exponentialform ist eine komplexwertige Funktion der reellen Variablen φ und periodisch mit der Periode 2π (da Sinus und Kosinus periodische Funktionen mit Periode 2π sind):

$$\begin{aligned} e^{i(\varphi+k \cdot 2\pi)} &= \cos(\varphi + k \cdot 2\pi) + i \cdot \sin(\varphi + k \cdot 2\pi) \\ &= \cos \varphi + i \cdot \sin \varphi \\ &= e^{i\varphi} \end{aligned}$$

- Den Beweis der Eulerschen Formel erhält man durch die Potenzreihenentwicklung von

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \dots,$$

wobei $x = i\varphi$ gesetzt und $i^2 = -1$ benutzt wird:

$$\begin{aligned} e^{i\varphi} &= 1 + i \frac{\varphi}{1!} - \frac{\varphi^2}{2!} - i \frac{\varphi^3}{3!} + \frac{\varphi^4}{4!} + i \frac{\varphi^5}{5!} - \frac{\varphi^6}{6!} - i \frac{\varphi^7}{7!} + \dots \\ &= \underbrace{\left(1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} - \frac{\varphi^6}{6!} + \dots\right)}_{\cos \varphi} + i \underbrace{\left(\varphi - \frac{\varphi^3}{3!} + \frac{\varphi^5}{5!} - \frac{\varphi^7}{7!} + \dots\right)}_{\sin \varphi} \\ &= \cos \varphi + i \sin \varphi \end{aligned}$$

Aufgabe 4.7 [12]:

- Zeichnen Sie die folgenden komplexen Zahlen in der komplexen Zahleebene mit ihren Zeigern ein:

$$\begin{aligned} z_1 &= 3e^{i\frac{\pi}{4}} & z_2 &= 4e^{i\frac{2}{3}\pi} & z_3 &= 4e^{i\frac{3}{2}\pi} \\ z_4 &= 3e^{-i \cdot 110^\circ} & z_5 &= 6e^{i\pi} & z_6 &= 4e^{i \cdot 340^\circ} \end{aligned}$$

4.8.1.3 Grundrechenarten

Analog zu den reellen Zahlen definieren wir bei den komplexen Zahlen die vier Grundrechenarten Addition (+), Subtraktion (-), Multiplikation (\cdot) und Division ($:$).

Definition 4.12: Summe, Differenz, Produkt und Division komplexer Zahlen

Es sei $z_1 = x_1 + iy_1$ und $z_2 = x_2 + iy_2$.

- Die Summation ist definiert als

$$z_1 + z_2 = (x_1 + x_2) + i(y_1 + y_2)$$

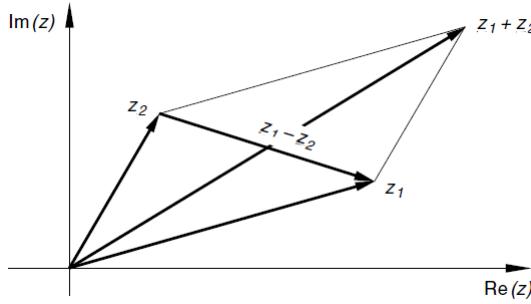


Abbildung 4.9: Zur Addition und Subtraktion zweier komplexer Zahlen (aus [12]).

- Die Subtraktion ist definiert als

$$z_1 - z_2 = (x_1 - x_2) + i(y_1 - y_2)$$

- Die Multiplikation ist definiert als

$$z_1 \cdot z_2 = (x_1 x_2 - y_1 y_2) + i(x_1 y_2 + x_2 y_1)$$

Dabei wird das Produkt $z_1 \cdot z_2 = (x_1 + iy_1)(x_2 + iy_2)$ wie im Reellen durch Ausmultiplizieren der Klammerausdrücke unter Benutzung der Beziehung $i^2 = -1$ berechnet

- Die Division ist definiert als

$$\begin{aligned} \frac{z_1}{z_2} &= \frac{z_1 \cdot z_2^*}{z_2 \cdot z_2^*} = \frac{(x_1 + iy_1)(x_2 - iy_2)}{(x_2 + iy_2)(x_2 - iy_2)} \\ &= \frac{(x_1 x_2 + y_1 y_2) + i(y_1 x_2 - x_1 y_2)}{x_2^2 + y_2^2} = \frac{(x_1 x_2 + y_1 y_2)}{x_2^2 + y_2^2} + i \frac{(y_1 x_2 - x_1 y_2)}{x_2^2 + y_2^2} \end{aligned}$$

Die Summation bzw. Subtraktion zweier komplexer Zahlen kann als Summe bzw. Differenz der beiden Zeiger interpretiert werden (vgl. Abb. 4.9).

Bemerkungen:

1. Multiplikation und Division lassen sich in der trigonometrischen bzw. exponentiellen Darstellung besonders einfach durchführen:

$$\begin{aligned} (a) \quad z_1 \cdot z_2 &= r_1 e^{i\varphi_1} \cdot r_2 e^{i\varphi_2} = r_1 r_2 e^{i(\varphi_1 + \varphi_2)} \\ (b) \quad z_1/z_2 &= r_1 e^{i\varphi_1} / (r_2 e^{i\varphi_2}) = \frac{r_1}{r_2} e^{i(\varphi_1 - \varphi_2)} \end{aligned}$$

2. Wir hatten bereits den Betrag einer komplexen Zahl $z = x + iy$ definiert als $|z| = \sqrt{x^2 + y^2}$. Es gilt also $|z|^2 = x^2 + y^2$ bzw.

$$\begin{aligned} |z|^2 &= z \cdot z^* = (x + iy)(x - iy) = x^2 + y^2 \\ |z| &= \sqrt{z \cdot z^*} \end{aligned}$$

3. Die vier Grundrechenoperationen für komplexe Zahlen genügen den folgenden Grundgesetzen:

- (a) Summe, Differenz, Produkt und Quotient zweier komplexer Zahlen ergibt wieder eine komplexe Zahl (Division durch 0 ist nicht erlaubt).
- (b) Addition und Multiplikation sind kommutativ

$$\begin{aligned} z_1 + z_2 &= z_2 + z_1 \\ z_1 \cdot z_2 &= z_2 \cdot z_1 \end{aligned}$$

(c) Addition und Multiplikation assoziativ

$$\begin{aligned} z_1 + (z_2 + z_3) &= (z_1 + z_2) + z_3 \\ z_1 \cdot (z_2 \cdot z_3) &= (z_1 \cdot z_2) \cdot z_3 \end{aligned}$$

(d) Addition und Multiplikation sind über das Distributivgesetz miteinander verbunden:

$$z_1 \cdot (z_2 + z_3) = z_1 \cdot z_2 + z_1 \cdot z_3$$

Aufgabe 4.8 [12]:

- Berechnen Sie von $z_1 = 4 - 8i$ und $z_2 = 3 + 4i$ die Summe, die Differenz, das Produkt und den Quotienten

4.8.1.4 Potenzieren und Radizieren / Fundamentalsatz der Algebra

Die n -te Potenz einer komplexe Zahl lässt sich einfach berechnen, wenn diese in der trigonometrischen oder der Exponentialform vorliegt:

Definition 4.13: Potenz einer komplexen Zahl in der Polarform [12]

- Sei $n \in \mathbb{N}$:

$$z = r \cdot e^{i\varphi} \Rightarrow z^n = (r e^{i\varphi})^n = r^n e^{in\varphi} = r^n (\cos(n\varphi) + i \cdot \sin(n\varphi))$$

Aus der Algebra ist bekannt, dass eine algebraische Gleichung n -ten Grades

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0$$

mit reellen Koeffizienten und Variablen $a_i, x \in \mathbb{R}$ höchstens n reelle Lösungen hat (auch Wurzeln genannt). Im komplexen Zahlenraum gibt es hingegen genau n Lösungen. Dies ist der Fundamentalsatz der Algebra:

Satz 4.6: Fundamentalsatz der Algebra [12]

- Eine algebraische Gleichung n -ten Grades mit komplexen Koeffizienten und Variablen $a_i, z \in \mathbb{C}$

$$a_n z^n + a_{n-1} z^{n-1} + \dots + a_1 z + a_0 = 0$$

besitzt in der Menge \mathbb{C} der komplexen Zahlen genau n Lösungen.

Bemerkungen:

- Die linke Seite der algebraischen Gleichung ist ein Polynom vom Grad n und lässt sich wie im Reellen in Linearfaktoren zerlegen mit den Nullstellen z_i :

$$P_n(z) = a_n(z - z_1)(z - z_2)(z - z_3) \dots (z - z_n)$$

- Sind sämtliche Koeffizienten $a_i (i = 0, 1, 2, \dots, n)$ ausschliesslich reell, treten komplexe Lösungen immer als Paare von zueinander konjugiert komplexer Zahlen auf.

Beispiel: das Polynom

$$P(z) = z^3 - z^2 + 4z - 4$$

hat nur eine reelle Nullstelle ($z_1 = 1$) und zwei zueinander komplex konjugiert Nullstellen ($z_2 = 2i$ und $z_3 = -2i$). Es lässt sich also in die Linearfaktoren

$$P(z) = (z - 1)(z - 2i)(z + 2i)$$

zerlegen.

Definition 4.14: Wurzel einer komplexen Zahl [12]

- Eine komplexe Zahl z wird als n -te Wurzel von $a \in \mathbb{C}$ bezeichnet, wenn sie der algebraischen Gleichung

$$z^n = a$$

genügt. Man schreibt

$$z = \sqrt[n]{a}$$

Satz 4.7: Lösungen der algebraisch Gleichung $z^n = a$ [12]

- Die Gleichung

$$z^n = a = r_0 e^{i\varphi} \quad (r_0 > 0; n = 2, 3, 4, \dots)$$

besitzt in der Menge \mathbb{C} genau n verschiedene Lösungen (Wurzeln)

$$z_k = r(\cos \varphi_k + i \cdot \sin \varphi_k) = r e^{i\varphi_k}$$

mit

$$r = \sqrt[n]{r_0} \text{ und } \varphi_k = \frac{\varphi + k \cdot 2\pi}{n} \quad (\text{für } k = 0, 1, 2, \dots, n-1).$$

Die zugehörigen Bildpunkte liegen in der komplexen Zahlebene auf einem Kreis um den Nullpunkt mit dem Radius $r = \sqrt[n]{r_0}$ und bilden die Ecken eines regelmässigen n -Ecks.

Bemerkung:

1. Es gilt wegen der 2π -Periodizität von

$$e^{i\varphi} = e^{i(\varphi+k \cdot 2\pi)}$$

wie gewünscht

$$z_k^n = \left(\sqrt[n]{r_0} \cdot e^{i\frac{\varphi+k \cdot 2\pi}{n}} \right)^n = r_0 e^{i(\varphi+k \cdot 2\pi)} = r_0 e^{i\varphi} = a$$

Beispiel 4.18 [12]:

- Berechnen Sie alle Lösungen der Gleichung für $z \in \mathbb{C}$:

$$z^6 = 1$$

- Lösung: In \mathbb{R} hat die Gleichung nur die zwei Lösungen $z = \pm 1$, in \mathbb{C} aber wegen

$$z^6 = 1 = 1 \cdot e^{i(0+k \cdot 2\pi)}$$

die 6 verschiedenen (paarweise komplex konjugierten) Lösungen

$$z_k = r e^{i\frac{0+k \cdot 2\pi}{6}} \quad (k = 0, 1, \dots, 5)$$

mit $r = \sqrt[6]{1} = 1$. Konkret lauten die Lösungen (vgl. Abb. 4.10):

$$\begin{aligned} k = 0 : \quad z_0 &= 1 \cdot e^{i \cdot 0} = 1 \\ k = 1 : \quad z_1 &= 1 \cdot e^{i \cdot \frac{\pi}{3}} = \cos\left(\frac{\pi}{3}\right) + i \cdot \sin\left(\frac{\pi}{3}\right) = 0.5 + \frac{\sqrt{3}}{2}i \\ k = 2 : \quad z_2 &= 1 \cdot e^{i \cdot \frac{2\pi}{3}} = \cos\left(\frac{2\pi}{3}\right) + i \cdot \sin\left(\frac{2\pi}{3}\right) = -0.5 + \frac{\sqrt{3}}{2}i \\ k = 3 : \quad z_3 &= 1 \cdot e^{i \cdot \frac{3\pi}{3}} = \cos(\pi) + i \cdot \sin(\pi) = -1 \\ k = 4 : \quad z_4 &= 1 \cdot e^{i \cdot \frac{4\pi}{3}} = \cos\left(\frac{4\pi}{3}\right) + i \cdot \sin\left(\frac{4\pi}{3}\right) = -0.5 - \frac{\sqrt{3}}{2}i \\ k = 5 : \quad z_5 &= 1 \cdot e^{i \cdot \frac{5\pi}{3}} = \cos\left(\frac{5\pi}{3}\right) + i \cdot \sin\left(\frac{5\pi}{3}\right) = 0.5 - \frac{\sqrt{3}}{2}i \end{aligned}$$

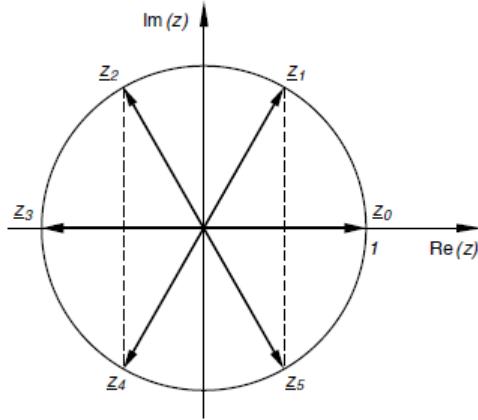


Abbildung 4.10: Die Lösungen der Gleichung $z^6 = 1$ (aus [12]).

Aufgabe 4.9 [12]:

- Berechnen Sie die Wurzeln der Gleichung

$$z^4 = 3 + 2j$$

Bringen Sie dafür die rechte Seite zuerst auf Exponentialform.

4.8.2 Einführung in Eigenwerte und Eigenvektoren ¹³

Eine $n \times n$ Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ bildet einen Vektor $\mathbf{x} \in \mathbb{C}^n$ auf einen Bildvektor $\mathbf{y} = \mathbf{Ax}$ ab (mit $\mathbf{y} \in \mathbb{C}^n$). Interessant für technische Anwendungen sind solche Vektoren \mathbf{x} , für die die lineare Abbildung \mathbf{Ax} ein zu \mathbf{x} paralleler, um einen Faktor λ gestreckter Vektor ist, es also gilt

$$\mathbf{Ax} = \lambda \mathbf{x}.$$

Solche Vektoren \mathbf{x} nennt man Eigenvektoren von \mathbf{A} , die Faktoren λ Eigenwerte. Schwingende Systeme (z.B. in der Elektrotechnik, Mechanik, Naturwissenschaften, etc.) werden wesentlich durch ihre Eigenwerte geprägt. Für Anwendungen ist es deshalb wichtig, die Eigenwerte zu kennen und ihre Grösse abschätzen zu können. Für den in der Einleitung (Kap. 4.1) beschriebenen PageRank-Algorithmus gibt der Eigenvektor der Google-Matrix zum Eigenwert 1 die relative Wichtigkeit der Webseiten an. Eigenwerte spielen auch in der Quantenmechanik eine besondere Rolle. Zum Beispiel geben die Eigenwerte der bekannten Schrödinger-Gleichung¹⁴ die erlaubten Energiewerte der Elektronen und die Eigenvektoren deren Wellenfunktionen an (die den Ort oder den Impuls beschreiben).

4.8.2.1 Grundbegriffe

Wir betrachten in diesem Kapitel nur reelle $n \times n$ Matrizen. In der Regel gelten die Definitionen und Sätze aber ebenfalls für komplexe Matrizen.

Definition 4.15: Eigenwert und Eigenvektor [13]

- Es sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. $\lambda \in \mathbb{C}$ heisst **Eigenwert** von \mathbf{A} , wenn es einen Vektor $\mathbf{x} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$ gibt mit

$$\mathbf{Ax} = \lambda \mathbf{x}.$$

\mathbf{x} heisst dann **Eigenvektor** von \mathbf{A} .

¹³Kapitel hauptsächlich übernommen aus [13]

¹⁴Eine partielle Differentialgleichung zur Beschreibung der quantenmechanischen Zustände eines nicht-relativistischen Systems (z.B. eines Wasserstoffatoms) benannt nach dem Österreichischen Physiker Erwin Schrödinger (1887-1961)

Bemerkungen:

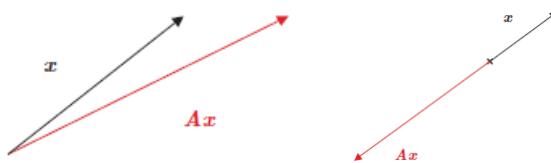
- Der Nullvektor $\mathbf{x} = \mathbf{0}$ wird als triviale Lösung von $\mathbf{Ax} = \lambda\mathbf{x}$ nicht berücksichtigt.
- Eigenvektoren werden i.d.R. auf die Länge 1 normiert, das heisst wir multiplizieren den Vektor mit dem Kehrwert seines Betrages. Wir hatten gesehen, dass der Betrag einer komplexen Zahl $z = x + iy$ definiert wird als

$$| z | = \sqrt{z \cdot z^*}.$$

Analog gilt für einen komplexen Vektor

$$\mathbf{z} = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} \Rightarrow | \mathbf{z} | = \sqrt{z_1 \cdot z_1^* + \dots + z_n \cdot z_n^*}$$

- In der linken Abbildung ist \mathbf{x} kein Eigenvektor von \mathbf{A} , rechts ist \mathbf{x} hingegen ein Eigenvektor von \mathbf{A} zum Eigenwert -2 (aus [13]).



Beispiel 4.19 [13]

- Die Matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & 1 \\ 7 & -5 & 9 \\ 6 & -6 & 9 \end{pmatrix}$$

hat den Eigenwert 1 mit Eigenvektor $(3, -1, -3)^T$ denn

$$\begin{pmatrix} 2 & 0 & 1 \\ 7 & -5 & 9 \\ 6 & -6 & 9 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \\ -3 \end{pmatrix} = 1 \cdot \begin{pmatrix} 3 \\ -1 \\ -3 \end{pmatrix}$$

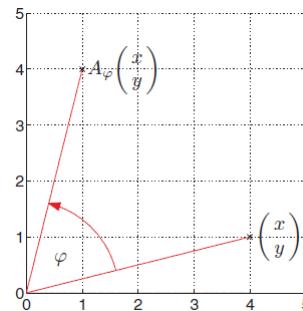
Desweiteren hat \mathbf{A} den Eigenwert 2 zum Eigenvektor $(1, 1, 0)^T$ und den Eigenwert 3 zum Eigenvektor $(1, 2, 1)^T$, wie man sich durch Nachrechnen leicht überzeugen kann. Hier sind die Eigenvektoren nicht normiert worden.

Aufgabe 4.10 [13]

- Die Drehmatrix zum Winkel $\varphi \in [0, 2\pi)$ in $\mathbb{R}^{2 \times 2}$ lautet

$$\mathbf{A}_\varphi = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}$$

und dreht einen Vektor \mathbf{x} in der Ebene um den Vektor φ :



Welche Vektoren ergeben nach einer Drehung nun ein Vielfaches von sich selbst? Diese Vektoren sind die Eigenvektoren, die Vielfachheitsfaktoren die Eigenwerte. Geben Sie ohne etwas zu rechnen die (reellen) Eigenvektoren und Eigenwerte basierend auf einer rein geometrischen Überlegung an.

4.8.2.2 Eigenschaften von Eigenwerten

Aus

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

folgt

$$\mathbf{A}\mathbf{x} - \lambda\mathbf{x} = \mathbf{0} \iff (\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{x} = \mathbf{0}$$

mit der $n \times n$ Einheitsmatrix \mathbf{I}_n . \mathbf{x} ist also die Lösung eines homogenen linearen Gleichungssystems. Dieses hat, wie wir aus der linearen Algebra wissen, nur dann eine nicht triviale Lösung $\mathbf{x} \neq \mathbf{0}$, wenn die Determinante $\det(\mathbf{A} - \lambda\mathbf{I}_n)$ gleich Null ist.

Satz 4.8: Eigenwerte und charakteristisches Polynom / Spur ([13])

- Es sei $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$. Dann gilt

$$\lambda \text{ ist Eigenwert von } \mathbf{A} \iff \det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$$

- Die Abbildung p definiert durch

$$p : \lambda \mapsto \det(\mathbf{A} - \lambda\mathbf{I}_n)$$

ist ein Polynom vom Grad n und wird **charakteristisches Polynom** von \mathbf{A} genannt. Die Eigenwerte von \mathbf{A} sind also die Nullstellen des charakteristischen Polynoms. Damit hat \mathbf{A} also genau n Eigenwerte, von denen manche mehrfach vorliegen können.

- Die Determinante der Matrix \mathbf{A} ist gerade das Produkt ihrer Eigenwerte $\lambda_1, \dots, \lambda_n$. Die Summe der Eigenwerte ist gleich der Summe der Diagonalelemente von \mathbf{A} , d.h. gleich der **Spur** (engl: trace, abgekürzt tr) von \mathbf{A} :

$$\begin{aligned} - \det \mathbf{A} &= \lambda_1 \cdot \lambda_2 \cdots \lambda_n \\ - \text{tr} \mathbf{A} &= a_{11} + a_{22} + \dots + a_{nn} = \lambda_1 + \lambda_2 + \dots + \lambda_n \end{aligned}$$

- Ist λ_i ein Eigenwert der regulären Matrix \mathbf{A} , so ist der Kehrwert $1/\lambda_i$ ein Eigenwert der inversen Matrix \mathbf{A}^{-1} .

Definition 4.16: Algebraische Vielfachheit / Spektrum [13]

- Es sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Die Vielfachheit, mit der λ als Nullstelle des charakteristischen Polynoms von \mathbf{A} auftritt, heisst **algebraische Vielfachheit** von λ .
- Das **Spektrum** $\sigma(\mathbf{A})$ ist die Menge aller Eigenwerte von \mathbf{A} .

Bemerkungen:

- Falls \mathbf{A} eine Diagonalmatrix ist, dann gilt

$$\mathbf{A} - \lambda\mathbf{I}_n = \begin{pmatrix} a_{11} - \lambda & 0 & \cdots & 0 \\ 0 & a_{22} - \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} - \lambda \end{pmatrix}$$

mit

$$p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}_n) = (a_{11} - \lambda)(a_{22} - \lambda) \dots (a_{nn} - \lambda).$$

Die Diagonalelemente der Diagonalmatrix sind also in diesem Fall die Nullstellen des charakteristischen Polynoms und damit genau die Eigenwerte von \mathbf{A} .

- Die gleiche Feststellung gilt auch für obere oder untere Dreiecksmatrizen, da für diese die Determinante ebenfalls gerade das Produkt der Diagonalelemente ist, wie wir im Zusammenhang mit dem Gauss-Algorithmus festgestellt hatten (vgl. Aufg. 4.2), also z.B. für

$$\mathbf{A} - \lambda \mathbf{I}_n = \begin{pmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} - \lambda \end{pmatrix}$$

gilt ebenfalls

$$p(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I}_n) = (a_{11} - \lambda)(a_{22} - \lambda) \dots (a_{nn} - \lambda).$$

Wir können also zusammenfassen: die Eigenwerte einer Diagonalmatrix oder einer Dreiecksmatrix sind deren Diagonalelemente.

- Wie sieht es jetzt für allgemeine Matrizen aus? Man könnte $\det(\mathbf{A} - \lambda \mathbf{I}_n)$ jeweils über den Gauss-Algorithmus berechnen, der eine Matrix ja in die obere Dreiecksform bringt. Für grosse Matrizen \mathbf{A} ist dies bekanntlich aber zu aufwändig, weshalb wir andere Methoden benötigen. Bevor wir auf diese Methoden eingehen können, brauchen wir aber noch zusätzliche Informationen zu Eigenwerten und -vektoren.

Beispiele 4.20 [13]

1. Das charakteristische Polynom einer 2×2 -Matrix können wir einfach berechnen

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \Rightarrow \mathbf{A} - \lambda \mathbf{I}_2 = \begin{pmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{pmatrix}$$

mit $p(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I}_n) = (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21}$.

Gemäß dem Fundamentalsatz der Algebra kann das charakteristische Polynom aber natürlich komplexe Nullstellen aufweisen, auch wenn alle Elemente der Matrix und damit die Koeffizienten des charakteristischen Polynoms reell sind. Nicht reelle Nullstellen treten immer als konjugiert komplexe Paare auf.

2. Wir hatten bereits in Aufg. 4.9 aufgrund geometrischer Überlegungen gesehen, dass die Drehmatrix

$$\mathbf{A}_\varphi = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}$$

nur für den Fall von $\varphi = 0$ oder $\varphi = \pi$ reelle Eigenwerte haben kann. Überprüfen Sie das anhand des charakteristischen Polynoms von \mathbf{A}_φ und geben Sie die algebraische Vielfachheit der Eigenwerte an.

Lösung:

$$p(\lambda) = \det \begin{pmatrix} \cos \varphi - \lambda & -\sin \varphi \\ \sin \varphi & \cos \varphi - \lambda \end{pmatrix} = (\cos \varphi - \lambda)^2 + \sin^2 \varphi = 0$$

Wir erhalten doppelte Nullstellen:

$$\begin{aligned} \varphi &= 0 : p(\lambda) = (1 - \lambda)^2 = 0 \Rightarrow \lambda = 1 \text{ mit alg. Vielfachheit 2} \\ \varphi &= \pi : p(\lambda) = (-1 - \lambda)^2 = 0 \Rightarrow \lambda = -1 \text{ mit alg. Vielfachheit 2} \end{aligned}$$

Aufgabe 4.11 [13]

- Bestimmen Sie die Eigenwerte der Matrix sowie die Determinante und die Spur:

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

Generell ist es aber eine schlechte Idee, die Eigenwerte einer Matrix über die Nullstellen des charakteristischen Polynoms zu berechnen, weil die Berechnung der Determinanten einer Matrix zu aufwändig ist und ein schlecht konditioniertes Problem daraus entstehen kann. Betrachte Sie dazu das folgende Beispiel.

Beispiel 4.21

- Wir hatten in Kap.2 in Bsp. 2.5 bereits das Wilkinson-Polynom kennengelernt:

$$p(x) = \prod_{k=1}^{20} (x - k) = (x - 1)(x - 2) \cdot \dots \cdot (x - 20) = x^{20} - 210x^{19} + \dots + 20!$$

mit den Nullstellen

$$x_k = k \quad (k = 1, \dots, 20)$$

Wir können nun das Wilkinson-Polynom als charakteristisches Polynom der 20×20 Diagonalmatrix mit den Diagonalelementen $a_{kk} = k$ interpretieren und die Nullstellen $x_k = \lambda_k$ also als die Eigenwerte der Diagonalmatrix. Wir hatten gesehen, dass eine minime Änderung des Koeffizienten von x^{19} von -210 um 2^{-23} auf -210.0000001192093 eine deutliche Verschiebung der Nullstellen bewirkt und sich reelle Nullstellen sogar in komplexe änderten:

$$\begin{aligned} \tilde{x}_k \in & \{1.00000, 2.00000, 3.00000, 4.00000, 5.00000, 6.00001, 6.99970, 8.00727, 8.91725, \\ & 10.09527 \pm 0.64350i, 11.79363 \pm 1.65233i, 13.99236 \pm 2.51883i, 16.73074 \pm 2.81262i, \\ & 19.50244 \pm 1.94033i, 20.84691\}, \end{aligned}$$

Berechnen Sie also in der Praxis die Eigenwerte für $n > 3$ nie über das charakteristische Polynom und dessen Nullstellen!

4.8.2.3 Eigenschaften von Eigenvektoren

Hat man zwei Eigenvektoren \mathbf{x}, \mathbf{y} zum selben Eigenwert $\lambda \in \mathbb{C}$ einer Matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$, so ist $\mathbf{x} + \mathbf{y}$ und auch jedes Vielfache von \mathbf{x} ebenfalls ein Eigenvektor zum Eigenwert λ :

$$\begin{aligned} \mathbf{A}(\mathbf{x} + \mathbf{y}) &= \mathbf{Ax} + \mathbf{Ay} = \lambda\mathbf{x} + \lambda\mathbf{y} = \lambda(\mathbf{x} + \mathbf{y}) \\ \mathbf{A}(\mu\mathbf{x}) &= \mu\mathbf{Ax} = \mu\lambda\mathbf{x} = \lambda\mu\mathbf{x} \end{aligned}$$

Die Eigenvektoren zum gleichen Eigenwert λ bilden also einen Untervektorraum von \mathbb{C}^n , wenn man noch den Nullvektor (welcher nicht als Eigenvektor zählt), hinzunimmt. Man spricht dann vom Eigenraum von λ . Es gibt also nicht nur einen Eigenvektor zum Eigenwert λ , sondern unendlich viele, wobei jeder eine Linearkombination der Basisvektoren des Eigenraums ist. Dies stimmt mit unserem Wissen aus der linearen Algebra überein, dass das homogene lin. Gleichungssystem

$$(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{x} = 0$$

mit der Koeffizientenmatrix $\mathbf{A} - \lambda\mathbf{I}_n$ nur nichttriviale Lösungen hat, falls $\det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$ gilt. Es gibt in diesem Fall dann unendlich viele Lösungen \mathbf{x} (d.h. Eigenvektoren). Bei der Bestimmung der Eigenvektoren treten $n - r$ freie Parameter auf. Dabei ist $r = \text{Rg}(\mathbf{A} - \lambda\mathbf{I}_n) < n$ der Rang der Koeffizientenmatrix. Bei einem mehrfachen Eigenwert können auch mehrere Eigenvektoren auftreten (müssen aber nicht). Die Bestimmung der Eigenvektoren zu einem gegebenen Eigenwert λ entspricht also der Bestimmung der Basis des entsprechenden Eigenraums.

Satz 4.9: Eigenraum (nach [8] und [13])

- Es sei $\lambda \in \mathbb{C}$ ein Eigenwert von $\mathbf{A} \in \mathbb{R}^{n \times n}$. Dann bilden die Eigenvektoren zum Eigenwert λ zusammen mit dem Nullvektor $\mathbf{0}$ einen Unterraum von \mathbb{C}^n , den sogenannten **Eigenraum**.
- Der Eigenraum des Eigenwertes λ ist die Lösungsmenge des homogenen lin. Gleichungssystems

$$(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{x} = 0,$$

welches nur dann eine nicht-triviale (d.h. von $\mathbf{0}$ verschiedene) Lösung aufweist, wenn $\text{Rg}(\mathbf{A} - \lambda\mathbf{I}_n) < n$.

- Die Dimension des Eigenraumes von λ wird die **geometrische Vielfachheit** von λ genannt. Sie berechnet sich als

$$n - \text{Rg}(\mathbf{A} - \lambda\mathbf{I}_n)$$

und gibt die Anzahl der lin. unabhängigen Eigenvektoren zum Eigenwert λ .

- Geometrische und algebraische Vielfachheit eines Eigenwerts müssen nicht gleich sein. Die geom. Vielfachheit ist aber stets kleiner oder gleich der algebraischen Vielfachheit. Das heisst konkret:
 - Sind alle n Eigenwerte verschieden, so gehört zu jedem Eigenwert genau ein linear unabhängiger Eigenvektor, der bis auf einen (beliebigen) Faktor eindeutig bestimmt ist.
 - Tritt ein Eigenwert $k - fach$ auf (d.h. mit der algebraischen Vielfachheit k), so gehören dazu mindestens ein, höchstens aber k linear unabhängige Eigenvektoren. Beim Auftreten mehrfacher Eigenwerte kann also die Gesamtzahl linear unabhängiger Eigenvektoren kleiner sein als n .
 - Die zu verschiedenen Eigenwerten gehörenden Eigenvektoren sind immer linear unabhängig.
- Eigenvektoren zu konjugiert komplexen Eigenwerten sind ebenfalls zueinander konjugiert komplex.

Bemerkungen:

- Zur Erinnerung (für die exakten Definitionen der Begriffe wird auf die Vorlesung Lineare Algebra verwiesen):
 - die Dimension eines Vektorraumes ist die Anzahl seiner Basisvektoren
 - die Basis ist eine Teilmenge von linear unabhängigen (Basis-)Vektoren eines Vektorraumes, mit deren Hilfe sich jeder Vektor des Raumes eindeutig als endliche Linearkombination darstellen lässt
 - n Vektoren $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n \in \mathbb{C}^n$ sind linear unabhängig, falls

$$\mu_1 \mathbf{e}_1 + \mu_2 \mathbf{e}_2 + \dots + \mu_n \mathbf{e}_n = \mathbf{0}$$

nur für $\mu_1 = \mu_2 = \dots = \mu_n = 0$ erfüllt werden kann. Verschwinden nicht alle Koeffizienten μ_i dieser Gleichung, so heissen die Vektoren linear abhängig

- der Rang einer Matrix \mathbf{A} entspricht der Anzahl linear unabhängiger Zeilenvektoren von \mathbf{A} und lässt sich z.B. bestimmen aus der Anzahl der Zeilenvektoren ungleich $\mathbf{0}$, die nach der Umformung mittels Gauss-Algorithmus in die Zeilenstufenform übrig bleiben

Beispiel 4.22

- Berechnen Sie die Eigenwerte, die Eigenvektoren und Eigenräume der Matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 5 \\ -1 & -2 \end{pmatrix}$$

sowie den Rang der jeweiligen Koeffizientenmatrix $\text{Rg}(\mathbf{A} - \lambda \mathbf{I}_n)$.

- Lösung

- Eigenwerte:

$$p(\lambda) = \det \begin{pmatrix} 2 - \lambda & 5 \\ -1 & -2 - \lambda \end{pmatrix} = (2 - \lambda)(-2 - \lambda) - 5 \cdot (-1) = \lambda^2 + 1 = 0 \Rightarrow \lambda_{1,2} = \pm \sqrt{-1} = \pm i$$

- Eigenvektor zu $\lambda_1 = +i$:

Wir müssen das folgende homogene Gleichungssystem lösen

$$(\mathbf{A} - \lambda_1 \mathbf{I}_2) \mathbf{x} = \begin{pmatrix} 2 - i & 5 \\ -1 & -2 - i \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

respektive

$$(\mathbf{A} - \lambda_1 \mathbf{I}_2 \mid \mathbf{0}) = \left(\begin{array}{cc|c} 2 - i & 5 & 0 \\ -1 & -2 - i & 0 \end{array} \right).$$

Mit dem Gaußschen Eliminationsalgorithmus erhalten wir

$$i = 2, j = 1 \Rightarrow z_2 \equiv z_2 - \frac{(-1)}{(2-i)} z_1 \Rightarrow \left(\begin{array}{cc|c} 2-i & 5 & 0 \\ 0 & \underbrace{-2-i - \frac{-5}{(2-i)}}_{=0} & 0 \end{array} \right).$$

Wir sehen aus der Zeilenstufenform, dass die zweite Zeile durchgängig zu 0 wird, weil

$$-2 - i - \frac{-5}{(2-i)} = -2 - i - \frac{-5}{(2-i)} \cdot \frac{(2+i)}{(2+i)} = -2 - i - \frac{-10 - 5i}{4+1} = -2 - i - (-2 - i) = 0.$$

Damit ist $\text{Rg}(\mathbf{A} - \lambda_1 \mathbf{I}_2) = 1$ wie erwartet kleiner als $n = 2$ und wir erwarten einen linear unabhängigen Eigenvektor, da $n - \text{Rg}(\mathbf{A} - \lambda_1 \mathbf{I}_2) = 1$. Das wussten wir eigentlich schon, da wir gemäß obigen Satz bei zwei verschiedenen Eigenwerten auch zwei verschiedene linear unabhängige Eigenvektoren erwarten. Aus

$$0 \cdot x_2 = 0$$

folgt, dass das Gleichungssystem für beliebige Werte x_2 lösbar ist. Um x_1 bestimmen zu können, müssen wir uns auf einen Wert festlegen, z.B. $x_2 = 1$. Aus der ersten Zeile

$$(2 - i)x_1 + 5x_2 = 0$$

folgt

$$x_1 = \frac{-5}{2-i}x_2 = \frac{-5}{2-i} \cdot 1 = \frac{-5}{2-i} \cdot \frac{2+i}{2+i} = \frac{-10i - 5i}{5} = -2 - i$$

und wir erhalten den Eigenvektor

$$\mathbf{x}_1 = \begin{pmatrix} -2 - i \\ 1 \end{pmatrix}.$$

Bei der Berechnung der Eigenvektoren mit Programmen wie Matlab oder Python werden die Eigenvektoren i.d.R. noch auf die Länge 1 normiert, hier wäre das

$$\tilde{\mathbf{x}}_1 = \frac{1}{\sqrt{(-2-i)(-2+i)+1^2}} \begin{pmatrix} -2 - i \\ 1 \end{pmatrix} = \frac{1}{\sqrt{6}} \begin{pmatrix} -2 - i \\ 1 \end{pmatrix} = \begin{pmatrix} -0.8165 - 0.4082i \\ 0.4082 \end{pmatrix}.$$

Für den Eigenraum erhalten wir also

$$\text{E}_{\lambda_1} = \{ \mathbf{x} \mid \mathbf{x} = \mu \begin{pmatrix} -2 - i \\ 1 \end{pmatrix}, \mu \in \mathbb{R} \}$$

– Eigenvektor zu $\lambda_2 = -i$:

Die analoge Rechnung ergibt:

$$(\mathbf{A} - \lambda_2 \mathbf{I}_2 \mid \mathbf{0}) = \left(\begin{array}{cc|c} 2+i & 5 & 0 \\ -1 & -2+i & 0 \end{array} \right)$$

$$i = 2, j = 1 \Rightarrow z_2 \equiv z_2 - \frac{(-1)}{(2+i)} z_1 \Rightarrow \left(\begin{array}{cc|c} 2+i & 5 & 0 \\ 0 & \underbrace{-2+i - \frac{-5}{(2+i)}}_{=0} & 0 \end{array} \right)$$

weil in der zweiten Zeile

$$-2 + i - \frac{-5}{(2+i)} = -2 + i - \frac{-5}{(2+i)} \cdot \frac{(2-i)}{(2-i)} = -2 + i - \frac{-10 + 5i}{4+1} = -2 + i - (-2 + i) = 0.$$

Wähle $x_2 = 1$, dann folgt aus der ersten Zeile

$$(2+i)x_1 + 5x_2 = 0 \Rightarrow x_1 = \frac{-5}{2+i}x_2 = \frac{-5}{2+i} \cdot 1 = \frac{-5}{2+i} \cdot \frac{2-i}{2-i} = \frac{-10i + 5i}{5} = -2 + i$$

und wir erhalten den Eigenvektor

$$\mathbf{x}_2 = \begin{pmatrix} -2+i \\ 1 \end{pmatrix}$$

resp. normiert

$$\tilde{\mathbf{x}}_2 = \begin{pmatrix} -0.8165 + 0.4082i \\ 0.4082 \end{pmatrix}.$$

und für den Eigenraum

$$E_{\lambda_2} = \{ \mathbf{x} \mid \mathbf{x} = \mu \begin{pmatrix} -2+i \\ 1 \end{pmatrix}, \mu \in \mathbb{R} \}$$

Wie gemäss obigem Satz erwartet, sind die Eigenvektoren zueinander komplex konjugiert, also $\mathbf{x}_1 = \mathbf{x}_2^*$, da auch die zugehörigen Eigenwerte $\lambda_1 = \lambda_2^*$ zueinander komplex konjugiert sind.

Aufgabe 4.12 [12]

- Bestimmen Sie die Eigenwerte und Eigenvektoren der Matrix

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

Zur Erinnerung: die Determinante einer 3×3 Matrix \mathbf{B} berechnet sich z.B. als

$$\begin{aligned} \det \mathbf{B} &= \begin{vmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{vmatrix} = b_{11} \begin{vmatrix} b_{22} & b_{23} \\ b_{32} & b_{33} \end{vmatrix} - b_{12} \begin{vmatrix} b_{21} & b_{23} \\ b_{31} & b_{33} \end{vmatrix} + b_{13} \begin{vmatrix} b_{21} & b_{22} \\ b_{31} & b_{32} \end{vmatrix} \\ &= b_{11}(b_{22}b_{33} - b_{23}b_{32}) - b_{12}(b_{21}b_{33} - b_{23}b_{31}) + b_{13}(b_{21}b_{32} - b_{22}b_{31}) \end{aligned}$$

4.8.3 Numerische Berechnung von Eigenwerten und Eigenvektoren

Die Verfahren für die numerische Berechnung von Eigenwerten und Eigenvektoren einer Matrix \mathbf{A} beruhen meist auf der Idee einer Koordinatentransformation. Dabei wird nach einer neuen Basis gesucht, in der die Matrix eine Form erhält (z.B. die einer Dreiecksmatrix), in der sich die Eigenwerte leichter ablesen lassen. Ein solcher Basiswechsel führt zu einer sogenannten Ähnlichkeitstransformation der Matrix.

Betrachten Sie dazu die folgende Definition und den Satz:

Definition 4.17: Ähnliche Matrizen / Diagonalisierbarkeit [13]

- Es seien $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ und \mathbf{T} eine reguläre Matrix mit

$$\mathbf{B} = \mathbf{T}^{-1} \mathbf{A} \mathbf{T},$$

so heissen \mathbf{B} und \mathbf{A} zueinander **ähnliche Matrizen**. Man sagt, \mathbf{B} geht aus \mathbf{A} durch eine Ähnlichkeitstransformation hervor.

- Im Spezialfall, dass $\mathbf{B} = \mathbf{D}$ eine Diagonalmatrix ist, also

$$\mathbf{D} = \mathbf{T}^{-1} \mathbf{A} \mathbf{T},$$

nennt man \mathbf{A} **diagonalisierbar**.

Satz 4.10: Eigenwerte und Eigenvektoren ähnlicher / diagonalisierbarer Matrizen [13]

- Es seien $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ zueinander ähnliche Matrizen. Dann gilt:
 1. \mathbf{A} und \mathbf{B} haben dieselben Eigenwerte, inkl. deren algebraischer Vielfachheit.
 2. Ist \mathbf{x} ein Eigenvektor zum Eigenwert λ von \mathbf{B} , dann ist $\mathbf{T}\mathbf{x}$ ein Eigenvektor zum Eigenwert λ von \mathbf{A} .
 3. Im Spezialfall, dass \mathbf{A} diagonalisierbar ist (also $\mathbf{D} = \mathbf{T}^{-1}\mathbf{AT}$ eine Diagonalmatrix ist), sind die n Diagonalelemente von \mathbf{D} die Eigenwerte von \mathbf{A} und die n linear unabhängigen Eigenvektoren von \mathbf{A} stehen in den Spalten von \mathbf{T} .

Bemerkungen:

- Die erste Aussage folgt aus dem Umstand, dass die Einheitsmatrix (wie alle Matrizen) ähnlich zu sich selbst ist (da $\mathbf{I}_n = \mathbf{T}^{-1}\mathbf{T} = \mathbf{T}^{-1}\mathbf{I}_n\mathbf{T}$), es also gilt

$$\lambda\mathbf{I}_n = \mathbf{T}^{-1}(\lambda\mathbf{I}_n)\mathbf{T} = \mathbf{T}^{-1}\lambda\mathbf{T}$$

und damit

$$\mathbf{B} - \lambda\mathbf{I}_n = \underbrace{\mathbf{T}^{-1}\mathbf{AT}}_{\mathbf{B}} - \underbrace{\mathbf{T}^{-1}\lambda\mathbf{T}}_{\lambda\mathbf{I}_n} = \mathbf{T}^{-1}(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{T}.$$

Deshalb ist das charakteristische Polynom von \mathbf{B} identisch zum charakteristischen Polynom von \mathbf{A} , denn unter Benutzung der Regeln für Determinanten erhalten wir

$$\det(\mathbf{B} - \lambda\mathbf{I}_n) = \det(\mathbf{T}^{-1}(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{T}) = \det \mathbf{T}^{-1} \cdot \det(\mathbf{A} - \lambda\mathbf{I}_n) \cdot \det \mathbf{T} = \det(\mathbf{A} - \lambda\mathbf{I}_n)$$

wegen $\det \mathbf{T}^{-1} = (\det \mathbf{T})^{-1}$.

- Die zweite Aussage folgt aus

$$(\mathbf{B} - \lambda\mathbf{I}_n)\mathbf{x} = \mathbf{0} \iff \mathbf{T}^{-1}(\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{T}\mathbf{x} = \mathbf{0} \iff (\mathbf{A} - \lambda\mathbf{I}_n)\mathbf{T}\mathbf{x} = \mathbf{0}$$

- Die dritte Aussage folgt aus Aussagen eins und zwei. Oder im Detail: es gilt

$$\mathbf{T}^{-1}\mathbf{AT} = \mathbf{D} \iff \mathbf{T}\mathbf{T}^{-1}\mathbf{AT} = \mathbf{TD} \iff \mathbf{AT} = \mathbf{TD}$$

Seien $\lambda_1, \dots, \lambda_n$ die Diagonalelemente von \mathbf{D} und \mathbf{e}_i der i -te Einheitsvektor von \mathbb{R}^n , dann ist $\mathbf{T}\mathbf{e}_i$ gerade die i -te Spalte von \mathbf{T} und es gilt

$$\mathbf{AT}\mathbf{e}_i = \mathbf{TD}\mathbf{e}_i = \mathbf{T}\lambda_i\mathbf{e}_i = \lambda_i\mathbf{T}\mathbf{e}_i.$$

Also ist $\mathbf{T}\mathbf{e}_i$ der Eigenvektor von \mathbf{A} zum Eigenwert λ_i .

Wenn es uns nun also gelingt, eine zu \mathbf{A} ähnliche Matrix \mathbf{B} zu finden, die entweder eine Diagonalmatrix oder eine Dreiecksmatrix ist, können wir aus den Diagonalelementen von \mathbf{B} die Eigenwerte von \mathbf{A} ablesen und die Eigenvektoren bestimmen. Dies Idee wird im QR -Verfahren im nächsten Abschnitt angewendet.

Beispiel 4.23

- Die Matrizen

$$\mathbf{A} = \begin{pmatrix} 2 & 2 & 2 \\ 0 & 3 & 1 \\ 1 & -2 & 1 \end{pmatrix} \quad \text{und} \quad \mathbf{B} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{pmatrix}$$

sind zueinander ähnlich, da

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{AT}$$

mit

$$\mathbf{T} = \frac{1}{3} \begin{pmatrix} 2 & -1 & 1 \\ 1 & 1 & -1 \\ -2 & 1 & 2 \end{pmatrix}$$

Da \mathbf{B} eine Dreiecksmatrix ist, sind die Eigenwerte gerade die Diagonalelemente, also $\lambda_1 = 1$, $\lambda_2 = 2$ und $\lambda_3 = 3$. Wegen der Ähnlichkeit zwischen \mathbf{A} und \mathbf{B} sind das auch gerade die Eigenwerte von \mathbf{A} .

Aufgabe 4.13

- Bestimmen Sie die Eigenwerte und Eigenvektoren sowie deren algebraische und geometrische Vielfachheiten von

$$\mathbf{A} = \begin{pmatrix} 3 & 1 & -1 \\ 2 & 4 & -2 \\ 1 & 1 & 1 \end{pmatrix}$$

unter Benutzung der Ähnlichkeit

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

4.8.3.1 Das QR-Verfahren

Konkret suchen wir nun nach einer rechts-oberen Dreiecksmatrix \mathbf{B} , die zu \mathbf{A} ähnlich ist. Die Diagonalelemente von \mathbf{B} sind dann gerade die Eigenwerte von \mathbf{A} . Wir können auch schon zufrieden sein, wenn \mathbf{B} keine ‘perfekte’ rechts-obere Dreiecksmatrix ist, d.h. wenn unterhalb der Diagonalelemente von \mathbf{B} nicht alle Elemente verschwinden, aber betragsmäßig ausreichend klein sind. Die Diagonalelemente von \mathbf{B} sollten dann immer noch eine akzeptable Näherung der Eigenwerte von \mathbf{A} sein.

Besonders nützlich wäre es, für die Ähnlichkeitstransformation $\mathbf{B} = \mathbf{T}^{-1} \mathbf{A} \mathbf{T}$ für \mathbf{T} orthogonale Matrizen \mathbf{Q} zu verwenden, da für diese ja bekanntlich $\mathbf{Q}^{-1} = \mathbf{Q}^T$ gilt und wir deshalb keine Inversen bestimmen müssen. Mit der \mathbf{QR} -Zerlegung in Kap. 4.5.2 haben wir bereits auch eine Zerlegung kennengelernt, die \mathbf{A} in eine orthogonale Matrix \mathbf{Q} und in eine rechtsobere Matrix \mathbf{R} zerlegt. Das ist aber aber noch keine Ähnlichkeitstransformation. Durch eine fortlaufende Wiederholung der \mathbf{QR} -Zerlegung können wir aber hoffen, die gesuchte rechts-obere Matrix \mathbf{B} (die ähnlich zu \mathbf{A} sein soll), wie folgt zu erhalten:

1. Schritt: führe die \mathbf{QR} -Zerlegung von \mathbf{A} aus:

$$\mathbf{A} = \mathbf{Q}_1 \mathbf{R}_1$$

Wir erhalten daraus durch Auflösen nach \mathbf{R}_1

$$\mathbf{R}_1 = \mathbf{Q}_1^T \mathbf{A}$$

und durch beidseitige Multiplikation mit \mathbf{Q}_1

$$\underbrace{\mathbf{R}_1 \mathbf{Q}_1}_{\mathbf{A}_1} = \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_1$$

eine Ähnlichkeitstransformation und identifizieren die linke Seite als $\mathbf{A}_1 = \mathbf{R}_1 \mathbf{Q}_1$. Natürlich hat \mathbf{A}_1 wegen der Ähnlichkeit mit \mathbf{A} die gleichen Eigenwerte.

2. Schritt: führe nun die \mathbf{QR} -Zerlegung von \mathbf{A}_1 aus:

$$\mathbf{A}_1 = \mathbf{Q}_2 \mathbf{R}_2$$

Wir erhalten daraus durch Auflösen nach \mathbf{R}_2

$$\mathbf{R}_2 = \mathbf{Q}_2^T \mathbf{A}_1$$

und durch beidseitige Multiplikation mit \mathbf{Q}_2

$$\underbrace{\mathbf{R}_2 \mathbf{Q}_2}_{\mathbf{A}_2} = \mathbf{Q}_2^T \mathbf{A}_1 \mathbf{Q}_2$$

eine Ähnlichkeitstransformation und identifizieren die linke Seite als $\mathbf{A}_2 = \mathbf{R}_2 \mathbf{Q}_2$. Wieder hat \mathbf{A}_2 wegen der Ähnlichkeit mit \mathbf{A}_1 und \mathbf{A} die gleichen Eigenwerte.

3. Schritt: etc.

Tatsächlich konvergieren die Matrizen \mathbf{A}_i für $i \rightarrow \infty$ gegen eine Matrix \mathbf{A}_∞ , deren reelle Eigenwerte auf der Diagonalen stehen, für deren komplexe Eigenwerte aber 2×2 Blöcke auf der Diagonalen übrigbleiben (siehe auch Bsp. 4.23):

$$\mathbf{A}_\infty = \begin{pmatrix} A_{11} & * & * & * \\ 0 & A_{22} & * & * \\ \vdots & \ddots & \ddots & * \\ 0 & \dots & 0 & A_{ss} \end{pmatrix}$$

mit 1×1 oder 2×2 Matrizen A_{11}, \dots, A_{ss} . Dies ist das QR -Verfahren zur numerischen Berechnung der Eigenwerte von \mathbf{A} .

QR -Verfahren [13]:

Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$

- $\mathbf{A}_0 := \mathbf{A}$
- $\mathbf{P}_0 := \mathbf{I}_n$
- Für $i = 0, 1, 2, \dots$:
 - $\mathbf{A}_i := \mathbf{Q}_i \cdot \mathbf{R}_i$ # berechne die QR -Zerlegung von \mathbf{A}_i
 - $\mathbf{A}_{i+1} := \mathbf{R}_i \cdot \mathbf{Q}_i$
 - $\mathbf{P}_{i+1} := \mathbf{P}_i \cdot \mathbf{Q}_i$

Dann konvergiert die Folge der Matrizen \mathbf{A}_i für $i \rightarrow \infty$ gegen eine Matrix \mathbf{A}_∞ , die auf der Diagonalen nur einzelne Elemente oder 2×2 Blöcke aufweist. Die Eigenwerte von \mathbf{A} sind dann die Diagonalelemente und die Eigenwerte der 2×2 Blöcke (für die konjugiert komplexen Eigenwertpaare, welche z.B. als Nullstellen des charakteristischen Polynoms bestimmt werden können).

Bemerkungen:

- Falls alle Eigenwerte der Matrix betragsmäßig verschieden sind (also $|\lambda_i| \neq |\lambda_j|$ für $i \neq j$), so hat die Matrix \mathbf{A} genau n reelle Eigenwerte und es treten keine 2×2 Blöcke auf. \mathbf{A}_∞ ist dann eine “perfekte” obere Dreiecksmatrix.
- Das Verfahren berechnet die Eigenwerte, nicht aber die Eigenvektoren (diese müssen basierend auf den konkreten Eigenwerten separat berechnet werden), ausser falls \mathbf{A} symmetrisch ist.
- Falls \mathbf{A} symmetrisch ist, also $\mathbf{A}^T = \mathbf{A}$, und $|\lambda_i| \neq |\lambda_j|$ für die n Eigenwerte von \mathbf{A} gilt, so konvergiert die Folge der Matrizen $\mathbf{P}_k = \mathbf{Q}_0 \mathbf{Q}_1 \cdot \dots \cdot \mathbf{Q}_k$ gegen eine orthogonale Matrix, deren Spalten gerade die Eigenvektoren von \mathbf{A} bilden.

Beispiel 4.24

- Wir berechnen die Eigenwerte und Eigenvektoren der Matrix

$$\mathbf{A} = \begin{pmatrix} 1 & -2 & 0 \\ 2 & 0 & 1 \\ 0 & -2 & 1 \end{pmatrix}$$

- Lösung: Das QR -Verfahren liefert z.B. nach 100 Iterationen die Matrix

$$\mathbf{B} = \left(\begin{array}{cc|c} -0.1072 & -2.5543 & -0.6586 \\ 2.3955 & 1.1072 & -0.2575 \\ \hline 0 & 0 & 1 \end{array} \right).$$

Offensichtlich handelt es sich nicht um eine obere Dreiecksmatrix, das Element $a_{21} = 2.3955$ wird unabhängig von der Anzahl Iterationen nie 0. Das heisst, Die Matrix besteht also aus einem 2×2 Block und dem Diagonalelement a_{33} . Wir wissen also, die Matrix hat den reellen Eigenwert

$$\lambda_3 = a_{33} = 1$$

und zwei komplex konjugierte Eigenwerte, die den Eigenwerten des 2×2 Blocks

$$\begin{pmatrix} -0.1072 & -2.5543 \\ 2.3955 & 1.1072 \end{pmatrix}$$

entsprechen, mit dem charakteristischen Polynom

$$\begin{aligned} p(\lambda) &= (-0.1072 - \lambda)(1.1072 - \lambda) - (-2.5543)(2.3955) \\ &\approx \lambda^2 - 1\lambda + 6 \end{aligned}$$

und den Nullstellen

$$\lambda_{1,2} = 0.5 \pm 2.3979i$$

Für die Berechnung der Eigenvektoren müssen wir nun noch die zugehörigen Gleichungssysteme

$$(\mathbf{A} - \lambda_i \mathbf{I}_n) \mathbf{x}_i = 0$$

separat lösen und erhalten die normierten Eigenvektoren

$$\mathbf{x}_1 = \begin{pmatrix} 0.1091 + 0.5233i \\ 0.6547 \\ 0.1091 + 0.5233i \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} 0.1091 - 0.5233i \\ 0.6547 \\ 0.1091 - 0.5233i \end{pmatrix}, \quad \mathbf{x}_3 = \begin{pmatrix} -0.4472 \\ 0 \\ 0.8944 \end{pmatrix}$$

4.8.3.2 Vektoriteration

In verschiedenen numerischen Verfahren spielt der betragsmässig grösste Eigenwert eine wichtige Rolle. Dieser definiert den sogn. Spektralradius einer Matrix. Statt mit dem QR -Verfahren alle Eigenwerte zu bestimmen, kann mit weniger Aufwand nur der betragsmässig grösste Eigenwert und der zugehörige Eigenvektor bestimmt werden.

Definition 4.18: Spektralradius [14]

- Der Spektralradius $\rho(\mathbf{A})$ einer Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ ist definiert als

$$\rho(\mathbf{A}) = \max\{|\lambda| \mid \lambda \text{ ist ein Eigenwert von } \mathbf{A} \in \mathbb{R}^{n \times n}\}$$

Wir hatten den Spektralradius bereits bei den Matrizennormen in Def. 4.5 kurz kennengelernt:

$$\text{2-Norm, Spektralnorm : } \|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^T \mathbf{A})}$$

Das Verfahren, welches uns den Spektralradius und den zugehörigen Eigenvektor berechnet, ist die Vektoriteration bzw. von-Mises-Iteration (im Englischen auch ‘‘Power Method’’).

Vektoriteration / von-Mises-Iteration [14]:

- Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ eine diagonalisierbare Matrix mit den Eigenwerten $\lambda_1, \dots, \lambda_n$ und dem betragsmässig grössten Eigenwert λ_1 mit

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|,$$

so konvergieren für (fast) jeden Startvektor $\mathbf{v}^{(0)} \in \mathbb{C}^n$ mit Länge 1 die Folgen

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

für $k \rightarrow \infty$ gegen einen Eigenvektor \mathbf{v} zum Eigenwert λ_1 von \mathbf{A} (also $\mathbf{v}^{(k)} \rightarrow \mathbf{v}$ und $\lambda^{(k)} \rightarrow \lambda_1$).

Beispiel 4.25 [14]

- Bestimmen Sie mit Python den betragsmässig grössten Eigenwert und einen zugehörigen Eigenvektor der Matrix

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

mit der Vektoriteration und dem Startvektor $\mathbf{x} = (1, 0, 0)^T$.

- Lösung:

k	$x^{(k)}$	$\lambda^{(k)}$	k	$x^{(k)}$	$\lambda^{(k)}$
0	(1.0000, 0.0000, 0.0000) ^T		5	(0.2970, 0.6109, 0.7339) ^T	2.7303
1	(0.2673, 0.8018, 0.5345) ^T	1.0000	6	(0.3086, 0.5942, 0.7427) ^T	2.9408
2	(0.5298, 0.5298, 0.6623) ^T	1.8571	7	(0.2979, 0.5996, 0.7428) ^T	3.0306
3	(0.2923, 0.6577, 0.6942) ^T	3.4912	8	(0.3005, 0.5958, 0.7448) ^T	2.9869
4	(0.3463, 0.5860, 0.7326) ^T	2.7303	9	(0.2981, 0.5970, 0.7448) ^T	3.0068

Kapitel 5

Numerische Lösung nicht linearer Gleichungssysteme

In Kapitel 3 haben wir Verfahren kennengelernt, die Nullstellen nichtlinearer Funktionen mit einer Veränderlichen zu bestimmen, also die Gleichung $f(x_0) = 0$ zu lösen für ein nichtlineares $f : \mathbb{R} \rightarrow \mathbb{R}$. In Kapitel 4 behandelten wir dann den Fall von Systemen von n linearen Gleichungen für n Unbekannte, was man als Nullstellenbestimmung einer linearen Funktion $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ auffassen kann. In diesem Kapitel geht es nun um die Verallgemeinerung bzw. Anwendung der in Kap. 3 und 4 kennengelernten Verfahren auf Systeme von nichtlinearen Gleichungen, also um die Nullstellenbestimmung von nichtlinearen Funktionen $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

Lernziele:

- Sie kennen die Definition einer Funktion mit mehreren Variablen und wissen, wie diese grafisch dargestellt werden kann.
- Sie können die partiellen Ableitungen erster Ordnung einer Funktion mit mehreren Variablen definieren und berechnen.
- Sie kennen die Definition der Jacobi-Matrix und können diese für eine gegebene Funktion berechnen. Sie können damit Funktionen linearisieren.
- Sie können die in diesem Kapitel vorgestellten Algorithmen auf nichtlineare Gleichungssysteme anwenden und implementieren.
- Sie können die Unterschiede zwischen den Algorithmen erklären.

5.1 Einleitendes Beispiel

Zur historischen Entwicklung der Theorie nichtlinearer Gleichungssysteme ist in der Literatur wenig zu finden. Einer der Gründe mag sein, dass nichtlineare Gleichungssysteme im Gegensatz zu linearen Gleichungssystemen wesentlich komplexer sein können und Aussagen über die Lösbarkeit oder Konvergenz i.d.R. stark vom spezifischen Problem abhängig sind. Allgemeine Aussagen der Art, wie sie für lineare Gleichungssystemen möglich sind, sind für nichtlineare Gleichungssysteme wesentlich schwieriger. Nichtlineare Gleichungssysteme treten fast automatisch bei der Beschreibung und Lösung realer (häufig zeitabhängiger) Prozesse in Natur und Technik auf, die i.d.R. in Form von nichtlinearen Differentialgleichungen beschrieben werden (z.B. Ausbreitung elektromagnetischer Wellen im dreidimensionalen Raum).

Als einführendes Beispiel betrachten wir ein einfaches System mit zwei nichtlinearen Gleichungen und zwei Variablen (aus [6]):

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 + x_2 - 11 = 0 \\ f_2(x_1, x_2) &= x_1 + x_2^2 - 7 = 0 \end{aligned}$$

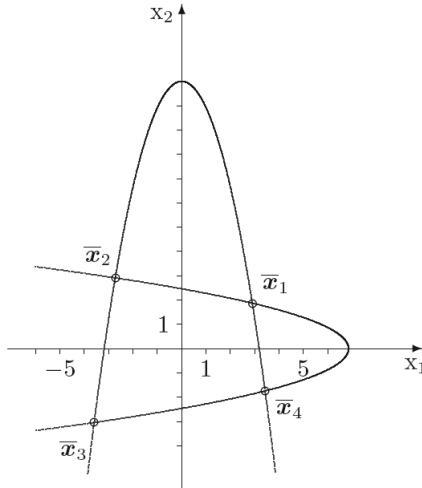


Abbildung 5.1: Grafische Darstellung der durch $f_1(x_1, x_2) = 0$ und $f_2(x_1, x_2) = 0$ implizit definierten Kurven sowie ihre Schnittpunkten (aus [6]).

Gesucht sind die Lösungen des Gleichungssystems. Diese lassen sich interpretieren als die Nullstellen der Funktion $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ gemäss

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Offensichtlich lässt sich ein solches System nicht in der Form $\mathbf{Ax} = \mathbf{b}$ darstellen, wie wir es von linearen Gleichungssystemen her kennen. Geometrisch lassen sich die Lösungen in diesem Beispiel bestimmen, indem wir die durch $f_1(x_1, x_2) = 0$ und $f_2(x_1, x_2) = 0$ implizit definierten Kurven in ein (x_1, x_2) -Koordinatensystem einzeichnen und die Schnittpunkte bestimmen, wie in Abb. 5.1 dargestellt. Dabei lautet die explizite Darstellung der Kurven

$$\begin{aligned} x_2 &= -x_1^2 + 11 \\ x_2 &= \sqrt{-x_1 + 7} \end{aligned}$$

und die Schnittpunkte sind

$$\bar{\mathbf{x}}_1 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \bar{\mathbf{x}}_2 = \begin{pmatrix} -2.8 \\ 3.2 \end{pmatrix}, \bar{\mathbf{x}}_3 = \begin{pmatrix} -3.8 \\ -3.3 \end{pmatrix}, \bar{\mathbf{x}}_4 = \begin{pmatrix} 3.4 \\ -1.7 \end{pmatrix}$$

5.2 Funktionen mit mehreren Variablen

Wie wir bei diesem einführenden Beispiel gesehen haben, ist für die numerische Lösung von nichtlinearen Gleichungssystemen das Verständnis von Funktionen mit mehreren Variablen unabdingbar. Deshalb wollen wir die wichtigsten Begriffe, insbesondere denjenigen der partiellen Ableitung, in diesem Abschnitt nochmals in Erinnerung rufen bzw. neu einführen, sofern nötig¹.

5.2.1 Definition einer Funktion mit mehreren Variablen

Die Definition für Funktionen

$$\begin{aligned} f : \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\mapsto y = f(x) \end{aligned}$$

mit der abhängigen Variablen y und der unabhängigen Variablen x kennen wir bereits aus der Analysis und erweitern Sie analog auf Funktionen mit mehreren Variablen:

¹Wir orientieren uns dabei am Kapitel IV ‘‘Differential- und Integralrechnung für Funktionen von mehreren Variablen’’ aus ‘‘Mathematik für Ingenieure und Naturwissenschaftler: Band 2’’ von Papula [8], greifen aber nur die für uns wichtigsten Punkte auf und passen sie, soweit sinnvoll, an.

Definition 5.1: Skalarwertige Funktionen mit mehreren Variablen

- Unter einer Funktion mit n unabhängigen Variablen x_1, \dots, x_n und einer abhängigen Variablen y versteht man eine Vorschrift, die jedem geordneten Zahlentupel (x_1, x_2, \dots, x_n) aus einer Definitionsmenge $D \subset \mathbb{R}^n$ genau ein Element y aus einer Wertemenge $W \subset \mathbb{R}$ zuordnet. Symbolische Schreibweise:

$$\begin{aligned} f : D \subset \mathbb{R}^n &\longrightarrow W \subset \mathbb{R} \\ (x_1, x_2, \dots, x_n) &\mapsto y = f(x_1, x_2, \dots, x_n) \end{aligned}$$

- Da das Ergebnis $y \in \mathbb{R}$ ein Skalar (eine Zahl) ist, redet man auch von einer **skalarwertigen** Funktion.

Bemerkungen:

- Die obige Definition lässt sich einfach erweitern auf beliebige **vektorwertige** Funktionen, die nicht einen Skalar, sondern einen Vektor als Wert zurückgeben:

$$\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^m,$$

mit

$$\mathbf{f}(x_1, \dots, x_n) = \begin{pmatrix} y_1 = f_1(x_1, x_2, \dots, x_n) \\ y_2 = f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ y_m = f_m(x_1, x_2, \dots, x_n) \end{pmatrix},$$

wobei die m Komponenten $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ ($i = 1, \dots, m$) von \mathbf{f} wieder skalarwertige Funktionen sind, entsprechend Def. 5.1.

- Skalar- und vektorwertige Funktionen mit mehreren Variablen werden auch **multivariat** genannt.
- Wie bei einem Vektor \mathbf{x} stellen wir zur besseren Unterscheidbarkeit vektorwertige Funktionen \mathbf{f} fett gedruckt dar, im Gegensatz zu einem Skalar x und einer skalarwertigen Funktion f .
- Wir werden uns bei der Lösung nichtlinearer Gleichungssysteme auf vektorwertige Funktionen $\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ konzentrieren.

Beispiele 5.1:

- Addition und Multiplikation:

Wir können die Addition (bzw. Subtraktion) und die Multiplikation (bzw. Division) auffassen als skalarwertige Funktionen $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ mit

$$\begin{aligned} f(x, y) &= x + y \\ g(x, y) &= x \cdot y \end{aligned}$$

- Ohmsches Gesetz:

Die an einem ohmschen Widerstand R abfallende Spannung U hängt vom Widerstand R und der Stromstärke I gemäss dem ohmschen Gesetz $U = R \cdot I$ ab. Also haben wir für die abhängige Variable $U = f(R, I) = RI$ die skalarwertige Funktion $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ mit den unabhängigen Variablen R und I . Häufig schreibt man auch direkt

$$U = U(R, I) = R \cdot I$$

und bringt dadurch die Abhängigkeit der Variable U von den unabhängigen Variablen R und I zum Ausdruck, wie wir es auch bereits vom eindimensionalen Fall kennen, z.B. $y = y(x)$.

- Reihenschaltung von Widerständen:

Bei der Reihenschaltung von n ohmschen Widerständen R_1, R_2, \dots, R_n ergibt sich der Gesamtwiderstand R gemäss

$$R = R(R_1, R_2, \dots, R_n) = R_1 + R_2 + \dots + R_n$$

4. Vektorwertige Funktionen:

Im einleitenden Beispiel in Kap. 5.1 haben wir bereits eine vektorwertige nichtlineare Funktion $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ kennengelernt. Ein weiteres Beispiel für $\mathbf{f} : \mathbb{R}^3 \rightarrow \mathbb{R}^4$ ist

$$f(x_1, x_2, x_3) = \begin{pmatrix} x_1^2 + x_2^2 \\ x_1^2 + x_3^2 \\ x_2^2 + x_3^2 \\ x_1^2 + x_2^2 + x_3^2 \end{pmatrix},$$

Aufgabe 5.1:

1. Geben Sie je ein konkretes (nicht zu kompliziertes) Beispiel einer skalarwertigen Funktion $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ sowie einer vektorwertigen Funktion $\mathbf{g} : \mathbb{R}^4 \rightarrow \mathbb{R}^3$ an.
2. Geben sie die lineare Funktion $\mathbf{f} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ an, für die die Lösung \mathbf{x} des linearen Gleichungssystems

$$\mathbf{Ax} = \mathbf{b} \text{ mit } \mathbf{A} = \begin{pmatrix} 4 & -1 & 1 \\ -2 & 5 & 1 \\ 1 & -2 & 5 \end{pmatrix} \text{ und } \mathbf{b} = \begin{pmatrix} 5 \\ 11 \\ 12 \end{pmatrix}$$

gerade $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ ergibt.

5.2.2 Darstellungsformen

Insbesondere bei der Interpretation von Resultaten einer Berechnung oder Messung ist eine geeignete Darstellungsform der Daten oft entscheidend für das eigene Verständnis und das Verständnis anderer. Während Funktionen mit einer unabhängigen Variablen noch einfach darstellbar sind, wird es bei Funktionen mehrerer Variablen schnell schwierig und unübersichtlich. Wir gehen hier auf die wichtigsten Darstellungsformen ein, ohne Anspruch auf Vollständigkeit.

5.2.2.1 Analytische Darstellung

Die Funktion liegt in Form einer Gleichung vor. Man unterscheidet:

- Explizite Darstellung: die Funktionsgleichung ist nach einer Variablen aufgelöst

$$y = f(x_1, x_2, \dots, x_n)$$

Beispiel: $y = 2 \cdot e^{x_1^2 + x_2^2}$

- Implizite Darstellung: die Funktionsgleichung ist nicht nach einer Variablen aufgelöst (deshalb handelt es sich hier um eine Funktion mit nur $n - 1$ unabhängigen Variablen ... warum?).

$$F(x_1, x_2, \dots, x_n) = 0$$

Beispiel: $x_1^2 + x_2^2 + x_3^2 - 1 = 0$

5.2.2.2 Darstellung durch Wertetabelle

Betrachten wir den Fall $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Setzt man in die (als bekannt) vorausgesetzte Funktionsgleichung $z = f(x, y)$ für die beiden unabhängigen Variablen x und y der Reihe nach bestimmte Werte ein, so erhält man eine Wertetabelle bzw. Matrix (siehe Abb. 5.2).

5.2.2.3 Grafische Darstellung

Wir beschränken uns hier auf skalarwertige Funktionen mit zwei unabhängigen Variablen $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, für die es noch anschauliche grafische Darstellungsmöglichkeiten gibt. Dazu betrachten wir die Funktion $z = f(x, y)$ in einem dreidimensionalen kartesischen Koordinatensystem mit den Koordinatenachsen x, y, z .

		2. unabhängige Variable y						
		y_1	y_2	\dots	y_k	\dots	y_n	
1. unabhängige Variable x	x_1	z_{11}	z_{12}	\dots	z_{1k}	\dots	z_{1n}	
	x_2	z_{21}	z_{22}	\dots	z_{2k}	\dots	z_{2n}	
	\vdots	\vdots	\vdots	\dots	\vdots	\dots	\vdots	
	x_i	z_{i1}	z_{i2}	\dots	z_{ik}	\dots	z_{in}	← i -te Zeile
	\vdots	\vdots	\vdots	\dots	\vdots	\dots	\vdots	
	x_m	z_{m1}	z_{m2}	\dots	z_{mk}	\dots	z_{mn}	
								\uparrow k -te Spalte

Abbildung 5.2: Wertetabelle für $z = f(x, y)$ für verschiedene Werte von x und y (aus [8]).

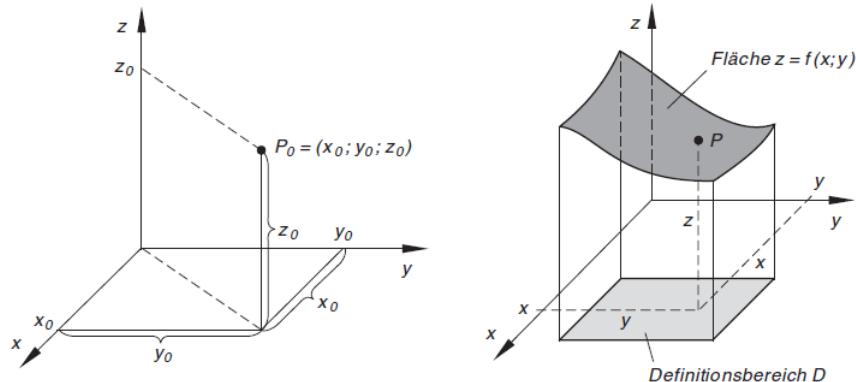


Abbildung 5.3: Links: kartesische Koordinaten eines Raumpunktes. Rechts: Darstellung einer Funktion $z = f(x, y)$ als Fläche im Raum (aus [8]).

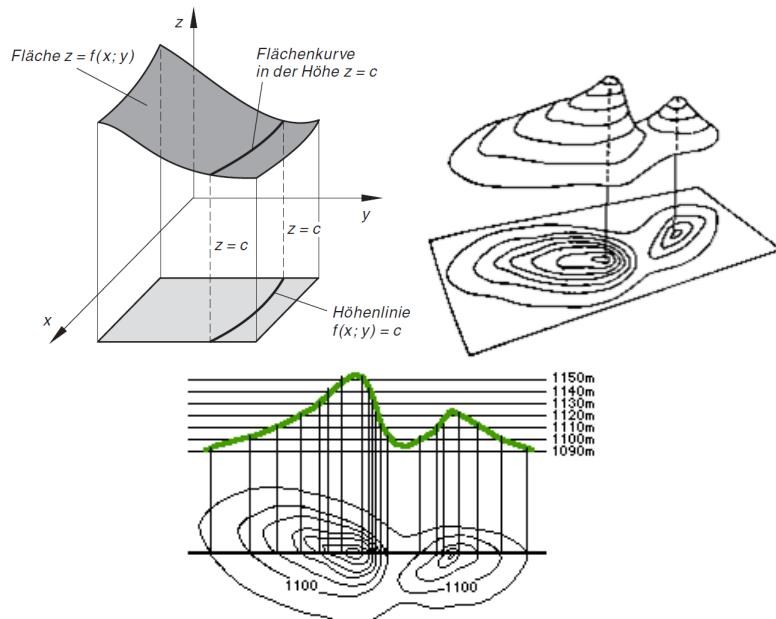


Abbildung 5.4: Zum Prinzip einer Höhenlinie (aus [8]) bzw. eines Höhenliniendiagramms.

- Darstellung einer Funktion als Fläche im Raum:

Die Funktion f ordnet jedem Punkt $(x, y) \in D$ in der Ebene einen Wert $z = f(x, y)$ zu, der als Höhenkoordinate verstanden werden kann. Durch die Anordnung der Punkte $(x, y, f(x, y))$ im dreidimensionalen Koordinatensystem wird eine über dem Definitionsbereich D liegende Fläche ausgezeichnet (siehe Abb. 5.3).

- Schnittkurvendiagramm

Wird die Fläche $z = f(x, y)$ bei einer konstanten Höhe $z = \text{const.}$ geschnitten, ergibt sich eine Schnittkurve. Wird diese in die (x, y) -Ebene projiziert, spricht man von einer Höhenlinie bzw. bei der Abbildung von einem Höhenliniendiagramm, wie wir es z.B. von Wanderkarten her kennen. Natürlich kann man auch andere Schnitte als $z = \text{const.}$ (Schnittebene parallel zur (x, y) -Ebene) wählen, z.B. $x = \text{const.}$ (Schnittebene parallel zur (y, z) -Ebene) oder $y = \text{const.}$ (Schnittebene parallel zur (x, z) -Ebene). Siehe Abb. 5.4.

5.2.3 Partielle Ableitungen

Für eine Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ mit einer Variablen ist die Ableitung an der Stelle x_0 bekanntlich definiert als

$$f'(x_0) = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x},$$

aus geometrischer Sicht entspricht dies der Steigung $m = f'(x_0)$ der im Punkt $(x_0, f(x_0))$ angelegten Kurventangente t mit der Tangentengleichung

$$t(x) = f(x_0) + f'(x_0)(x - x_0)$$

Wir werden die Definition der Ableitung jetzt erweitern auf Funktionen mit mehreren unabhängigen Variablen. Die Idee dabei ist, jede unabhängige Variable einzeln zu betrachten und sämtliche anderen unabhängigen Variablen "einzufrieren" (d.h. als festgelegte Parameter zu behandeln). So kann man ein n -dimensionales Problem auf n eindimensionale Probleme reduzieren und die obige Definition der Ableitung verwenden.

Betrachten wir der Einfachheit halber eine Funktion mit zwei unabhängigen Variablen $z = f(x, y)$ und auf der dadurch definierten Fläche den Punkt P mit den Koordinaten (x_0, y_0, z_0) , wobei $z_0 = f(x_0, y_0)$. Wir legen durch den Flächenpunkt P zwei Schnittebenen, die erste verläuft parallel zur (x, z) -Ebene, die zweite zur (y, z) -Ebene (Abb. 5.5). Wir erhalten so durch den Punkt P zwei Schnittkurven K_1 und K_2 auf der Fläche $z = f(x, y)$. Dabei hängt die Funktionsgleichung von K_1 nur noch von der Variablen x ab (d.h. für K_1 gilt $z = f(x, y_0) =: g(x)$, denn y_0 ist fixiert). Analog hängt die Funktionsgleichung von K_2 nur noch von der Variablen y ab (d.h. für K_2 gilt $z = f(x_0, y) =: h(y)$, denn x_0 ist fixiert). Die beiden Schnittkurven sind ebenfalls in Abb. 5.5 dargestellt.

Indem wir nun diese beiden Schnittkurven $g(x) = f(x, y_0)$ und $h(y) = f(x_0, y)$ gemäß unserer bisherigen Definition je einmal an der Stelle x_0 bzw. y_0 ableiten, erhalten wir die Steigung der Tangenten an die Fläche $z = f(x, y)$ im Punkt P , einmal in x -Richtung und einmal in y -Richtung. Konkret berechnen wir die beiden Grenzwerte:

$$\begin{aligned} g'(x_0) &= \lim_{\Delta x \rightarrow 0} \frac{g(x_0 + \Delta x) - g(x_0)}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x} =: \frac{\partial f}{\partial x}(x_0, y_0) \\ h'(y_0) &= \lim_{\Delta y \rightarrow 0} \frac{h(y_0 + \Delta y) - h(y_0)}{\Delta y} = \lim_{\Delta y \rightarrow 0} \frac{f(x_0, y_0 + \Delta y) - f(x_0, y_0)}{\Delta y} =: \frac{\partial f}{\partial y}(x_0, y_0) \end{aligned}$$

Wir bezeichnen diese Grenzwerte als partielle Ableitung 1. Ordnung von f an der Stelle (x_0, y_0) .

Definition 5.2 [8]: Partielle Ableitungen 1. Ordnung

Unter den partiellen Ableitungen 1. Ordnung einer Funktion $z = f(x, y)$ an der Stelle (x, y) werden die folgenden Grenzwerte verstanden (falls sie vorhanden sind):

- Partielle Ableitung 1. Ordnung nach x :

$$\frac{\partial f}{\partial x}(x, y) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}$$

- Partielle Ableitung 1. Ordnung nach y :

$$\frac{\partial f}{\partial y}(x, y) = \lim_{\Delta y \rightarrow 0} \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y}$$

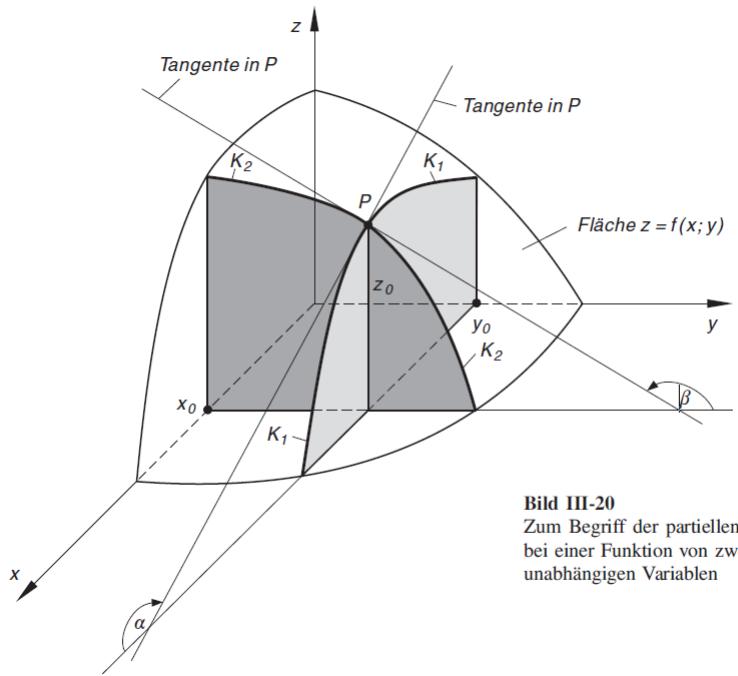


Bild III-20
Zum Begriff der partiellen Ableitung
bei einer Funktion von zwei
unabhängigen Variablen

Schnittkurve $K_1: z = f(x; y_0) = g(x)$

Schnittkurve $K_2: z = f(x_0; y) = h(y)$

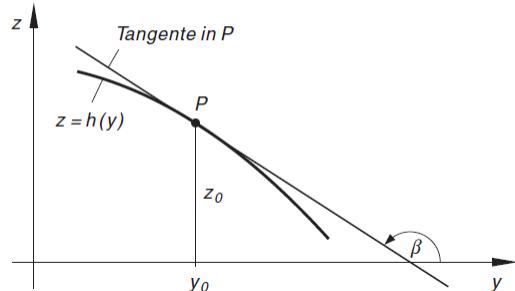
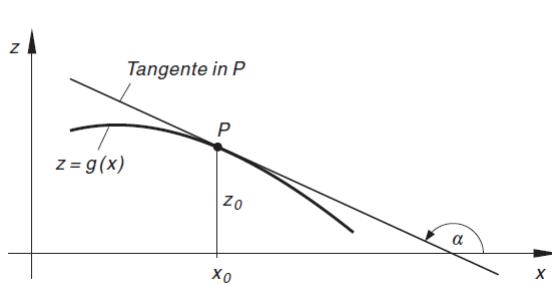


Abbildung 5.5: Fläche $z = f(x, y)$ mit dem Punkt $P = (x_0, y_0, z_0)$ und den Schnittflächen sowie den Schnittkurven K_1 und K_2 (aus [8]).

Bemerkungen:

1. Weitere übliche Symbole sind $f_x(x, y)$, $f_y(x, y)$ oder in abgekürzter Schreibweise f_x , f_y bzw. $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$.
 2. Geometrische Interpretation der partiellen Ableitungen der Funktion $z = f(x, y)$ an der Stelle (x_0, y_0) :
 - (a) $\frac{\partial f}{\partial x}(x_0, y_0)$ ist die Steigung der Flächentangente im Flächenpunkt $P = (x_0, y_0, z_0)$ in positiver x -Richtung
 - (b) $\frac{\partial f}{\partial y}(x_0, y_0)$ ist die Steigung der Flächentangente im Flächenpunkt $P = (x_0, y_0, z_0)$ in positiver y -Richtung
 3. Formal erhalten wir die partielle Ableitung $\frac{\partial f}{\partial x}(x, y)$, indem wir die Funktion $z = f(x, y)$ zunächst als eine nur von x abhängige Funktion betrachten und nach der Variablen x differenzieren. Während dieser Differentiation wird die Variable y als konstante Größe (Parameter) betrachtet. Für das Differenzieren selbst gelten dann die bereits aus der Analysis bekannten Ableitungsregeln für Funktionen von einer unabhängigen Variablen.
- Beispiel:
- $$\begin{aligned} z &= f(x, y) = 3xy^3 + 10x^2y + 5y + 3y \cdot \sin(5xy) \\ \frac{\partial f}{\partial x}(x, y) &= f_x(x, y) = 3 \cdot 1 \cdot y^3 + 10 \cdot 2x \cdot y + 0 + 3y \cdot \cos(5xy) \cdot 5 \cdot 1 \cdot y \end{aligned}$$

4. Analog erhalten wir die partielle Ableitung $\frac{\partial f}{\partial y}(x, y)$, indem wir die Funktion $z = f(x, y)$ zunächst als eine nur von y abhängige Funktion betrachten und nach der Variablen y differenzieren. Während dieser Differentiation wird die Variable x als konstante Größe (Parameter) betrachtet. Für das Differenzieren selbst gelten dann die bereits aus der Analysis bekannten Ableitungsregeln für Funktionen von einer unabhängigen Variablen.

- Beispiel:

$$\begin{aligned} z &= f(x, y) = 3xy^3 + 10x^2y + 5y + 3y \cdot \sin(5xy) \\ \frac{\partial f}{\partial y}(x, y) &= f_y(x, y) = 3x \cdot 3y^2 + 10x^2 \cdot 1 + 5 \cdot 1 + (3 \cdot 1 \cdot \sin(5xy) + 3y \cdot \cos(5xy) \cdot 5x \cdot 1) \end{aligned}$$

5. Die partielle Differentiation wird somit auf die gewöhnliche Differentiation, d. h. auf die Differentiation einer Funktion von einer Variablen zurückgeführt. Die Ableitungsregeln sind daher die gleichen wie bei den Funktionen einer Variablen. So lautet beispielsweise die Produktregel bei zwei unabhängigen Variablen, d. h. für eine Funktion vom Typ $z = f(x, y) = u(x, y) \cdot v(x, y)$:

- $f_x = u_x \cdot v + u \cdot v_x$
- $f_y = u_y \cdot v + u \cdot v_y$

6. Für Funktionen mit mehr als zwei unabhängigen Variablen geht man analog vor. Sei $y = f(x_1, x_2, \dots, x_n)$ eine Funktion mit n unabhängigen Variablen. Es lassen sich nun n partielle Ableitungen 1. Ordnung bilden gemäß

$$\frac{\partial f}{\partial x_k}(x_1, \dots, x_k, \dots, x_n) = \lim_{\Delta x_k \rightarrow 0} \frac{f(x_1, \dots, x_k + \Delta x_k, \dots, x_n) - f(x_1, \dots, x_k, \dots, x_n)}{\Delta x_k} \quad (k = 1, \dots, n)$$

Dabei werden wieder alle anderen Variablen ausser x_k als konstante Größe angenommen und es wird nach x_k abgeleitet.

Beispiel 5.2:

- Die partiellen Ableitungen erster Ordnung der Funktion

$$z = f(x, y) = 3xy^2 + \ln(x^3y^2)$$

lauten

$$\begin{aligned} f_x &= 3 \cdot 1 \cdot y^2 + \frac{1}{x^3y^2} \cdot 3x^2 \cdot y^2 \\ f_y &= 3x \cdot 2y + \frac{1}{x^3y^2} \cdot x^3 \cdot 2y \end{aligned}$$

wobei $f_x = f_x(x, y)$ und $f_y = f_y(x, y)$ natürlich wieder Funktionen von (x, y) sind. Die konkreten Steigungen der Tangenten an $f(x, y)$ an der Stelle $(x_0, y_0) = (-1, 1)$ in x - resp. y -Richtung lauten dann

$$\begin{aligned} f_x(-1, 1) &= 3 \cdot 1 \cdot 1^2 + \frac{1}{(-1)^3 \cdot 1^2} \cdot 3 \cdot (-1)^2 \cdot 1^2 = 0 \\ f_y(-1, 1) &= 3 \cdot (-1) \cdot 2 \cdot 1 + \frac{1}{(-1)^3 \cdot 1^2} \cdot (-1)^3 \cdot 2 \cdot 1 = -4 \end{aligned}$$

Aufgabe 5.2:

- Berechnen Sie die partiellen Ableitungen erster Ordnung für die folgenden Funktionen bei den vorgegebenen (x_0, y_0) -Werten.

1. $z = f(x, y) = x^2 y^4 + e^x \cdot \cos y + 10x - 2y^2 + 3$ an der Stelle $(x_0, y_0) = (0, 0)$
2. $z = f(x, y) = xy^2 \cdot (\sin x + \sin y)$ an der Stelle $(x_0, y_0) = (\frac{\pi}{2}, \pi)$
3. $z = f(x, y) = \ln(x + y^2) - e^{2xy} + 3x$ an der Stelle $(x_0, y_0) = (1, 0)$

5.2.4 Linearisierung von Funktionen mit mehreren Variablen

Wieder ausgehend vom eindimensionalen Fall haben wir für die an eine Funktion $y = f(x)$ im Punkt $(x_0, f(x_0))$ angelegten Kurventangente g die Tangentengleichung

$$g(x) = f(x_0) + f'(x_0)(x - x_0)$$

und wir wissen, dass in einer Umgebung von x_0 die Funktion $y = f(x)$ durch die lineare Tangente angenähert ('linearisiert') werden kann, also $f(x) \approx f(x_0) + f'(x_0)(x - x_0)$ gilt. Nun wollen wir dies auf Funktionen mit mehreren Variablen erweitern und führen dafür die sogenannte Jacobi-Matrix $Df(x)$ ein, welche die einfache Ableitung $f'(x)$ ersetzen wird.

Definition 5.3: Jacobi-Matrix / Linearisierung / Tangentialebene

- Sei $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mit $\mathbf{y} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} y_1 = f_1(\mathbf{x}) \\ y_2 = f_2(\mathbf{x}) \\ \vdots \\ y_m = f_m(\mathbf{x}) \end{pmatrix}$ und $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$. Die **Jacobi-Matrix** enthält sämtliche partiellen Ableitung 1. Ordnung von \mathbf{f} und ist definiert als

$$D\mathbf{f}(\mathbf{x}) := \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) & \cdots & \frac{\partial f_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \frac{\partial f_m}{\partial x_2}(x) & \cdots & \frac{\partial f_m}{\partial x_n}(x) \end{pmatrix}$$

- Die "verallgemeinerte Tangentengleichung"

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}^{(0)}) + D\mathbf{f}(\mathbf{x}^{(0)}) \cdot (\mathbf{x} - \mathbf{x}^{(0)})$$

beschreibt eine lineare Funktion und es gilt $\mathbf{f}(\mathbf{x}) \approx \mathbf{g}(\mathbf{x})$ in einer Umgebung eines gegebenen Vektors $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^T \in \mathbb{R}^n$. Man spricht deshalb auch von der **Linearisierung** der Funktion $\mathbf{y} = \mathbf{f}(\mathbf{x})$ in einer Umgebung von $\mathbf{x}^{(0)}$ (ein hochgestellter Index in Klammern $\mathbf{x}^{(k)}$ bezeichnet wie bisher einen Vektor aus \mathbb{R}^n nach der k -ten Iteration).

- Für den speziellen Fall $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ mit $y = f(x_1, x_2)$ und $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)})^T \in \mathbb{R}^2$ ist die Jacobi-Matrix nur ein Zeilenvektor mit zwei Elementen, nämlich

$$D\mathbf{f}(\mathbf{x}^{(0)}) = \left(\frac{\partial f}{\partial x_1}(x_1^{(0)}, x_2^{(0)}) \quad \frac{\partial f}{\partial x_2}(x_1^{(0)}, x_2^{(0)}) \right).$$

Dann liefert die Linearisierung

$$\begin{aligned} g(x_1, x_2) &= f(x_1^{(0)}, x_2^{(0)}) + \left(\frac{\partial f}{\partial x_1}(x_1^{(0)}, x_2^{(0)}) \quad \frac{\partial f}{\partial x_2}(x_1^{(0)}, x_2^{(0)}) \right) \cdot \begin{pmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \end{pmatrix} \\ &= f(x_1^{(0)}, x_2^{(0)}) + \frac{\partial f}{\partial x_1}(x_1^{(0)}, x_2^{(0)}) \cdot \begin{pmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \end{pmatrix} + \frac{\partial f}{\partial x_2}(x_1^{(0)}, x_2^{(0)}) \cdot \begin{pmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \end{pmatrix} \end{aligned}$$

die Gleichung der **Tangentialebene**. Sie enthält sämtliche im Flächenpunkt $P = (x_1^{(0)}, x_2^{(0)}, f(x_1^{(0)}, x_2^{(0)}))$ an die Bildfläche von $y = f(x_1, x_2)$ angelegten Tangenten.

Beispiele 5.3:

1. Betrachten wir konkret nochmals die Funktion $\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2 - 11 \\ x_1 + x_2^2 - 7 \end{pmatrix}$ aus dem einführenden Beispiel von Kap. 5.1 und linearisieren sie in der Umgebung von $\mathbf{x}^{(0)} = (1, 1)^T$. Für die Jacobi-Matrix erhalten wir

$$\mathbf{D}\mathbf{f}(x_1, x_2) = \begin{pmatrix} 2x_1 & 1 \\ 1 & 2x_2 \end{pmatrix}$$

und an der Stelle $\mathbf{x}^{(0)} = (1, 1)^T$ gilt

$$\mathbf{D}\mathbf{f}(x_1^{(0)}, x_2^{(0)}) = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

Damit erhalten wir

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= \mathbf{f}(\mathbf{x}^{(0)}) + \mathbf{D}\mathbf{f}(\mathbf{x}^{(0)}) \cdot (\mathbf{x} - \mathbf{x}^{(0)}) \\ \mathbf{g}(x_1, x_2) &= \begin{pmatrix} -9 \\ -5 \end{pmatrix} + \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 - 1 \\ x_2 - 1 \end{pmatrix} \\ &= \begin{pmatrix} -9 + 2(x_1 - 1) + (x_2 - 1) \\ -5 + (x_1 - 1) + 2(x_2 - 1) \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 - 12 \\ x_1 + 2x_2 - 8 \end{pmatrix} \end{aligned}$$

2. Betrachten wir konkret die Funktion $f(x_1, x_2) = x_1^2 + x_2^2$. Wir linearisieren sie in der Umgebung von $\mathbf{x}^{(0)} = (1, 2)^T$. Wir erhalten für die Jacobi-Matrix

$$\mathbf{D}\mathbf{f}(x_1, x_2) = \begin{pmatrix} 2x_1 & 2x_2 \end{pmatrix}$$

und damit

$$\begin{aligned} g(\mathbf{x}) &= f(\mathbf{x}^{(0)}) + \mathbf{D}\mathbf{f}(\mathbf{x}^{(0)}) \cdot (\mathbf{x} - \mathbf{x}^{(0)}) \\ g(x_1, x_2) &= 5 + \begin{pmatrix} 2 & 4 \end{pmatrix} \begin{pmatrix} x_1 - 1 \\ x_2 - 2 \end{pmatrix} \\ &= 5 + 2(x_1 - 1) + 4 \cdot (x_2 - 2) \\ &= 2x_1 + 4x_2 - 5. \end{aligned}$$

Dies ist nichts anderes als die Gleichung der Tangentialebene im kartesischen (x_1, x_2, x_3) -Koordinatensystem an die durch $x_3 = f(x_1, x_2) = x_1^2 + x_2^2$ definierte Fläche im Flächenpunkt $P = (1, 2, 5)$, wie in Abb. 5.6 dargestellt.

Aufgaben 5.3:

1. Linearisieren Sie für $\mathbf{x}^{(0)} = (\pi/4, 0, \pi)^T$ die Funktion

$$\mathbf{f}(x_1, x_2, x_3) = \begin{pmatrix} \sin(x_2 + 2x_3) \\ \cos(2x_1 + x_2) \end{pmatrix}$$

2. Berechnen Sie die Tangentialebene an der Stelle $\mathbf{x}^{(0)} = (0.5, 0.5)^T$ der Funktion

$$f(x_1, x_2) = e^{(x_1^2 + x_2^2)}$$

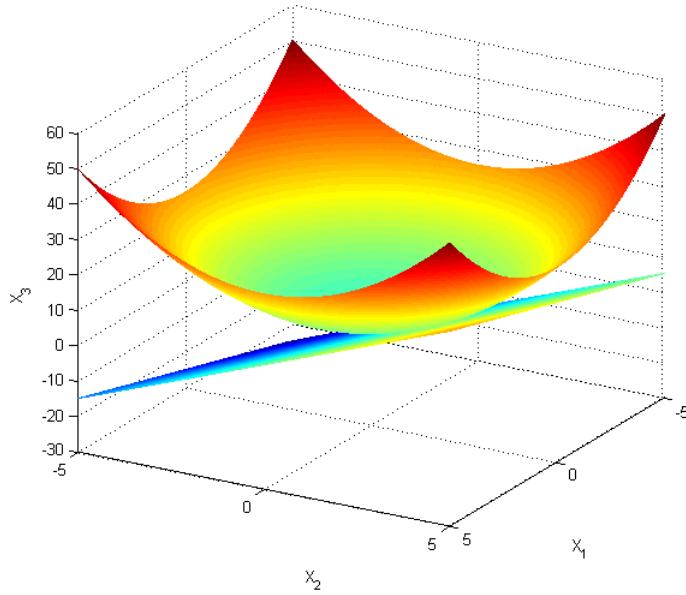


Abbildung 5.6: Grafische Darstellung der Fläche $x_3 = f(x_1, x_2) = x_1^2 + x_2^2$ sowie der Tangentialebene durch den Flächenpunkt (1,2,5) gemäss Bsp. 5.3.

5.3 Problemstellung zur Nullstellenbestimmung für nichtlineare Systeme

Die allgemeine Problemstellung zur Nullstellenbestimmung für nichtlineare Gleichungssysteme lautet:

Definition 5.4 [1]:

- Gegeben sei $n \in \mathbb{N}$ und eine Funktion $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Gesucht ist ein Vektor $\bar{\mathbf{x}} \in \mathbb{R}^n$ mit $\mathbf{f}(\bar{\mathbf{x}}) = 0$.
- Komponentenweise bedeutet dies: Gegeben sind n Funktionen $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, die die Komponenten von \mathbf{f} bilden. Gesucht ist ein Vektor $\bar{\mathbf{x}} \in \mathbb{R}^n$ mit $f_i(\bar{\mathbf{x}}) = 0$ (für $i = 1, \dots, n$). Dann heisst $\bar{\mathbf{x}} \in \mathbb{R}^n$ eine Lösung des **Gleichungssystems**

$$\mathbf{f}(x) = \mathbf{f}(x_1, \dots, x_n) = \begin{pmatrix} f_1(x_1, \dots, x_n) \\ f_2(x_1, \dots, x_n) \\ \vdots \\ f_n(x_1, \dots, x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Während es für lineare Gleichungssysteme relativ einfache Kriterien bezüglich der Lösbarkeit und der Anzahl von Lösungen gibt, ist diese Frage bei nichtlinearen Gleichungssystemen erheblich schwieriger zu beantworten, d.h. es gibt keine einfachen Methoden um festzustellen, ob ein nichtlineares Gleichungssystem lösbar ist und wie viele Lösungen es hat. Deshalb entscheidet die Wahl einer “geeigneten Startnäherung” meist über Erfolg oder Misserfolg der eingesetzten numerischen Verfahren.

Beispiel 5.4:

- Lösen Sie das folgende nichtlineare Gleichungssystem mit zwei Unbekannten:

$$\mathbf{f}(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8x_2^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- Lösung: Auflösen der ersten Gleichung nach x_2 und einsetzen in die zweite Gleichung liefert die drei Lösungen $(0, 0), (-2, 1), (2, -1)$.

5.4 Das Newton-Verfahren für Systeme

Im Abschnitt 3.4 haben wir für die Nullstellenbestimmung einer Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ mit einer Variablen das Newton-Verfahren hergeleitet. Aus der Linearisierung der Funktion f mittels der Tangente g in einer Umgebung der Stelle x_n

$$f(x) \approx g(x) = f(x_n) + f'(x_n)(x - x_n)$$

folgte die Iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (n = 0, 1, 2, 3, \dots).$$

In Definition 5.3 haben wir die Jacobi-Matrix und die Linearisierung eingeführt. Für den für uns interessanten Fall von $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ lautet die Jacobi-Matrix

$$\mathbf{D}\mathbf{f}(\mathbf{x}) := \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) & \cdots & \frac{\partial f_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \frac{\partial f_n}{\partial x_2}(x) & \cdots & \frac{\partial f_n}{\partial x_n}(x) \end{pmatrix}$$

und durch Linearisierung erhalten wir

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^{(n)}) + \mathbf{D}\mathbf{f}(\mathbf{x}^{(n)}) \cdot (\mathbf{x} - \mathbf{x}^{(n)}).$$

(wobei $\mathbf{x}^{(n)}$ wie üblich den Näherungs-Vektor für die Nullstelle \mathbf{x} nach der n -ten Iteration beschreibt). Wenn der Vektor $\mathbf{x}^{(n+1)}$ eine Nullstelle von \mathbf{f} ist, gilt:

$$\mathbf{f}(\mathbf{x}^{(n+1)}) = \mathbf{0} \approx \mathbf{f}(\mathbf{x}^{(n)}) + \mathbf{D}\mathbf{f}(\mathbf{x}^{(n)}) \cdot (\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}).$$

Durch Auflösen nach $\mathbf{x}^{(n+1)}$ erhalten wir dann die Iterationsvorschrift

$$\boxed{\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - (\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)}))^{-1} \cdot \mathbf{f}(\mathbf{x}^{(n)})}$$

wieder in Analogie zum eindimensionalen Fall

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Es wird aber nie die Inverse der Jacobi-Matrix berechnet, sondern die obige Gleichung wird zur Lösung eines linearen Gleichungssystems verwendet, indem man die Substitution

$$\boldsymbol{\delta}^{(n)} := -(\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)}))^{-1} \cdot \mathbf{f}(\mathbf{x}^{(n)})$$

als lineares Gleichungssystem auffasst gemäss

$$\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)})\boldsymbol{\delta}^{(n)} = -\mathbf{f}(\mathbf{x}^{(n)})$$

und so $\boldsymbol{\delta}^{(n)}$ bestimmen und anschliessend $\mathbf{x}^{(n+1)} := \mathbf{x}^{(n)} + \boldsymbol{\delta}^{(n)}$ berechnen kann.

5.4.1 Quadratisch-konvergentes Newton-Verfahren

Das obige Vorgehen führt zum folgenden Algorithmus:

Newton-Verfahren für Systeme [1]:

Gesucht sind Nullstellen von $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Sei $\mathbf{x}^{(0)}$ ein Startvektor in der Nähe einer Nullstelle. Das Newton-Verfahren zur näherungsweisen Bestimmung dieser Nullstelle lautet:

- für $n = 0, 1, \dots :$
 - Berechne $\boldsymbol{\delta}^{(n)}$ als Lösung des linearen Gleichungssystems
$$\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)})\boldsymbol{\delta}^{(n)} = -\mathbf{f}(\mathbf{x}^{(n)})$$
 - Setze
$$\mathbf{x}^{(n+1)} := \mathbf{x}^{(n)} + \boldsymbol{\delta}^{(n)}$$

Bemerkungen:

1. Mögliche Abbruchkriterien, $\epsilon > 0$ (gemäß [6]):

- (a) $n \geq n_{max}$, $n_{max} \in \mathbb{N}$
- (b) $\|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\| \leq \|\mathbf{x}^{(n+1)}\| \cdot \epsilon$
- (c) $\|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\| \leq \epsilon$
- (d) $\|\mathbf{f}(\mathbf{x}^{(n+1)})\| \leq \epsilon$

2. Es kann passieren, dass mit dem Newton-Verfahren statt einer Nullstelle von \mathbf{f} ein lokales Minimum \mathbf{x}_{min} gefunden wird, das ungleich $\mathbf{0}$ ist. In diesem Falle ist $\mathbf{D}\mathbf{f}(\mathbf{x}_{min})$ aber immer nicht regulär. Siehe untenstehendes Beispiel 5.6.

Beispiel 5.5 [1]:

- Wenden Sie das Newton-Verfahren auf das nichtlineare Gleichungssystem aus Beispiel 5.4 an:

$$\mathbf{f}(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8x_2^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- Lösung: Für die Jacobi-Matrix erhalten wir

$$\mathbf{D}\mathbf{f}(x_1, x_2) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_1, x_2) & \frac{\partial f_1}{\partial x_2}(x_1, x_2) \\ \frac{\partial f_2}{\partial x_1}(x_1, x_2) & \frac{\partial f_2}{\partial x_2}(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 4 & 24x_2^2 \end{pmatrix}$$

Wir wählen den Startvektor $\mathbf{x}^{(0)} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$. Daraus ergibt sich für die erste Iteration das lineare Gleichungssystem

$$\mathbf{D}\mathbf{f}(4, 2)\boldsymbol{\delta}^{(0)} = -\mathbf{f}(4, 2) \Rightarrow \begin{pmatrix} 2 & 4 \\ 4 & 96 \end{pmatrix} \boldsymbol{\delta}^{(0)} = -\begin{pmatrix} 16 \\ 80 \end{pmatrix} \Rightarrow \boldsymbol{\delta}^{(0)} = -\begin{pmatrix} \frac{76}{11} \\ \frac{11}{11} \end{pmatrix}$$

und für den ersten Newton-Schritt

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \boldsymbol{\delta}^{(0)} = \begin{pmatrix} -\frac{32}{11} \\ \frac{16}{11} \end{pmatrix} = \begin{pmatrix} -2.909\dots \\ 1.4545\dots \end{pmatrix}.$$

Die weiteren Schritte sind

i	0	1	2	3	4
$\mathbf{x}^{(i)}$	$\begin{pmatrix} 4 \\ 2 \end{pmatrix}$	$\begin{pmatrix} -2.909 \\ 1.455 \end{pmatrix}$	$\begin{pmatrix} -2.302 \\ 1.151 \end{pmatrix}$	$\begin{pmatrix} -2.051 \\ 1.025 \end{pmatrix}$	$\begin{pmatrix} -2.0018 \\ 1.0009 \end{pmatrix}$

Die Folge konvergiert gegen $(-2, 1)^T$, damit haben wir eine der drei Nullstellen gefunden.

Aufgabe 5.4:

- Programmieren Sie das obige Beispiel in Python und finden Sie Startvektoren, so dass das Newton-Verfahren mit diesen Startvektoren gegen die beiden anderen Nullstellen von \mathbf{f} konvergiert.

Tipp: Eine vektorwertige Funktion wie f kann in Python z.B. folgendermassen definiert werden:

```
def f(x):
    res = np.array ([[ float (2.*x[0]+4.*x[1])], [ float (4.*x[0]+8.*x[1]**3.)]])
    return res
```

Man sieht, dass das Newton-Verfahren konvergiert, wenn der Startvektor nahe genug bei einer Nullstelle liegt. Allgemein gilt:

Satz 5.1: Quadratische Konvergenz des Newton-Verfahrens für Systeme[1]

Das Newton-Verfahren konvergiert quadratisch für nahe genug an einer Nullstelle \bar{x} liegende Startvektoren, wenn $D\mathbf{f}(\bar{x})$ regulär und \mathbf{f} dreimal stetig differenzierbar ist.

Aufgabe 5.5 [1]:

- Das nichtlineare System

$$\mathbf{f}(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 5x_1^2 - x_2^2 \\ x_2 - 0.25(\sin x_1 + \cos x_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

besitzt in der Nähe von $\mathbf{x} = (0.25, 0.25)^T$ eine Lösung. Bestimmen Sie mit dem Newton-Verfahren eine Näherungslösung, die bezüglich der euklidischen Norm eine Genauigkeit von 10^{-5} besitzt.

Beispiel 5.6 [1]:

- Das Newtonverfahren für das nichtlineare System

$$\mathbf{f}(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^3 - x_2 - 1 \\ x_1^2 - x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

konvergiert für den Startvektor $\mathbf{x}^{(0)} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$ gegen $\mathbf{x} = \begin{pmatrix} 0 \\ -0.5 \end{pmatrix}$. Da aber $\mathbf{f}(\mathbf{x}) = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix} \neq 0$ ist das keine Nullstelle. Man sieht entsprechend, dass $D\mathbf{f}(\mathbf{x}) = \begin{pmatrix} 0 & -1 \\ 0 & -1 \end{pmatrix}$ nicht regulär ist.

5.4.2 Vereinfachtes Newton-Verfahren

Der Aufwand pro Schritt kann reduziert werden, wenn nicht bei jedem Schritt die Jacobi-Matrix $D\mathbf{f}(x^{(n)})$ ausgewertet, sondern immer wieder $D\mathbf{f}(x^{(0)})$ verwendet wird. Dies ist in Analogie zu Abschnitt 3.4.1 das vereinfachte Newtonverfahren:

Vereinfachtes Newton-Verfahren für Systeme [1]:

Gesucht sind Nullstellen von $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Sei $\mathbf{x}^{(0)}$ ein Startvektor in der Nähe einer Nullstelle. Das vereinfachte Newton-Verfahren zur näherungsweisen Bestimmung dieser Nullstelle lautet:

- für $n = 0, 1, \dots :$

- Berechne $\boldsymbol{\delta}^{(n)}$ als Lösung des linearen Gleichungssystems

$$D\mathbf{f}(\mathbf{x}^{(0)})\boldsymbol{\delta}^{(n)} = -\mathbf{f}(\mathbf{x}^{(n)})$$

- Setze

$$\mathbf{x}^{(n+1)} := \mathbf{x}^{(n)} + \boldsymbol{\delta}^{(n)}$$

Bemerkung:

- Das vereinfachte Newton-Verfahren konvergiert nur noch linear und nicht mehr quadratisch.

Beispiel 5.7 [1]:

Wenden Sie das vereinfachte Newton-Verfahren auf das nichtlineare Gleichungssystem aus Beispiel 5.4 an:

$$\mathbf{f}(x_1, x_2) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} 2x_1 + 4x_2 \\ 4x_1 + 8x_2^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Lösung: Wir wählen als Startvektor wieder $\mathbf{x}^{(0)} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$. Der erste Schritt des vereinfachten Newton-Verfahrens ist identisch mit dem ersten Schritt des Newton-Verfahrens. Im zweiten Schritt verwenden wir erneut die Jacobi-Matrix aus dem ersten Schritt; es ist also zu lösen

$$\begin{aligned} \mathbf{D}\mathbf{f}(\mathbf{x}^{(0)}) \boldsymbol{\delta}^{(1)} &= \mathbf{D}\mathbf{f}(4, 2) \boldsymbol{\delta}^{(1)} = -\mathbf{f}(\mathbf{x}^{(1)}) = -\mathbf{f}(-2.09, 1.45) \\ \iff \begin{pmatrix} 2 & 4 \\ 4 & 96 \end{pmatrix} \boldsymbol{\delta}^{(1)} &= -\begin{pmatrix} 6 \cdot 10^{-9} \\ -12.98 \end{pmatrix} \iff \boldsymbol{\delta}^{(1)} = \begin{pmatrix} 0.2951 \\ -0.1475 \end{pmatrix} \end{aligned}$$

Damit haben wir nach einem Newton-Schritt

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \boldsymbol{\delta}^{(1)} = \begin{pmatrix} -2.614 \\ 1.307 \end{pmatrix}$$

Einige weitere Iterierte:

i	0	1	2	5	10
$\mathbf{x}^{(i)}$	$\begin{pmatrix} 4 \\ 2 \end{pmatrix}$	$\begin{pmatrix} -2.909 \\ 1.455 \end{pmatrix}$	$\begin{pmatrix} -2.614 \\ 1.307 \end{pmatrix}$	$\begin{pmatrix} -2.258 \\ 1.129 \end{pmatrix}$	$\begin{pmatrix} -2.0817 \\ 1.041 \end{pmatrix}$

Offensichtlich konvergiert die Folge gegen $(-2, 1)^T$, jedoch deutlich langsamer als das Newton-Verfahren. ■

5.4.3 Gedämpftes Newton-Verfahren

Falls beim n -ten Iterationsschritt die Jacobi-Matrix $\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)})$ schlecht konditioniert (bzw. nicht oder fast nicht invertierbar) ist, kann wegen

$$\boldsymbol{\delta}^{(n)} := -\left(\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)})\right)^{-1} \cdot \mathbf{f}(\mathbf{x}^{(n)})$$

nicht generell erwartet werden, dass

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \boldsymbol{\delta}^{(n)}$$

eine bessere Näherung für die Nullstelle darstellt als $\mathbf{x}^{(n)}$. Unter Umständen entfernt sich in diesem Fall $\mathbf{x}^{(n+1)}$ sogar sehr weit von der eigentlichen Nullstelle, wie in Abb. 5.7 für einen eindimensionale Fall gezeigt ist. Falls dies der Fall ist, macht es Sinn,

$$\mathbf{x}^{(n)} + \boldsymbol{\delta}^{(n)}$$

zu verwerfen und es beispielsweise mit

$$\mathbf{x}^{(n)} + \frac{\boldsymbol{\delta}^{(n)}}{2}$$

zu probieren (d.h. wir verkleinern bzw. dämpfen die Schrittweite $\boldsymbol{\delta}^{(n)}$) und diesen Wert zu akzeptieren, sofern für die Länge des Vektors

$$\|\mathbf{f}\left(\mathbf{x}^{(n)} + \frac{\boldsymbol{\delta}^{(n)}}{2}\right)\|_2 < \|\mathbf{f}(\mathbf{x}^{(n)})\|_2$$

gilt, da wir ja eine Iteration von $\|\mathbf{f}(\mathbf{x}^{(n)})\|_2$ gegen 0 erreichen wollen. Das heisst, wir akzeptieren einen Iterationsschritt erst, wenn gilt

$$\|\mathbf{f}(\mathbf{x}^{(n+1)})\|_2 < \|\mathbf{f}(\mathbf{x}^{(n)})\|_2$$

Dies beschreibt der folgende Algorithmus:

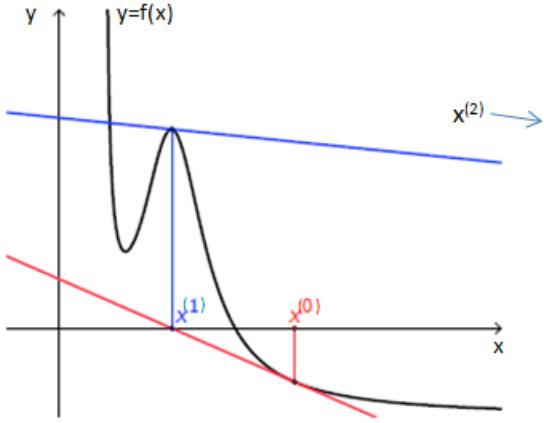


Abbildung 5.7: Eindimensionales Beispiel, in dem $x^{(1)}$ fast auf ein lokales Maximum zu liegen kommt und deshalb $f'(x^{(1)})$ beliebig klein bzw. $(f'(x^{(1)}))^{-1}$ beliebig gross wird. Die Iteration $x^{(2)}$ 'reißt' demzufolge aus und ist keine bessere Näherung für die Nullstelle von $f(x)$ als $x^{(1)}$.

Gedämpftes Newton-Verfahren für Systeme [6]:

Gesucht sind Nullstellen von $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Sei $\mathbf{x}^{(0)}$ ein Startvektor in der Nähe einer Nullstelle, $k_{max} \in \mathbb{N}$ sei vorgegeben. Das gedämpfte Newton-Verfahren zur näherungsweisen Bestimmung dieser Nullstelle lautet:

- für $n = 0, 1, \dots :$

- Berechne $\boldsymbol{\delta}^{(n)}$ als Lösung des linearen Gleichungssystems

$$\mathbf{D}\mathbf{f}(\mathbf{x}^{(n)})\boldsymbol{\delta}^{(n)} = -\mathbf{f}(\mathbf{x}^{(n)})$$

- Finde das minimale $k \in \{0, 1, \dots, k_{max}\}$ mit

$$\left\| \mathbf{f} \left(\mathbf{x}^{(n)} + \frac{\boldsymbol{\delta}^{(n)}}{2^k} \right) \right\|_2 < \left\| \mathbf{f} \left(\mathbf{x}^{(n)} \right) \right\|_2$$

- Falls kein minimales k gefunden werden kann, rechne mit $k = 0$ weiter
- Setze

$$\mathbf{x}^{(n+1)} := \mathbf{x}^{(n)} + \frac{\boldsymbol{\delta}^{(n)}}{2^k}$$

Bemerkungen [6]:

1. Natürlich kann man auch für das vereinfachte Newton-Verfahren die analoge Dämpfung einbauen.
2. Umfangreiche Tests haben ergeben, dass das gedämpfte Newton-Verfahren im Allgemeinen weit besser ist als das normale Newton-Verfahren oder andere, hier nicht behandelte Verfahren wie das Gradientenverfahren.
3. Die Dämpfungsgrösse k_{max} ist stark vom jeweiligen Problem abhängig. Das Verfahren kann bei gleichem Startvektor bei verschiedener Vorgabe von k_{max} einmal konvergieren und einmal divergieren; es kann insbesondere für verschiedene k_{max} auch gegen verschiedene Nullstellen konvergieren. Sofern nichts über sinnvolle Werte bekannt ist, kann zunächst mit $k_{max} = 4$ gerechnet werden.

Aufgabe 5.6:

- Der Druck, der benötigt wird, damit ein grosser, schwerer Gegenstand in einem weichen, auf einem harten Untergrund liegenden, homogenen Boden absinkt, kann über den Druck vorhergesagt werden, der zum Absinken

kleinerer Gegenstände in demselben Boden benötigt wird. Speziell der Druck p , der benötigt wird, damit ein runder flacher Gegenstand vom Radius r um d cm tief in den weichen Boden sinkt, kann über eine Gleichung der Form

$$p = k_1 e^{k_2 r} + k_3 r$$

approximiert werden, wobei k_1 , k_2 und k_3 Konstanten mit $k_2 > 0$ sind, die von d und der Konsistenz des Bodens, aber nicht vom Radius des Gegenstandes abhängen. Der harte Untergrund liege in einer Entfernung $D > d$ unter der Oberfläche.

a) Bestimmen Sie die Werte von k_1 , k_2 und k_3 , falls angenommen wird, dass ein Gegenstand vom Radius 1 cm einen Druck von 10 N/cm^2 benötigt, um 30 cm tief in einen schlammigen Boden zu sinken, ein Gegenstand vom Radius 2 cm einen Druck von 12 N/cm^2 benötigt, um 30 cm tief zu sinken und ein Gegenstand vom Radius 3 cm einen Druck von 15 N/cm^2 benötigt, um ebensoweit abzusinken (vorausgesetzt, der Schlamm ist tiefer als 30 cm). Benutzen Sie den Startvektor

$$\mathbf{k}^{(0)} = \begin{pmatrix} 10 \\ 0.1 \\ -1 \end{pmatrix}$$

b) Sagen Sie aufgrund Ihrer Berechnungen aus Übung a) die minimale Grösse eines runden Gegenstandes voraus, der eine Belastung von 500 N aushält und dabei weniger als 30 cm tief sinkt.

Kapitel 6

Ausgleichsrechnung

In der Auswertung von Daten ist man häufig mit dem Problem konfrontiert, Datenpunkte mit einer gewissen Streuung durch eine relativ einfache Funktion anzunähern. Dies geschieht mithilfe der sogen. Ausgleichsrechnung. Es handelt sich dabei um eine mathematische Optimierungsmethode, mit deren Hilfe für eine Reihe von Messdaten die unbekannten Parameter einer vorgegebenen Funktion bestimmt oder geschätzt werden können. Im allgemeinen handelt es sich dabei um überbestimmte Probleme, d.h. es liegen mehr Datenpunkte vor, als Parameter zu bestimmen sind (und damit mehr Gleichungen als Unbekannte). Bei der linearen Ausgleichsrechnung führt dies zu einem überbestimmten linearen Gleichungssystem (das wir mit den Methoden aus Kap. 4 lösen können), in der nicht-linearen Ausgleichsrechnung zu einem überbestimmten nichtlinearen Gleichungssystem (wo wir die Methoden aus Kap. 5 hinzuziehen). Typischer Weise haben überbestimmte Systeme keine oder keine eindeutige Lösung, es geht darum, eine optimale Funktion zu finden, die allfällige Fehler minimiert. In einem Spezialfall der linearen Ausgleichsrechnung jedoch, der Interpolation¹, erhalten wir ein eindeutig bestimmtes lineares Gleichungssystem mit einer eindeutigen Lösung (vgl. Abb. 6.1). In diesem Kapitel werden wir die Verfahren der linearen und nicht-linearen Ausgleichsrechnung kennen lernen.

Lernziele:

- Sie können mittels der Lagrange - Interpolationsformel eine Anzahl Messpunkte durch ein Polynom interpolieren.
- Sie können für vorgegebene Stützpunkte die natürliche kubische Splinefunktion berechnen.
- Sie können die Begriffe Ausgleichsproblem, Ansatzfunktion, Ausgleichsfunktion und Fehlerfunktional definieren.
- Sie kennen die Optimierung des Fehlerfunktional im Sinne der kleinsten Fehlerquadrate (least squares).
- Sie können das lineare sowie das allgemeine Ausgleichsproblem definieren und für spezifische Beispiele lösen.
- Sie können das gedämpfte Gauss-Newton Verfahren anwenden und in Python implementieren.

6.1 Zur historischen Entwicklung²

Die ersten Verfahren zur Interpolation, einem Spezialfall der linearen Ausgleichsrechnung, gehen zurück in die frühesten Anfänge der Astronomie bzw. dem Versuch, anhand der jährlichen Veränderungen am Firmament Kalender zu entwickeln und Voraussagen zu ermöglichen. Die Positionen von astronomischen Objekten wie Sonne, Planeten, Kometen etc. wurden hierzu in sogen. Ephemeriden (Positionstabellen) festgehalten. In den Keilschriften der

¹Das Wort “Interpolation” stammt vom lateinischen “interpolare” (auffrischen, umgestalten, verfälschen) und bezeichnet den Versuch, fehlende Daten aufgrund bestehender Daten zu schätzen (bzw. zu interpolieren).

²Gemäß Wikipedia:http://de.wikipedia.org/wiki/Methode_der_kleinsten_Quadrat Gemäß und Erik Merijering (2002), “A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing.”

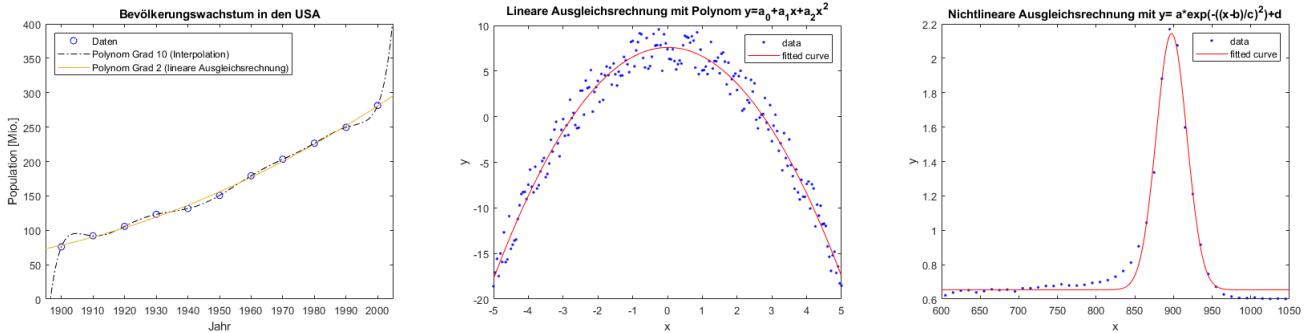


Abbildung 6.1: Drei Beispiele für Ausgleichsrechnung. Links: Die Bevölkerungszahlen im Zeitraum 1900-2000 (mit 11 Datenpunkten) lassen sich durch ein Polynom vom Grad 10 exakt interpolieren. Das Polynom geht also exakt durch jeden einzelnen Datenpunkt, wie es bei einer Interpolation gefordert ist. An den Rändern des Intervalls beginnt das Polynom aber stark zu oszillieren, was nicht ideal ist. Die Datenreihe lässt sich auch mit linearere Ausgleichsrechnung durch ein Polynom niedrigeren Grades annähern. Das gezeigte Polynom 2. Grades geht nicht mehr exakt durch die Datenpunkte, liefert dafür aber ein stabiles Verhalten an den Rändern. Mitte: ein weiteres Beispiel linearer Ausgleichsrechnung. Die 200 Datenpunkte werden durch ein Polynom 2. Grades angenähert. Rechts: Ein Beispiel für die nichtlineare Ausgleichsrechnung. Die Datenpunkte werden durch eine Gaussfunktion der Art $f(x) = a \cdot e^{-(x-b)/c^2} + d$ angenähert. Die Parameter a, b, c, d werden mittels der nichtlinearen Ausgleichsrechnung bestimmt.

Babylonier bzw. eines Nachfolgereichs der Seleukiden (3.-1. Jhr. v.Chr.) wurde basierend auf solchen Ephemeriden Interpolations-Verfahren (lineare Interpolation, aber offenbar auch komplexere Verfahren) verwendet, um Lücken zu füllen.

Die griechischen Astronomen Hipparchos (190-120 v.Chr.) und Ptolomäus (100-160 n.Chr.) verwendeten lineare Interpolation in ihren astronomischen Werken. Das Hauptwerk von Ptolomäus, der Almagest, zementierte das geozentrische Weltbild in Europa bis ins 16. Jahrhundert. Beispiele von polynominaler Interpolation (z.B. die Bestimmung einer Parabel durch drei gegebene Punkte) finden sich in mehreren chinesischen (6. Jhr. n.Chr.), persischen und arabischen Werken (11. und 15. Jhr. n.Chr.). In Europa lieferten die Astronomen Nikolaus Kopernikus (1473-1543), Johannes Kepler (1572-1630) und Galileo Galilei (1564-1642) wichtige Beiträge zur Theorie der Interpolation, welche in den Verfahren von Issac Newton (1643-1727) weiterentwickelt wurden. So stellte Newton das Problem "To find a curved line of the parabolic kind which shall pass through any given number of points" (hier ist parabolisch gleichzusetzen mit polynomial) und wendete dessen Lösung im darauffolgenden Lemma an " Certain observed places of a comet being given, to find the place of the same at any intermediate given time" (vgl. Abb. 6.2).

Eine elegante alternative Formulierung von Newtons Interpolationsformeln gelang 1795 dem italienischen Astronom und Mathematiker Giuseppe Lodovico Lagrangia (1736-1813), besser bekannt als Joseph-Louis Lagrange (siehe Kap. 6.2.2 zur Lagrange-Interpolationsformel). Weiterentwicklungen bzw. Verallgemeinerungen wurden von den französischen Mathematikern Augustin-Louis Cauchy (1789-1857) und Charles Hermite (1822-1901) publiziert.

Ergänzend zur Theorie der Polynom-Interpolation wurde der Ansatz der stückweisen Interpolation durch sogen. Splines (siehe Kap. 6.2.3) entwickelt. Zu den Pionieren der Splineforschung gehörte der rumänisch-amerikanischen Mathematiker Isaac Jacob Schoenberg (1903-1990). Der Begriff Spline beschreibt hier eine glatte mathematische Kurve, stückweise zusammengesetzt aus Polynomen höherer (z.B. dritter) Ordnung. Im Schiffbau ist ein Spline eine lange dünne Latte, die an einzelnen Punkten fixiert wird und z.B. die Form des Schiffsrumpfes definiert. Splines und weitere Interpolationstechniken kommen heute vor allem in der Signal- und Bildverarbeitung zum Einsatz.

Die Ausgleichsrechnung geht zurück auf die Methode der kleinsten Quadrate von Gauss (1777-1855). Am Neujahrstag 1801 entdeckte der italienische Astronom Giuseppe Piazzi (1746-1826) den Zwergplaneten Ceres. 40 Tage lang konnte er die Bahn verfolgen, dann verschwand Ceres hinter der Sonne. Im Laufe des Jahres versuchten viele Wissenschaftler erfolglos, anhand von Piazzi's Beobachtungen die Bahn zu berechnen – unter der Annahme einer Kreisbahn, denn nur für solche konnten damals die Bahnelemente aus beobachteten Himmelspositionen mathematisch ermittelt werden. Der 24-jährige Gauß hingegen konnte auch elliptische Bahnen aus drei Einzelbeobachtungen berechnen. Da aber deutlich mehr Bahnpunkte vorlagen, wandte er seine Methode der kleinsten Quadrate an, um so die Genauigkeit zu erhöhen. Als Astronomen im Dezember 1801 den Kleinplaneten genau an dem von Gauß vorhergesagten Ort wiederfanden, war das nicht nur ein großer Erfolg für Gauß' Methode: Piazzi's Ruf, der aufgrund

COR. 2. But their orbits will so near to parabolas, that parabolas may be used for them without sensible error.

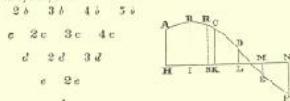
COR. 3. And, therefore, by Cor. 7, Prop. XVI Book I, the velocity of every comet will always be to the velocity of any planet, supposed to be revolved at the same distance in a circle about the sun, nearly in the subduplicate proportion of double the distance of the planet from the *center* of the sun to the distance of the comet from the sun's center, very nearly. Let us suppose the radius of the *orbis magnus*, or the greatest semi-diameter of the ellipse which the earth describes, to consist of 10000000 parts; and then the earth by its mean diurnal motion will describe 1729312 of those parts, and 13753 by its hourly motion. And therefore the comet, at the same mean distance of the sun, with a velocity which is to the velocity of the earth as $\sqrt{2}$ to 1, would by its diurnal motion describe 2123747 parts and 10234 parts by its hourly motion. But at greater or less distances than the diurnal and hourly motion will be to this diurnal and hourly motion in the reciprocal subduplicate proportion of the distances and is therefore given.

COR. 4. Wherefore if the *latens rectus* of the planet is quadruple of the radius of the *orbis magnus*, and the square of that radius is supposed to consist of 10000000 parts, the area which the comet will daily describe by a radius drawn to the sun will be 1216373 parts, and the hourly area will be 30382 parts. But if the *latens rectus* is greater or less in any proportion, the diurnal and hourly area will be less or greater in the subproportion of the same proportion reciprocally.

LEMMA V.

To find a curve line of the parabolic kind which shall pass through any given number of points.

Let those points be A, B, C, D, E, F &c., and from the sun to any right line HN, g vpon in position, let fall as many perpendiculars AH, BI, CK, DL, EM, FN &c.



CASE 1. If HI, IK, KL, LM, &c., the intervals of the points H, I, K, L, M, N, &c., are equal, take $\frac{b}{2b}, \frac{3b}{3b}, \frac{4b}{4b}, \frac{5b}{5b}$, &c., the first differences of the perpendiculars AH, BI, CK, &c.; their second differences $\frac{c}{2c}, \frac{3c}{3c}, \frac{4c}{4c}$, &c.; their third differences $\frac{d}{2d}, \frac{3d}{3d}, \frac{4d}{4d}$, &c.; their fourth differences, divided by the intervals between every four; and so forth; that is, in such manner,

$CK - 2b, CK - DL - 3b, DL - EM - 4b, EM - PN - 5b$, &c., then $b - 2b = c$, &c., and so on to the last differences, which is here f . Then, erecting my perpendicular RS, which may be considered as an ordinate of the curve required, in order to find the length of this ordinate, suppose the intervals HI, IK, KL, LM, &c., to be unity, and let AH = a , $- HS = p$, $\frac{1}{2}f$ into $- IS = q$, $\frac{1}{2}g$ into $- SK = r$, $\frac{1}{2}h$ into $- SL = s$, $\frac{1}{2}i$ into $- SM = t$; proceeding, to wit, to ME, the last perpendicular but one, and prefixing negative signs before the terms HS, IS, &c., which lie from S towards A; and affirmative signs before the terms SK, SL, &c., which lie on the other side of the point S; and observing well the signs, RS will be $= a + b + cq + dr + es + ft + &c.$

CASE 2. But if HI, IK, &c., the intervals of the points H, I, K, L, &c., are unequal, take $\frac{b}{2b}, \frac{3b}{3b}, \frac{4b}{4b}, \frac{5b}{5b}$, &c., the first differences of the perpendiculars AH, BI, CK, &c., divided by the intervals between those perpendiculars; $\frac{c}{2c}, \frac{3c}{3c}, \frac{4c}{4c}$, &c., their second differences, divided by the intervals between every two; $\frac{d}{2d}, \frac{3d}{3d}, \frac{4d}{4d}$, &c., their third differences, divided by the intervals between every three; $\frac{e}{2e}, \frac{3e}{3e}, \frac{4e}{4e}$, &c., their fourth differences, divided by the intervals between every four; and so forth; that is, in such manner, that b may be $= \frac{AH - BI}{HI}, 2b = \frac{BI - CK}{IK}, 3b = \frac{CK - DL}{KL}, \text{ &c.}$, then $c = \frac{b - 2b}{HI}, 2c = \frac{2b - 3b}{IK}, 3c = \frac{3b - 4b}{KL}, \text{ &c.}$, then $d = \frac{b - 2c}{HI}, 2d = \frac{2c - 3c}{IK}, 3d = \frac{3c - 4c}{KL}, \text{ &c.}$ And those differences being found, let AH be $= a$, $- HS = p$, $\frac{1}{2}f$ into $- IS = q$, $\frac{1}{2}g$ into $- SK = r$, $\frac{1}{2}h$ into $- SL = s$, $\frac{1}{2}i$ into $- SM = t$; proceeding, to wit, to ME, the last perpendicular but one; and the ordinate RS will be $= a + bp + cq + dr + es + ft + &c.$

CASE 3. Hence the areas of all curves may easily be found; for if some number of points of the curve to be squared are found, and a parabola be supposed to be drawn through those points, the area of this parabola will be nearly the same with the area of the curvilinear figure proposed to be squared; but the parabola can always be squared geometrically by methods vulgarly known.

LEMMA VI.

Given observed places of a comet being given, to find the place of the sun to use in each given time.

Let HI, IK, KL, LM (in the preceding Fig.) represent the times between the observations; HA, IE, KC, LD, ME, the observed longitudes of the comet; and HS the given time between the first observation and the longitude required. Then if a regular curve ABCDE is supposed to be drawn through the points A, B, C, D, E, and the ordinate RS is found out by the preceding lemma, RS will be the longitude required.

Abbildung 6.2: Auszüge aus Isaac Newtons Hauptwerk “Mathematical Principles of Natural Philosophy” (1687) zur Polynominterpolation.

seiner nicht zu einer Kreisbahn passen wollenden Bahnpunkte stark gelitten hatte, war ebenfalls wiederhergestellt.

Die Grundlagen seines Verfahrens hatte Gauß schon 1795 im Alter von 18 Jahren entwickelt. Basis war eine Idee des französischen Mathematikers Pierre-Simon Laplace (1749-1827), die Beträge von Fehlern aufzusummen, so dass sich die Fehler zu Null addieren. Gauß nahm stattdessen die Fehlerquadrate und konnte die Nullsummen-Anforderung an die Fehler weglassen. Unabhängig davon entwickelte der französische Mathematiker Adrien-Marie Legendre (1752-1833) dieselbe Methode erstmals im Jahre 1805 am Schluss eines kleinen Werkes über die Berechnung der Kometenbahnen und veröffentlichte eine zweite Abhandlung darüber im Jahr 1810. Von ihm stammt der Name Méthode des moindres carrés (Methode der kleinsten Quadrate).

1809 publizierte Gauß dann im zweiten Band seines himmelsmechanischen Werkes “Theoria motus corporum coelestium in sectionibus conicis solem ambientium” (Theorie der Bewegung der Himmelskörper, welche in Kegelschnitten die Sonne umlaufen) sein Verfahren, inklusive der Normalgleichungen und des Gaußschen Eliminationsverfahrens. Dabei erwähnte er, dass er es schon vor Legendre entdeckt und benutzt habe, was zu einem Prioritätenstreit zwischen den beiden führte. Die Methode der kleinsten Quadrate wurde nun schnell das Standardverfahren zur Behandlung von astronomischen oder geodätischen Datensätzen.

Gauß benutzte dann das Verfahren intensiv bei seiner Vermessung des Königreichs Hannover durch Triangulation. 1821 und 1823 erschien die zweiteilige Arbeit sowie 1826 eine Ergänzung zur Theoria combinationis observationum erroribus minimis obnoxiae (Theorie der den kleinsten Fehlern unterworfenen Kombination der Beobachtungen), in denen Gauß eine Begründung liefern konnte, weshalb sein Verfahren im Vergleich zu den anderen so erfolgreich war: Die Methode der kleinsten Quadrate ist in einer breiten Hinsicht optimal, also besser als andere Methoden. Die genaue Aussage ist als der Satz von Gauß-Markow bekannt, da die Arbeit von Gauß wenig Beachtung fand und schließlich im 20. Jahrhundert vom russischen Mathematiker Andrei Andrejewitsch Markow (1856-1922) wiederentdeckt und bekannt gemacht wurde. Theoria Combinationis enthält ferner wesentliche Fortschritte beim effizienten Lösen der auftretenden linearen Gleichungssysteme, wie das Gauß-Seidel-Verfahren und die LR-Zerlegung.

Der französische Vermessungsoffizier und Mathematiker André-Louis Cholesky (1875-1918) entwickelte während des Ersten Weltkrieges die Cholesky-Zerlegung, die gegenüber den Lösungsverfahren von Gauß nochmal einen erheblichen Effizienzgewinn darstellte. In den 1960er Jahren entwickelte der amerikanische Mathematiker Gene Golub (1932-2007) die Idee, die auftretenden linearen Gleichungssysteme mittels QR-Zerlegung zu lösen.

6.2 Interpolation

Die Interpolation ist ein Spezialfall der linearen Ausgleichsrechnung, bei dem wir zu einer Menge von vorgegebenen Punkten eine Funktion suchen, die exakt (und nicht nur näherungsweise) durch diese Punkte verläuft.

6.2.1 Problemstellung

Bei der Interpolation geht es z.B. darum, bei einer Wertetabelle, die Lücken aufweist, die fehlenden Funktionswerte anzunähern. Es sei also eine Wertetabelle einer Funktion f mit $y_i = f(x_i)$ der Art

x	x_0	x_1	x_2	...	x_{j-1}	x_j	x_{j+1}	...	x_n
y	y_0	y_1	y_2	...	y_{j-1}	?	y_{j+1}	...	y_n

gegeben. Die $n + 1$ Wertepaare (x_i, y_i) heißen Stützpunkte, die x_i Stützstellen und die y_i Stützwerte. Gesucht ist nun eine möglichst gute Näherung des fehlenden Wertes y_j , d.h. wir möchten eine (stetige) Funktion finden, welche die eigentliche Funktion $f(x)$ an der Stelle $f(x_j)$ möglichst gut approximiert und exakt durch die bekannten Stützpunkte in der Umgebung von x_j geht. Eine solche Funktion nennt man Interpolierende. Interpolation kommt vor allem zur Anwendung, wenn die Funktion $f(x)$ nicht genügend bekannt oder nur schwer exakt zu berechnen ist. Auch in der digitalen Bildbearbeitung wird häufig interpoliert.

Wir definieren also:

Definition 6.1: Interpolationsproblem [1]

- Gegeben sind $n + 1$ Wertepaare (x_i, y_i) , $i = 0, \dots, n$, mit $x_i \neq x_j$ für $i \neq j$. Gesucht ist eine stetige Funktion g mit der Eigenschaft $g(x_i) = y_i$ für alle $i = 0, \dots, n$.

Beispiele 6.1:

1. Aus einer Temperaturmessung ergeben sich im Tagesverlauf die Werte. Gesucht ist zum Beispiel die Temperatur um 11 Uhr.

t	08.00	10.00	12.00	14.00	16.00	18.00	Uhr
T	11.2	13.4	15.3	19.5	18.9	16.1	°C

2. Wir betrachten die (nicht einfach zu berechnende) Funktion $f(x) = \int_0^x e^{\sin t} dt$, welche in tabellierter Form vorliegt. Gesucht ist der Wert $f(0.66) = ?$

x	0.4	0.5	0.6	0.7	0.8	0.9
y	0.4904	0.6449	0.8136	0.9967	1.1944	1.4063

3. Ein Bild mit der Auflösung 800×800 Pixel soll vergrößert werden auf die neue Auflösung 1200×1200 . Die zusätzlichen Pixel müssen 'gefüllt' bzw. interoliert werden. Dabei werden nicht neue Informationen erzeugt, sondern nur die bestehenden Pixel sozusagen 'gestreckt'. Das vergrößerte Bild erscheint deshalb unscharf, aber dafür ohne Lücken.³



Die Interpolation geschieht also nach dem folgenden Prinzip: man sucht eine Interpolierende $g(x)$, welche leicht berechenbar sein soll und für die gilt

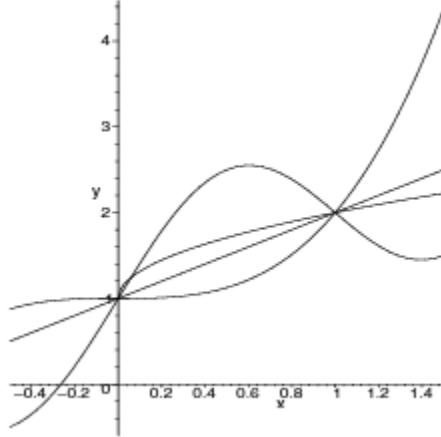
$$g(x_i) = f(x_i)$$

für einige gegebenen Stützstellen in der Nähe von x_j . Es ist offensichtlich, dass bei einer solchen Fragestellung unendlich viele Lösungen möglich sind, wie im folgenden Beispiel gezeigt:

³Bild übernommen von <http://de.wikipedia.org/w/index.php?title=Datei:ReconstructionsFromPixels.png&filetimestamp=20070524010356>

Beispiel 6.2:

- Es soll eine Interpolierende für die beiden Stützpunkte $(0, 1)$ und $(1, 2)$ gefunden werden.
- Lösung: im einfachsten Fall könnte man eine Gerade durch die beiden Punkte legen, also $g(x) = x + 1$. Aber auch $g(x) = \sqrt{x} + 1$, $g(x) = x^3 + 1$ oder $g(x) = \sin(\pi x) + x + 1$ sind Lösungen, wie in der folgenden Abbildung (gemäß [1]) dargestellt.



Im Folgenden betrachten wir den Fall, bei dem die Interpolierende ein Polynom ist.

6.2.2 Polynominterpolation

Gegeben sind $n + 1$ Stützpunkte

$$\begin{array}{c|ccccc} x & x_0 & x_1 & x_2 & \dots & x_n \\ \hline y & y_0 & y_1 & y_2 & \dots & y_n \end{array}$$

und gesucht ist ein Polynom $P_n(x)$, welches diese Punkte interpoliert. Wenn wir uns das Polynom in der üblichen Form nach Potenzen von x entwickelt denken, dann ist

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n.$$

Folglich gibt uns jeder Stützpunkt eine lineare Gleichung für die Bestimmung der Koeffizienten a_k . Wenn der Grad des Polynoms gleich n gewählt wird, erhalten wir ein lineares Gleichungssystem mit gleich vielen Gleichungen wie unbekannten Koeffizienten:

$$\begin{aligned} P_n(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_n x_0^n &= y_0 \\ P_n(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_n x_1^n &= y_1 \\ &\vdots && \vdots \\ P_n(x_n) &= a_0 + a_1 x_n + a_2 x_n^2 + \dots + a_n x_n^n &= y_n \end{aligned}$$

bzw.

$$\begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}.$$

Die hierbei auftretende Matrix wird Vandermonde-Matrix genannt. Im Prinzip könnten wir dieses lineare Gleichungssystem mit den uns bekannten Verfahren nach den Koeffizienten nun auflösen. Allerdings ist dies unnötig aufwendig und die Matrix des Systems ist typischerweise schlecht konditioniert. Durch eine andere Darstellung des Polynoms P_n lässt sich die Berechnung vereinfachen. Dazu ist zu sagen, dass die Polynom-Interpolation früher bei der manuellen Rechnung einen wichtigen Stellenwert hatte. Allerdings ist die Berechnung für grosse $n \gtrsim 20$ numerisch instabil. Aus diesem Grund kommen, wenn viele Stützpunkte berücksichtigt werden müssen, neuere

Techniken wie die Spline-Interpolation zum Zug. Zur Einführung konzentrieren wir uns aber vorderhand auf die Polynom-Interpolation.

Die Existenz, Eindeutigkeit und Konstruktion des Interpolationspolynoms ist in folgendem Satz gegeben:

Satz 6.1: Lagrange Interpolationsformel [2]

- Durch $n + 1$ Stützpunkte mit verschiedenen Stützstellen (d.h. $x_i \neq x_j$ für $i \neq j$) gibt es genau ein Polynom $P_n(x)$ vom Grade $\leq n$, welches alle Stützpunkte interpoliert, d.h. wo gilt

$$P_n(x_i) = y_i, \quad i = 0, 1, \dots, n$$

- $P_n(x)$ lautet in der Lagrangeform

$$P_n(x) = \sum_{i=0}^n l_i(x)y_i,$$

dabei sind die $l_i(x)$ die Lagrange-Polynome vom Grad n definiert durch

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \quad (i = 0, 1, \dots, n)$$

Beispiel 6.3:

- Wir nehmen die Temperaturmessung aus Beispiel 6.1 und bestimmen die Temperatur um 11 Uhr. Um die Rechnung etwas zu vereinfachen, benutzen wir nur die ersten vier Messwerte.

t	08.00	10.00	12.00	14.00	Uhr
T	11.2	13.4	15.3	19.5	$^{\circ}\text{C}$

- Lösung:

- Zur Demonstration bestimmen wir das Interpolationspolynom vollständig (d.h. für beliebige x -Werte). Dies ist aufwendig. In Anwendungen würde man nur den gesuchten x -Wert benutzen und direkt einsetzen (also $x = 11$, siehe weiter unten).
- Wir haben $n + 1 = 4$ Stützpunkte, also ist $n = 3$ und das Interpolationspolynom hat die Form

$$P_n(x) = \sum_{i=0}^3 l_i(x)y_i = 11.2 \cdot l_0(x) + 13.4 \cdot l_1(x) + 15.3 \cdot l_2(x) + 19.5 \cdot l_3(x).$$

Die Lagrange-Polynome berechnen sich zu

$$\begin{aligned} l_0(x) &= \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} = \frac{(x - 10)(x - 12)(x - 14)}{(-2)(-4)(-6)} = -\frac{1}{48}x^3 + \frac{3}{4}x^2 - \frac{107}{12}x + 35 \\ l_1(x) &= \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} = \frac{(x - 8)(x - 12)(x - 14)}{(2)(-2)(-4)} = +\frac{1}{16}x^3 - \frac{17}{8}x^2 + \frac{47}{2}x - 84 \\ l_2(x) &= \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} = \frac{(x - 8)(x - 10)(x - 14)}{(4)(2)(-2)} = -\frac{1}{16}x^3 + 2x^2 - \frac{83}{4}x + 70 \\ l_3(x) &= \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} = \frac{(x - 8)(x - 10)(x - 12)}{(6)(4)(2)} = +\frac{1}{48}x^3 - \frac{5}{8}x^2 + \frac{37}{6}x - 20 \end{aligned}$$

- Damit ergibt sich

$$\begin{aligned} P_n(x) &= 11.2 \cdot l_0(x) + 13.4 \cdot l_1(x) + 15.3 \cdot l_2(x) + 19.5 \cdot l_3(x) \\ &= +\frac{13}{240}x^3 - \frac{133}{80}x^2 + \frac{2137}{120}x - \frac{263}{5} \end{aligned}$$

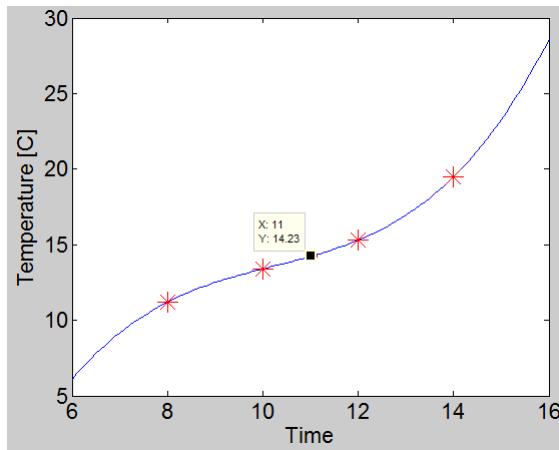
bzw. $P_n(11) = 14.225^{\circ}\text{C}$

- Falls man wirklich nur den einen Wert bei $x = 11$ sucht, wäre die Rechnung einiges einfacher:

$$\begin{aligned} l_0(11) &= \frac{(11-10)(11-12)(11-14)}{(-2)(-4)(-6)} = \frac{(1)(-1)(-3)}{(-2)(-4)(-6)} = -\frac{1}{16} \\ l_1(11) &= \frac{(11-8)(11-12)(11-14)}{(2)(-2)(-4)} = \frac{(3)(-1)(-3)}{(2)(-2)(-4)} = \frac{9}{16} \\ l_2(11) &= \frac{(11-8)(11-10)(11-14)}{(4)(2)(-2)} = \frac{(3)(1)(-3)}{(4)(2)(-2)} = \frac{9}{16} \\ l_3(11) &= \frac{(11-8)(11-10)(11-12)}{(6)(4)(2)} = \frac{(3)(1)(-1)}{(6)(4)(2)} = -\frac{1}{16} \end{aligned}$$

und damit $P_n(11) = 11.2 \cdot l_0(11) + 13.4 \cdot l_1(11) + 15.3 \cdot l_2(11) + 19.5 \cdot l_3(11) = 14.225$, wie zuvor.

- Die folgende Abbildung zeigt die vier Messpunkte (rot) und das Interpolationspolynom (blau) sowie den interpolierten Punkt (schwarz):



Aufgabe 6.1 [1]:

- Wir betrachten die Funktion $f(x) = \int_0^x e^{\sin t} dt$ aus Beispiel 6.1. Berechnen Sie den Wert $f(0.66) = ?$ anhand der drei Punkte

x	0.6	0.7	0.8
y	0.8136	0.9967	1.1944

Wie man schnell sieht, ist es für viele Aufgabenstellungen einfacher, nicht ein Interpolationspolynom für alle gegebenen Stützpunkte zu berechnen, sondern nur die Punkte in unmittelbarer Umgebung des gesuchten Wertes zu berücksichtigen. Im einfachsten Fall nimmt man jeweils die zwei benachbarten Stützpunkte und verbindet sie mit einer Geraden. Das Interpolationspolynom ist dann also erster Ordnung und man spricht deshalb auch von (*stückweise*) linearer Interpolation.

Wie gross ist nun aber der Fehler, den wir bei einer Interpolation machen? Dazu gilt der folgende Satz:

Satz 6.2: Fehlerabschätzung [2]

- Sind die y_i Funktionswerte einer genügend oft stetig differenzierbaren Funktion f (also $y_i = f(x_i)$), dann ist der Interpolationsfehler an einer Stelle x gegeben durch

$$|f(x) - P_n(x)| \leq \frac{|(x-x_0)(x-x_1) \cdots (x-x_n)|}{(n+1)!} \max_{x_0 \leq \xi \leq x_n} |f^{(n+1)}(\xi)|$$

Das heisst, wir müssen das Maximum der $(n+1)$ -ten Ableitung der Funktion $f(x)$ auf dem Intervall $[x_0, x_n]$ kennen. Die Fehlerabschätzung ist also nur anwendbar, wenn wir die Funktion $f(x)$ bzw. ihre Ableitungen selbst kennen. Deshalb ist diese Fehlerabschätzung theoretischer Natur.

Aufgabe 6.2 [1]:

- Von der Funktion $y = f(x) = 2^x$ seien die drei Stützpunkte

x	-1	1	3
y	0.5	2	8

gegeben. Wie gross ist der Interpolationsfehler im Punkt $x = 2$, wenn diese Punkte durch ein Polynom interpoliert werden? Man schätze den Fehler mittels der obigen Fehlerabschätzung und berechne anschliessend das (allgemeine) Polynom und den exakten Fehler.

6.2.3 Splineinterpolation

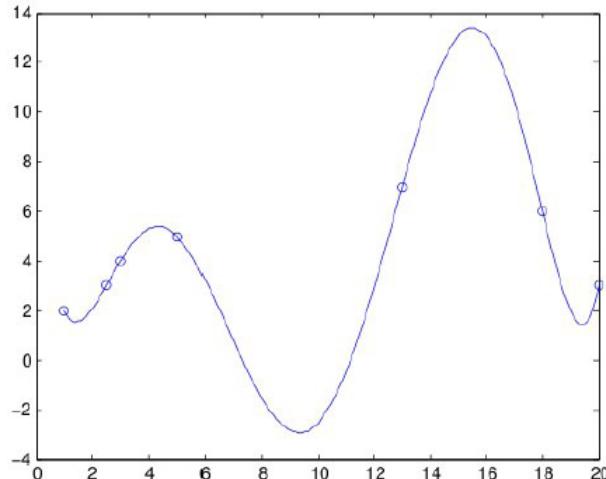
Polynome mit einem hohen Grad oszillieren. Dieses Verhalten führt dazu, dass bei einer grossen Anzahl Stützpunkten das Interpolationspolynom meist keine gute Näherung mehr für die zu interpolierende Funktion $f(x)$ darstellt, wie im folgenden Beispiel dargestellt.

Beispiel 6.4 [2]:

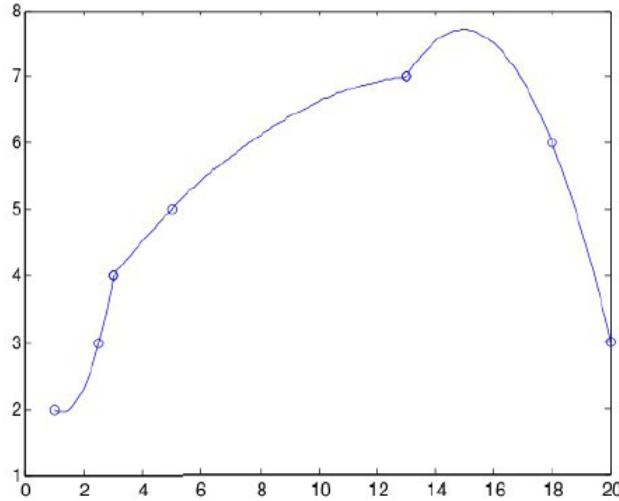
- Durch die Stützpunkte (in diesem Zusammenhang auch 'Knoten' genannt)

x	1	2.5	3	5	13	18	20
y	2	3	4	5	7	6	3

verläuft das folgenden Interpolationspolynom



- Wahrscheinlich ist dieses Polynom keine gute Näherung für die zugrundeliegende Funktion. Es könnte sich z.B. um physikalische Messwerte handeln, die immer positiv sein müssen. Dann ist eine Funktion, die zwischen-durch negativ wird, nicht brauchbar. Als Alternative könnte man versuchen, die Stützpunkte stückweise zu interpolieren, z.B. jeweils drei aufeinanderfolgende Punkte durch eine Parabel anzunähern. Man erhält dann das folgende Bild (aus [2]):



- Hier stören jetzt vor allem die Knicke an den Übergängen. Die Idee der Spline-Interpolation besteht nun darin, durch Polynome niederen Grades zu interpolieren und damit die Schwingungen zu unterdrücken, wobei man gleichzeitig sicherstellt, dass keine Knicke entstehen. Um dies zu erreichen, müssen die Polynome an den Anschlussstellen nicht nur denselben Funktionswert, sondern auch *dieselbe Ableitung* haben (d.h. die Steigung der Tangente in diesen Punkten stimmt überein).

Man kann z.B. für jedes Intervall

$$[x_i, x_{i+1}], \quad i = 0, 1, 2, \dots, n - 1$$

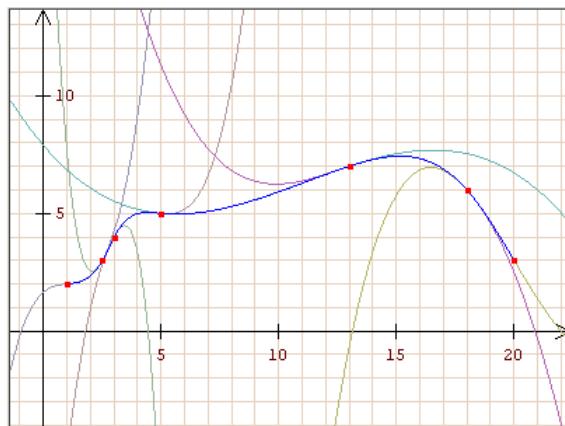
genau ein Polynom s_i ansetzen (der Index i bezeichnet hier nicht den Grad des Polynoms sondern die Nummer des Intervalls). Die Polynome s_i müssen dann folgende Bedingungen erfüllen:

$$\begin{aligned} s_i(x_i) &= y_i, \quad s_{i+1}(x_{i+1}) = y_{i+1}, \dots && \text{Interpolation} \\ s_i(x_{i+1}) &= s_{i+1}(x_{i+1}), \quad s_{i+1}(x_{i+2}) = s_{i+2}(x_{i+2}) \dots && \text{stetiger Übergang} \\ s'_i(x_{i+1}) &= s'_{i+1}(x_{i+1}), \quad s'_{i+1}(x_{i+2}) = s'_{i+2}(x_{i+2}), \dots && \text{keine Knicke} \end{aligned}$$

Zusätzlich sollen die Polynome in den Übergangspunkten die gleiche Krümmung aufweisen, d.h. auch die zweite Ableitung soll übereinstimmen:

$$s''_i(x_{i+1}) = s''_{i+1}(x_{i+1}), \quad s''_{i+1}(x_{i+2}) = s''_{i+2}(x_{i+2}), \dots \quad \text{gleiche Krümmung}$$

Polynome, die all diese Bedingungen erfüllen, müssen mindestens den Grad 3 haben (siehe folgendes Beispiel). Man spricht dann auch von *kubischen Splines*. Folgende Abbildung zeigt die Approximation der obigen Stützpunkte durch kubische Splines⁴:



⁴von <http://www.arndt-bruenner.de/mathe/scripts/kubspline.htm>

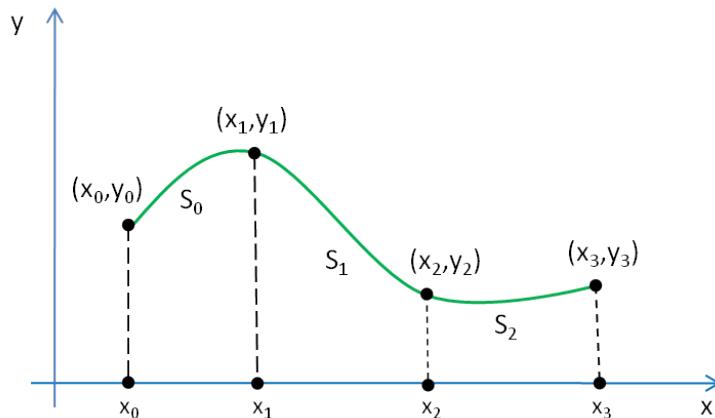
Die Bezeichnung 'Spline' kommt aus dem Englischen und bezeichnet eine schlanke, biegsame Holzlatte, die im Schiffsbau eingesetzt wurde, um durch eine Anzahl vorgegebener Punkte eine möglichst glatte Kurve zu legen (beispielsweise für die Planken des Schiffsrumpfes). Um den Reibungswiderstand im Wasser klein zu halten, ist es wichtig, dass keine Knicke oder abrupte Krümmungsänderungen auftreten. Wird eine solche Latte an vorgegebenen Punkten fixiert, nimmt sie die Form eines Polynomes, eben eines Splines, an.

Kubische Splines werden unter anderem zur Berechnung des Bahnverlaufes bei Achterbahnen verwendet, um ruckartige Beschleunigungswechsel für die Fahrgäste zu vermeiden. Des weiteren finden sie Anwendung bei der exakten Verlegung der Schienen bei Hochgeschwindigkeitsstrecken der Eisenbahn. Auch beim Entwurf von Kurven und Oberflächen (sogen. „Freiformkurven und -flächen“), wie sie häufig im Schiff-, Flugzeug- und Automobilbau vorkommen, sind Splines von Bedeutung. Die Eignung von Splines für solche Anwendungen liegt daran, dass für jeden Polynomabschnitt Randbedingungen erstens in Form von Punkten aber auch in Form von Werten für die erste und zweite Ableitung (und in Abhängigkeit davon Steigung und Krümmung/Kurvenradius) vorgeben lassen. Dadurch kann erreicht werden, dass die Krümmung entlang der Kurve immer stetig ist. Das hat den Vorteil, dass Querbeschleunigungen beim Abfahren der Kurve immer allmählich aufgebaut werden bzw. an den Knotenpunkten vorgegebene Werte einhalten⁵.

Betrachten wir dazu nun folgendes Beispiel:

Beispiel 6.5 [6]:

- Gegeben sind die vier Stützpunkte (x_i, y_i) für $i = 0, \dots, 3$. Wir suchen nun die Splinefunktion S , die durch diese vier Punkte geht und sich zusammensetzt aus den drei kubischen Polynomen S_0 , S_1 und S_2 .



- Mit dem Ansatz

$$\begin{aligned} S_0 &= a_0 + b_0(x - x_0) + c_0(x - x_0)^2 + d_0(x - x_0)^3, \quad x \in [x_0, x_1] \\ S_1 &= a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3, \quad x \in [x_1, x_2] \\ S_2 &= a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3, \quad x \in [x_2, x_3] \end{aligned}$$

müssen wir $3 \cdot 4 = 12$ Koeffizienten berechnen, wofür wir 12 Bedingungen benötigen. Diese lauten:

- Interpolation der Stützpunkte:

$$\begin{aligned} S_0(x_0) &= y_0 \\ S_1(x_1) &= y_1 \\ S_2(x_2) &= y_2 \\ S_2(x_3) &= y_3 \end{aligned}$$

- Stetiger Übergang an den Stellen x_1 und x_2 :

$$\begin{aligned} S_0(x_1) &= S_1(x_1) \\ S_1(x_2) &= S_2(x_2) \end{aligned}$$

⁵Aus Wikipedia.

3. Erste Ableitung an den Übergangsstellen muss übereinstimmen (keine Knicke):

$$\begin{aligned} S'_0(x_1) &= S'_1(x_1) \\ S'_1(x_2) &= S'_2(x_2) \end{aligned}$$

4. Zweite Ableitung an den Übergangsstellen muss übereinstimmen (gleiche Krümmung):

$$\begin{aligned} S''_0(x_1) &= S''_1(x_1) \\ S''_1(x_2) &= S''_2(x_2) \end{aligned}$$

Jetzt haben wir allerdings erst 10 Bedingungen für die 12 Koeffizienten. Das heisst, wir brauchen noch zwei zusätzliche. Diese können, in Abhängigkeit der Problemstellung, 'frei' gewählt werden und beziehen sich häufig auf die beiden Randstellen, hier x_0 und x_3 . Man unterscheidet zum Beispiel:

- die *natürliche kubische Splinefunktion*:

$$\begin{aligned} S''_0(x_0) &= 0 \\ S''_2(x_3) &= 0 \end{aligned}$$

mit einem möglichen Wendepunkt im Anfangs- und Endpunkt.

- die *periodische kubische Splinefunktion*

$$\begin{aligned} S'_0(x_0) &= S'_2(x_3) \\ S''_0(x_0) &= S''_2(x_3) \end{aligned}$$

wenn man die Periode $p = x_3 - x_0$ hat und damit $y_0 = y_3$ bzw. $S_0(x_0) = S_2(x_3)$ gilt.

- die kubische Splinefunktion mit *not-a-knot Bedingung*:

$$\begin{aligned} S'''_0(x_1) &= S'''_1(x_1) \\ S'''_1(x_2) &= S'''_2(x_2) \end{aligned}$$

so dass also auch die dritte Ableitung in x_1 und x_2 noch stetig ist (damit sind x_1 und x_2 keine 'echten' Knoten mehr, also 'not a knot').

Je nach Wahl dieser zusätzlichen zwei Randbedingungen ergibt sich ein leicht anderer Verlauf der kubischen Splinefunktion S .

Übersetzt man alle 12 Gleichungen in ein 12×12 Gleichungssystem, erhält man im Falle der natürlichen kubischen Splinefunktion:

$$\left(\begin{array}{cccccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & (x_1 - x_0) & 0 & 0 & (x_1 - x_0)^2 & 0 & 0 & (x_1 - x_0)^3 & 0 \\ 0 & 1 & 0 & 0 & (x_2 - x_1) & 0 & 0 & (x_2 - x_1)^2 & 0 & 0 & (x_2 - x_1)^3 \\ 0 & 0 & 1 & 0 & 0 & (x_3 - x_2) & 0 & 0 & (x_3 - x_2)^2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 2(x_1 - x_0) & 0 & 0 & 3(x_1 - x_0)^2 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 2(x_2 - x_1) & 0 & 0 & 3(x_2 - x_1)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 6(x_1 - x_0) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 6(x_2 - x_1) \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6(x_3 - x_2) & 0 \end{array} \right) \left(\begin{array}{c} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \\ c_0 \\ c_1 \\ c_2 \\ d_0 \\ d_1 \\ d_2 \end{array} \right) = \left(\begin{array}{c} y_0 \\ y_1 \\ y_2 \\ y_1 \\ y_2 \\ y_3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \right)$$

Durch eine Verallgemeinerung des Gleichungssystems für beliebig viele Stützpunkte erhält man den folgenden Algorithmus zur Berechnung der Koeffizienten für die natürliche kubische Splinefunktion:

Algorithmus: natürliche kubische Splinefunktion [6]

- Gegeben seien $n+1$ Stützpunkte (x_i, y_i) mit monoton aufsteigenden Stützstellen (Knoten) $x_0 < x_1 < \dots < x_n$ ($n \geq 2$). Gesucht ist die natürliche kubische Splinefunktion $S(x)$, welche in jedem Intervall $[x_i, x_{i+1}]$ mit $i = 0, 1, \dots, n-1$ durch ein kubisches Polynom

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

dargestellt wird, also $S(x) = S_i(x)$ mit $x \in [x_i, x_{i+1}]$.

- Die Koeffizienten a_i, b_i, c_i und d_i der Polynome $S_i(x)$ für $i = 0, 1, \dots, n-1$ berechnen sich wie folgt:

1. $a_i = y_i$
2. $h_i = x_{i+1} - x_i$
3. $c_0 = 0, c_n = 0$

4. Berechnung der Koeffizienten c_1, c_2, \dots, c_{n-1} aus dem Gleichungssystem

- (a) $i = 1 :$

$$2(h_0 + h_1)c_1 + h_1c_2 = 3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0}$$

- (b) falls $n \geq 4$ gilt für $i = 2, \dots, n-2 :$

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_ic_{i+1} = 3\frac{y_{i+1} - y_i}{h_i} - 3\frac{y_i - y_{i-1}}{h_{i-1}}$$

- (c) $i = n-1 :$

$$h_{n-2}c_{n-2} + 2(h_{n-2} + h_{n-1})c_{n-1} = 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}}$$

$$5. b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i)$$

$$6. d_i = \frac{1}{3h_i}(c_{i+1} - c_i)$$

Das Gleichungssystem unter Punkt 4 im obigen Algorithmus hat die Form

$$\mathbf{A}\mathbf{c} = \mathbf{z}$$

mit

$$\mathbf{A} = \begin{pmatrix} 2(h_0 + h_1) & h_1 & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & \\ & h_2 & 2(h_2 + h_3) & h_3 & \\ & & \ddots & \ddots & \\ & & & h_{n-3} & 2(h_{n-3} + h_{n-2}) \\ & & & & h_{n-2} \\ & & & & & 2(h_{n-2} + h_{n-1}) \end{pmatrix}$$

und

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} 3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0} \\ 3\frac{y_3 - y_2}{h_2} - 3\frac{y_2 - y_1}{h_1} \\ \vdots \\ 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}} \end{pmatrix}$$

Die Matrix A ist immer invertierbar. Zur numerischen Lösung sollte man den Gaußschen Algorithmus verwenden (siehe Kap. 4). Das System ist gut konditioniert, Pivotsuche ist nicht erforderlich. Natürlich bietet Python auch bereits fertige Funktionen für Splineinterpolation. Siehe dazu Aufgabe 6.4.

Aufgabe 6.3 [6]:

- Zu den folgenden Stützpunkten soll die natürliche kubische Splinefunktion bestimmt werden, d.h. bestimmen Sie manuell die Koeffizienten a_i, b_i, c_i, d_i der kubischen Polynome S_i für $i = 0, 1, 2$:

x_i	0	1	2	3
y_i	2	1	2	2

Aufgabe 6.4:

- Implementieren Sie den obigen Algorithmus in Python für eine beliebige vorgegebene Reihe von Stützpunkten. Ihre Funktion soll die Stützpunkte sowie die natürliche kubische Splinefunktion plotten.
- Testen Sie Ihren Algorithmus an der Zeitreihe der Bevölkerungszahl (in Mio.) der USA:

t	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990	2000
p(t)	75.995	91.972	105.711	123.203	131.669	150.697	179.323	203.212	226.505	249.633	281.422

- Benutzen Sie die Python-Funktionen
 - `scipy.interpolate.spline()`,

um diese Messreihe durch eine Splinefunktion zu interpolieren und vergleichen Sie das Resultat.

6.3 Ausgleichsrechnung

Nach dem Spezialfall der Interpolation wenden wir uns jetzt dem allgemeinen Problem zu, wie wir eine Menge von n Datenpunkten durch eine vorgegebene Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ annähern können.

6.3.1 Problemstellung

Im Unterschied zur Interpolation versuchen wir bei der Ausgleichsrechnung nicht, eine Funktion f zu finden, die exakt durch sämtliche n Datenpunkte geht, sondern diese nur möglichst gut approximiert. Dies ist insbesondere dann sinnvoll, wenn es eine grosse Anzahl von Datenpunkten gibt (die meist zusätzlich noch fehlerbehaftet sind) und diese durch eine Funktion mit nur wenigen Parametern beschrieben werden sollen. Es werden dabei zwei Fälle unterschieden:

- Bei der linearen Ausgleichsrechnung ist die gesuchte Funktion $f(x)$ eine Linearkombination von m sogn. Basisfunktionen $f_i(x)$ (mit $i = 1, 2, \dots, m$ und $m \leq n$), d.h. die gesuchten Parameter λ_i treten nur als multiplikative Faktoren bzw. als Koeffizienten der Linearkombination auf:

$$f(x) = \lambda_1 f_1(x) + \dots + \lambda_m f_m(x)$$

Beispiele:

- Das Polynom $f(x)$ vom Grad m setzt sich aus $m + 1$ Basisfunktionen $f_i(x) = x^{i-1}$ ($i = 1, \dots, m + 1$) zusammen: $f(x) = \lambda_1 \cdot \underbrace{1}_{f_1(x)} + \lambda_2 \cdot \underbrace{x}_{f_2(x)} + \lambda_3 \cdot \underbrace{x^2}_{f_3(x)} + \dots + \lambda_{m+1} \cdot \underbrace{x^m}_{f_{m+1}(x)}$
- Eine Linearkombination von beliebigen linearen und nicht-linearen Funktionen: $f(x) = \lambda_1 \cdot \underbrace{\sin(x)}_{f_1(x)} + \lambda_2 \cdot \underbrace{\cos(x)}_{f_2(x)} + \lambda_3 \cdot \underbrace{x}_{f_3(x)} + \lambda_4 \cdot \underbrace{\exp(x)}_{f_4(x)} + \lambda_5 \cdot \underbrace{1}_{f_5(x)} + \dots$

- Bei der nicht-linearen Ausgleichsrechnung lässt sich die gesuchte Funktion $f(x)$ nicht mehr als Linearkombination mit den gesuchten Parametern λ_i als Koeffizienten schreiben, diese treten “verwoben” mit der Funktionsgleichung auf:

$$f = f(\lambda_1, \lambda_2, \dots, \lambda_m, x)$$

Beispiele:

- Eine Gauss-Funktion: $f(x) = \lambda_1 e^{-\left(\frac{x-\lambda_2}{\lambda_3}\right)^2} + \lambda_4$
- Eine beliebige verkettete Funktion: $f(x) = \lambda_1 \cos(\lambda_2 x)$

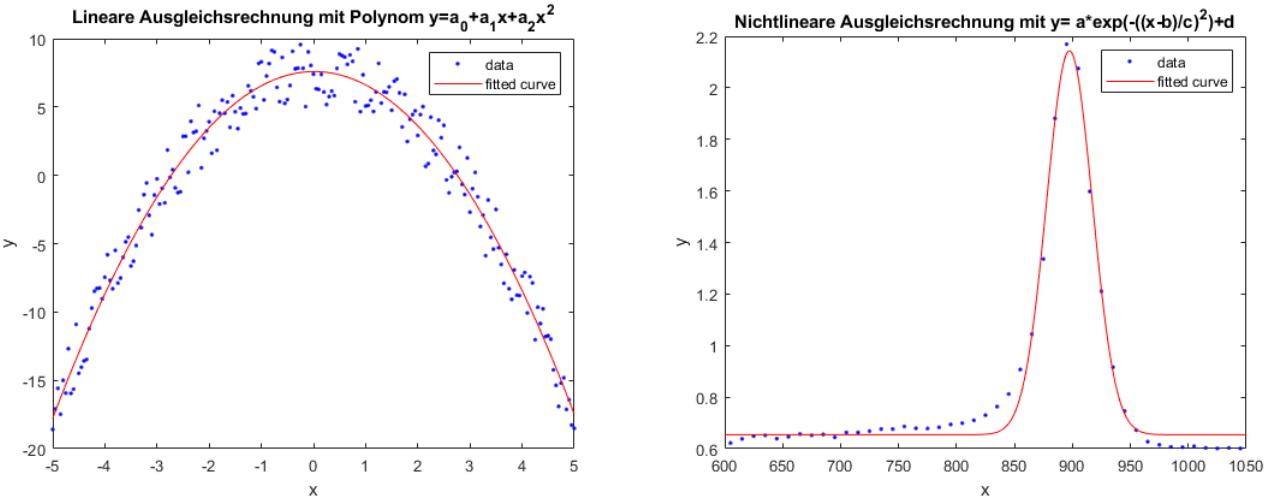


Abbildung 6.3: Nochmals wie Abb. 6.1. Links: Ein Beispiel der linearen Ausgleichsrechnung. Eine Datenwolke aus 200 Datenpunkten wird angenähert durch ein Polynom $f(x) = \lambda_1 + \lambda_2x + \lambda_3x^2$. Rechts: Ein Beispiel der nicht-linearen Ausgleichsrechnung. Die rund 50 Datenpunkte werden angenähert durch $f(x) = \lambda_1 e^{-\left(\frac{x-\lambda_2}{\lambda_3}\right)^2} + \lambda_4$. Natürlich können die Parameter beliebig benannt werden (z.B. a statt λ_1 etc.).

Definition 6.2: Ausgleichsproblem [1]

- Gegeben sind n Wertepaare (x_i, y_i) , $i = 1, \dots, n$ mit $x_i \neq x_j$ für $i \neq j$.
- Gesucht ist eine stetige Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$, die die Wertepaare in einem gewissen Sinn bestmöglich annähert, d.h. dass möglichst genau gilt

$$f(x_i) \approx y_i$$

für alle $i = 1, \dots, n$.

Definition 6.3: Ansatzfunktionen / Ausgleichsfunktion / Fehlerfunktional / kleinste Fehlerquadrate [1]

- Gegeben sei eine Menge F von stetigen **Ansatzfunktionen** f auf dem Intervall $[a, b]$ sowie n Wertepaare (x_i, y_i) , $i = 1, \dots, n$.
- Ein Element $f \in F$ heisst **Ausgleichsfunktion** von F zu den gegebenen Wertepaaren, falls das **Fehlerfunktional**

$$E(f) := \| \mathbf{y} - f(\mathbf{x}) \|_2^2 = \sum_{i=1}^n (y_i - f(x_i))^2$$

für f minimal wird, d.h. $E(f) = \min\{E(g) \mid g \in F\}$.

- Man nennt das so gefundene f dann optimal im Sinne der **kleinsten Fehlerquadrate** (least squares fit).

Bemerkungen [1]:

- Diese Forderung der kleinsten Fehlerquadrate bedeutet nichts anderes, als dass das Quadrat der 2-Norm (vgl. Kap. 4.6.1) bzw. das Quadrat des Betrags des Fehlervektors

$$\begin{pmatrix} y_1 - f(x_1) \\ \vdots \\ y_n - f(x_n) \end{pmatrix}$$

minimal sein soll. Andere Normen sind auch möglich, was zu anderen Ergebnissen führen kann. Die in der Praxis am häufigsten verwendete Norm ist allerdings die 2-Norm.

- Es ist möglich, die Fehler noch mit Gewichten $w_i > 0$ zu versehen, d.h. man minimiert

$$\sum_{i=1}^n w_i \cdot (y_i - f(x_i))^2.$$

Wenn z.B. gewisse Datenpunkte (x_i, y_i) grosse Messungenauigkeiten aufweisen, können diese mit einem kleineren Gewicht versehen werden, als Datenpunkte mit höherer Messgenauigkeit. Die Summe der Gewichte sollte dabei normiert sein, also $\sum_{i=1}^n w_i = 1$.

- Wählt man F als Menge aller Geraden, dann nennt man das so gefundene *f Ausgleichsgerade* oder in statistischen Zusammenhängen auch *Regressionsgerade*.

6.3.2 Lineare Ausgleichsprobleme

Beginnen wir mit dem einfachen Beispiel der linearen Regression. Ausgehend von der Wertetabelle

x_i	x_1	x_2	\dots	x_n
y_i	y_1	y_2	\dots	y_n

suchen wir die Ausgleichsgerade der Form $f(x) = ax + b$, also $F := \{af_1 + bf_2 \mid a, b \in \mathbb{R}\}$ mit den Basisfunktionen $f_1(x) = x$ und $f_2(x) = 1$.

Das Fehlerfunktional hat dann die Form

$$E(f)(a, b) := E(f) = \sum_{i=1}^n (y_i - f(x_i))^2 = \sum_{i=1}^n (y_i - (ax_i + b))^2.$$

Dieses soll minimal werden, d.h. die partiellen Ableitungen nach den Parametern a und b müssen verschwinden (in Analogie zum eindimensionalen Fall):

$$\begin{aligned} 0 &\stackrel{!}{=} \frac{\partial E}{\partial a} = \sum_{i=1}^n \frac{\partial}{\partial a} (y_i - (ax_i + b))^2 = \sum_{i=1}^n 2 \cdot (y_i - (ax_i + b)) \cdot (-x_i) = -2 \sum_{i=1}^n (y_i - (ax_i + b)) \cdot (x_i) \\ 0 &\stackrel{!}{=} \frac{\partial E}{\partial b} = \sum_{i=1}^n \frac{\partial}{\partial b} (y_i - (ax_i + b))^2 = \sum_{i=1}^n 2 \cdot (y_i - (ax_i + b)) \cdot (-1) = -2 \sum_{i=1}^n (y_i - (ax_i + b)) \end{aligned}$$

Wir haben also zwei Gleichungen für die Unbekannten a, b . Durch Umformung erhalten wir

$$\begin{aligned} 0 &\stackrel{!}{=} \sum_{i=1}^n (y_i - (ax_i + b)) \cdot (x_i) = \sum_{i=1}^n (x_i y_i - ax_i^2 - bx_i) = \sum_{i=1}^n x_i y_i - \sum_{i=1}^n a x_i^2 - \sum_{i=1}^n b x_i = \sum_{i=1}^n x_i y_i - a \sum_{i=1}^n x_i^2 - b \sum_{i=1}^n x_i \\ 0 &\stackrel{!}{=} \sum_{i=1}^n (y_i - (ax_i + b)) = \sum_{i=1}^n y_i - \sum_{i=1}^n a x_i - \sum_{i=1}^n b = \sum_{i=1}^n y_i - \sum_{i=1}^n a x_i - \sum_{i=1}^n b = \sum_{i=1}^n y_i - a \sum_{i=1}^n x_i - b \sum_{i=1}^n 1 \end{aligned}$$

Daraus folgt

$$\begin{aligned} a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i &= \sum_{i=1}^n x_i y_i \\ a \sum_{i=1}^n x_i + b \underbrace{\sum_{i=1}^n 1}_{=n} &= \sum_{i=1}^n y_i \end{aligned}$$

oder in Matrixschreibweise erhalten wir das lineare Gleichungssystem für die Unbekannten (a, b)

$$\left(\begin{array}{cc} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & n \end{array} \right) \left(\begin{array}{c} a \\ b \end{array} \right) = \left(\begin{array}{c} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{array} \right)$$

und können dieses nach a, b auflösen.

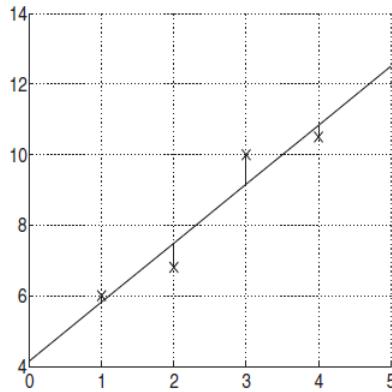


Abbildung 6.4: Ausgleichsgerade $f(x) = 1.67x + 4.15$ zu Bsp. 6.6 (aus [1]).

Beispiel 6.6 [1]:

- Wir bestimmen die Ausgleichsgerade $f(x) = ax + b$ für die folgende Wertetabelle:

x_i	1	2	3	4
y_i	6	6.8	10	10.5

- Lösung: mit $n = 4$ und

$$\sum_{i=1}^4 x_i^2 = 30, \quad \sum_{i=1}^4 x_i = 10, \quad \sum_{i=1}^4 x_i y_i = 91.6, \quad \sum_{i=1}^4 y_i = 33.3$$

erhalten wir das lineare Gleichungssystem

$$\begin{pmatrix} 30 & 10 \\ 10 & 4 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 91.6 \\ 33.3 \end{pmatrix}$$

mit der Lösung

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1.67 \\ 4.15 \end{pmatrix}.$$

Also ist die gesuchte Ausgleichsgerade $f(x) = 1.67x + 4.15$ (siehe Abb. 6.4).

Allgemein definieren wir das lineare Ausgleichproblem wie folgt:

Definition 6.4: Lineares Ausgleichproblem [1]

- Gegeben seien n Wertepaare (x_i, y_i) , $i = 1, \dots, n$, und m Basisfunktionen f_1, \dots, f_m auf einem Intervall $[a, b]$. Wir wählen F als die Menge der Ansatzfunktionen $f := \lambda_1 f_1 + \dots + \lambda_m f_m$ mit $\lambda_j \in \mathbb{R}$, also $F = \{f = \lambda_1 f_1 + \dots + \lambda_m f_m \mid \lambda_j \in \mathbb{R}, j = 1, \dots, m\}$.
- Es liegt dann ein **lineares Ausgleichproblem** vor mit dem Fehlerfunktional

$$\begin{aligned} E(f) &= \| \mathbf{y} - f(\mathbf{x}) \|_2^2 \\ &= \sum_{i=1}^n (y_i - f(x_i))^2 \\ &= \sum_{i=1}^n \left(y_i - \sum_{j=1}^m \lambda_j f_j(x_i) \right)^2 \\ &= \| \mathbf{y} - \mathbf{A}\boldsymbol{\lambda} \|_2^2 \end{aligned}$$

vor, wobei

$$\mathbf{A} = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix}$$

- Das System $\mathbf{A}\boldsymbol{\lambda} = \mathbf{y}$ heisst **Fehlergleichungssystem**.

Bemerkungen [1]:

- Das Fehlergleichungssystem besitzt n Gleichungen und m Unbekannte. In der Regel ist $n > m$, wir haben also mehr Gleichungen als Unbekannte, d.h. das Gleichungssystem ist überbestimmt. Man kann daher nicht damit rechnen, dass es eine Lösung gibt (bzw. dass das Fehlerfunktional $E(f) = 0$ erreicht werden kann).
- Für $m = n$ haben wir eine eindeutige Lösung und $E(f) = 0$. Dies ist gleichbedeutend damit, dass die Ausgleichsfunktion f exakt durch sämtliche Wertepaare geht. Dies ist der Spezialfall der Interpolation.

Wir können nun versuchen, das Fehlerfunktional zu minimieren. Dafür müssen sämtliche partiellen Ableitungen von $E(f)$ nach den unbekannten Koeffizienten λ_j verschwinden, also

$$\frac{\partial E(f)}{\partial \lambda_j} = 0.$$

Dies führt uns zu den Begriffen der Normalgleichungen und des Normalgleichungssystems.

Definition 6.5: Normalgleichungen / Normalgleichungssystem [1]

- Die Gleichungen

$$\frac{\partial E(f)(\lambda_1, \dots, \lambda_m)}{\partial \lambda_j} = 0, \quad j = 1, \dots, m$$

heissen **Normalgleichungen** des linearen Ausgleichproblems.

- Das System sämtlicher Normalgleichungen heisst **Normalgleichungssystem** und lässt sich als lineares Gleichungssystem schreiben

$$\mathbf{A}^T \mathbf{A} \boldsymbol{\lambda} = \mathbf{A}^T \mathbf{y}$$

- Die Lösung $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)^T$ des Normalgleichungssystems beinhaltet die gesuchten Parameter des linearen Ausgleichproblems.

Bemerkung:

- Das lineare Normalgleichungssystem lässt sich mit den Verfahren aus Kap. 4 lösen, im einfachsten Fall mit dem Gauss-Algorithmus bzw. der LR-Zerlegung. Die $m \times m$ Matrix $\mathbf{A}^T \mathbf{A}$ ist allerdings i.d.R. schlecht konditioniert, d.h. kleine Fehler in den Datenpunkten \mathbf{y} können grosse Auswirkungen auf die Lösung $\boldsymbol{\lambda}$ haben.

- Das Normalgleichungssystem lässt sich stabiler mithilfe der QR -Zerlegung der Matrix \mathbf{A} lösen (siehe Kap. 4.5.2). Zur Erinnerung: es gilt $\mathbf{A} = \mathbf{Q}\mathbf{R}$ mit einer Orthogonalmatrix \mathbf{Q} , wobei $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_n$ (\mathbf{I}_n ist die $n \times n$ Einheitsmatrix) resp. $\mathbf{Q}^{-1} = \mathbf{Q}^T$, \mathbf{R} ist eine rechtsobere Dreiecksmatrix.

Aus

$$\mathbf{A} = \mathbf{Q}\mathbf{R}$$

folgt für das Normalgleichungssystem

$$\mathbf{A}^T \mathbf{A} \boldsymbol{\lambda} = \mathbf{A}^T \mathbf{y}$$

das äquivalente, häufig aber besser konditionierte Gleichungssystem

$$\mathbf{R}\boldsymbol{\lambda} = \mathbf{Q}^T \mathbf{y}$$

da

$$\begin{aligned} \mathbf{A}^T \mathbf{A} \boldsymbol{\lambda} &= \mathbf{A}^T \mathbf{y} \\ (\mathbf{Q}\mathbf{R})^T(\mathbf{Q}\mathbf{R})\boldsymbol{\lambda} &= (\mathbf{Q}\mathbf{R})^T \mathbf{y} \\ \mathbf{R}^T \underbrace{\mathbf{Q}^T \mathbf{Q}}_{\mathbf{I}_n} \mathbf{R} \boldsymbol{\lambda} &= \mathbf{R}^T \mathbf{Q}^T \mathbf{y} \\ \mathbf{R}^T \mathbf{R} \boldsymbol{\lambda} &= \mathbf{R}^T \mathbf{Q}^T \mathbf{y} \\ \Rightarrow \mathbf{R} \boldsymbol{\lambda} &= \mathbf{Q}^T \mathbf{y} \end{aligned}$$

Auf der linken Seite des äquivalenten Gleichungssystems steht nun die rechtsobere Dreiecksmatrix \mathbf{R} , welche häufig besser konditioniert ist als $\mathbf{A}^T \mathbf{A}$. Das Gleichungssystem lässt sich dann einfach durch Rückwärtseinsetzen lösen. Der Vorteil der numerischen Stabilität bedingt den erhöhten Aufwand durch die QR -Zerlegung.

- In der Umsetzung berechnen wir die QR -Zerlegung der Matrix \mathbf{A} mit Python mit `numpy.linalg.qr()` und lösen das lineare Gleichungssystem $\mathbf{R}\boldsymbol{\lambda} = \mathbf{Q}^T \mathbf{y}$ mit `numpy.linalg.solve()`.

Beispiel 6.7 [1]:

- Stellen Sie für das lineare Ausgleichproblem in Bsp. 6.6 das Normalgleichungssystem auf.
- Lösung: mit $f_1(x) = x$ und $f_2(x) = 1$ erhalten wir

$$\mathbf{A} = \begin{pmatrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ f_1(x_3) & f_2(x_3) \\ f_1(x_4) & f_2(x_4) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{pmatrix}$$

$$\begin{aligned} \Rightarrow \mathbf{A}^T \mathbf{A} &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{pmatrix} = \begin{pmatrix} 30 & 10 \\ 10 & 4 \end{pmatrix} \\ \mathbf{A}^T \mathbf{y} &= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 6.8 \\ 10 \\ 10.5 \end{pmatrix} = \begin{pmatrix} 91.6 \\ 33.3 \end{pmatrix} \end{aligned}$$

Damit entspricht $\mathbf{A}^T \mathbf{A} \boldsymbol{\lambda} = \mathbf{A}^T \mathbf{y}$ dem Gleichungssystem aus Bsp. 6.6 und hat natürlich die gleiche Lösung $\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1.67 \\ 4.15 \end{pmatrix}$.

Aufgabe 6.5:

- Schreiben Sie ein Python-Skript, welches Beispiel 6.7 mittels der QR -Zerlegung löst und die Daten sowie die Ausgleichsgerade plottet.

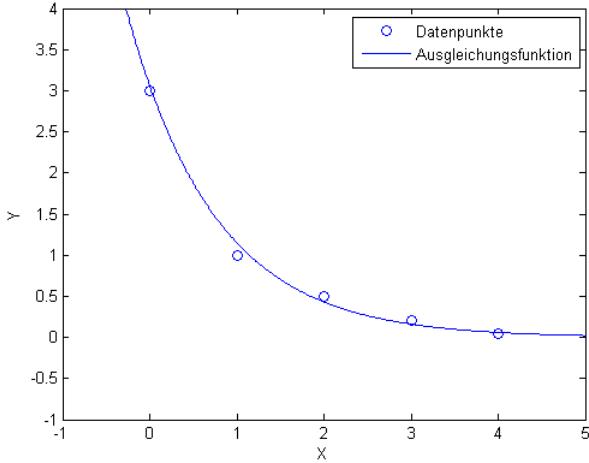


Abbildung 6.5: Ausgleichsfunktion $f(x) = 3.0638 \cdot e^{-0.9798x}$ zu Bsp. 6.8.

Beispiel 6.8 [1]:

- Bestimmen Sie mit den bisher behandelten Techniken eine Funktion der Form $f(x) = ae^{bx}$, die die folgenden Daten möglichst gut approximiert:

x_i	0	1	2	3	4
y_i	3	1	0.5	0.2	0.05

- Lösung: auf den ersten Blick liegt hier kein lineares Ausgleichproblem vor, da wir $f(x)$ nicht als Summe $af_1(x) + bf_2(x)$ schreiben können. Wir können aber durch beidseitiges Logarithmieren einen linearen Ansatz erreichen:

$$\ln f(x) = \ln(ae^{bx}) = \ln(a) + b \cdot \ln(e^x) = \ln(a) + b \cdot \underbrace{1}_{f_1(x)} + b \cdot \underbrace{x}_{f_2(x)}$$

Wir werden durch die Lösung des Normalgleichungssystems also $\ln(a)$ und b berechnen können, wenn wir die y_i logarithmieren:

$$\begin{aligned} \mathbf{A}^T \mathbf{A} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 10 & 30 \end{pmatrix} \\ \mathbf{A}^T \cdot \tilde{\mathbf{y}} &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix} \begin{pmatrix} \ln 3 \\ \ln 1 \\ \ln 0.5 \\ \ln 0.2 \\ \ln 0.05 \end{pmatrix} = \begin{pmatrix} -4.1997 \\ -18.1975 \end{pmatrix} \\ \mathbf{A}^T \mathbf{A} \boldsymbol{\lambda} &= \mathbf{A}^T \tilde{\mathbf{y}} \Rightarrow \boldsymbol{\lambda} = \begin{pmatrix} \ln a \\ b \end{pmatrix} = \begin{pmatrix} 1.11968... \\ -0.97981... \end{pmatrix} \Rightarrow f(x) = e^{\ln a} \cdot e^{bx} = 3.06388... \cdot e^{-0.97981...x}. \end{aligned}$$

Wir werden dieses Beispiel im nächsten Abschnitt bei nichtlinearen Ausgleichsproblemen gleich nochmals aufgreifen.

Aufgabe 6.6 [1]:

- Bestimmen Sie für die folgenden Daten die Funktion der Form $f(x) = ae^x + b$, die diese Daten bestmöglich im Sinne der kleinsten Fehlerquadrate approximiert:

x_i	0	1	2	3	4
y_i	6	12	30	80	140

6.3.3 Nichtlineare Ausgleichsprobleme

Wir haben in Beispiel 6.8 gesehen, dass man allenfalls in Spezialfällen ein nichtlineares Ausgleichsproblem in ein lineares Ausgleichsproblem umformen kann. Trotzdem wollen wir natürlich in der Lage sein, allgemeine nichtlineare Ausgleichsprobleme zu lösen.

Definition 6.6: Allgemeines Ausgleichsproblem [1]

- Gegeben seien n Wertepaare $(x_i, y_i), i = 1, \dots, n$, und die Menge F der Ansatzfunktionen $f_p = f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x)$ mit m Parametern $\lambda_j \in \mathbb{R}$, $j = 1, \dots, m$, also $F = \{f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x) \mid \lambda_j \in \mathbb{R}, j = 1, \dots, m\}$.
- Das **allgemeine Ausgleichsproblem** besteht darin, die m Parameter $\lambda_1, \dots, \lambda_m$ zu bestimmen, so dass das Fehlerfunktional E

$$\begin{aligned} E(f) &= \sum_{i=1}^n (y_i - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_i))^2 \\ &= \left\| \begin{pmatrix} y_1 - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_1) \\ y_2 - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_2) \\ \vdots \\ y_n - f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_n) \end{pmatrix} \right\|_2^2 \\ &= \| \mathbf{y} - \mathbf{f}(\boldsymbol{\lambda}) \|_2^2 \end{aligned}$$

minimal wird unter allen zulässigen Parameterbelegungen, wobei

$$\mathbf{f}(\boldsymbol{\lambda}) := \mathbf{f}(\lambda_1, \lambda_2, \dots, \lambda_m) := \begin{pmatrix} f_1(\lambda_1, \lambda_2, \dots, \lambda_m) \\ f_2(\lambda_1, \lambda_2, \dots, \lambda_m) \\ \vdots \\ f_n(\lambda_1, \lambda_2, \dots, \lambda_m) \end{pmatrix} := \begin{pmatrix} f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_1) \\ f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_2) \\ \vdots \\ f_p(\lambda_1, \lambda_2, \dots, \lambda_m, x_n) \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix}$$

Bemerkungen [1]:

- Falls die Ansatzfunktionen f_p linear in den Parametern sind, haben wir den Spezialfall des linearen Ausgleichsproblems mit $\mathbf{f}(\boldsymbol{\lambda}) = \mathbf{A}\boldsymbol{\lambda}$ (vgl. Def. 6.4).
- Das allg. Ausgleichsproblem ist also äquivalent zur Bestimmung des Minimums einer Funktion

$$E : \mathbb{R}^m \longrightarrow \mathbb{R}$$

und wir könnten wieder die Normalgleichungen aufstellen, indem wir die partiellen Ableitungen von f nach den Parametern λ_i gleich Null setzen und das entstehende, diesmal nicht lineare Gleichungssystem lösen. Wir werden im folgenden Beispiel dies einmal ansatzweise machen und werden dann aber mit Gauss-Newton-Verfahren im nächsten Abschnitt ein anderes, effektiveres Verfahren kennenlernen.

Beispiel 6.9 [1]:

- Wir nehmen wieder das Problem aus Beispiel 6.8, d.h. wir wollen eine Ansatzfunktion $f(x) = ae^{bx}$ bestmöglich im Sinn der kleinsten Fehlerquadrate an die gegebenen Daten anpassen:

x_i	0	1	2	3	4
y_i	3	1	0.5	0.2	0.05

- Lösung: Wir haben also zwei Parameter a und b in der Ansatzfunktion $f_p(a, b, x) = ae^{bx}$ so zu bestimmen, dass

$$E(f) = \sum_{i=1}^5 (y_i - f_p(a, b, x_i))^2 = \sum_{i=1}^5 (y_i - ae^{bx_i})^2$$

minimal wird. Nullsetzen der partiellen Ableitungen liefert

$$\begin{aligned} 0 &= \frac{\partial E(f)}{\partial a} = \sum_{i=1}^5 \frac{\partial}{\partial a} (y_i - ae^{bx_i})^2 = \sum_{i=1}^5 2 \cdot (y_i - ae^{bx_i}) \cdot (-e^{bx_i}) = -2 \sum_{i=1}^5 (y_i - ae^{bx_i}) \cdot e^{bx_i} \\ 0 &= \frac{\partial E(f)}{\partial b} = \sum_{i=1}^5 \frac{\partial}{\partial b} (y_i - ae^{bx_i})^2 = \sum_{i=1}^5 2 \cdot (y_i - ae^{bx_i}) \cdot (-ae^{bx_i}) \cdot x_i = -2 \sum_{i=1}^5 (y_i - ae^{bx_i}) \cdot ae^{bx_i} \cdot x_i \end{aligned}$$

Dies sind zwei nichtlineare Gleichungen, die mit dem Newton-Verfahren für Systeme (vgl. Kap. 5) gelöst werden können. Wegen der Länge der Ausdrücke verzichten wir hier an dieser Stelle auf die Details, ein funktionierendes Python Skript ist unten angegeben. Für die Startwerte $a_0 = 3$ und $b_0 = -1$ und einer Fehlerschranke von 10^{-5} liefert es nach zweimaliger Iteration

$$a = 2.98165..., b = -1.00328...$$

Zum Vergleich: in Bsp. 6.8 hatten wir durch logarithmieren das nichtlineare Ausgleichsproblem auf ein lineares zurückgeführt und für die Parameter

$$a = 3.06388..., b = -0.97981...$$

erhalten. Der Unterschied wird nicht durch Rechenengenauigkeiten verursacht, tatsächlich handelt es sich um zwei verschiedene Modellanpassungen.

- Python-Skript

```
import numpy as np
import sympy as sp
a, b = sp.symbols('a b')
x = np.array([0, 1, 2, 3, 4])
y = np.array([3, 1, 0.5, 0.2, 0.05])

f1 = 0;
for i in range(0, 5):
    f1 = f1 + (y[i] - a * sp.exp(b * x[i])) * sp.exp(b * x[i])
f1 = -2 * f1

f2 = 0;
for i in range(0, 5):
    f2 = f2 + (y[i] - a * sp.exp(b * x[i])) * a * sp.exp(b * x[i]) * x[i]
f2 = -2 * f2

f = sp.Matrix([f1, f2])
lam = sp.Matrix([a, b]) # Achtung: die Unbekannten sind a und b (nicht x)
Df = f.jacobian(lam)

f = sp.lambdify(([a], [b]), f, "numpy")
Df = sp.lambdify(([a], [b]), Df, "numpy")
```

- Fortsetzung:

```
# Newtons Methode für Systeme
def newton(f, Df, x0, tol):
    n=0
    x=np.copy(x0)
    err = np.linalg.norm(f(x),2)
    while err > tol:
        delta = np.linalg.solve(Df(x), -f(x))
        x = x+delta
        err = np.linalg.norm(f(x),2)
        n = n+1
    return(x,n)

# Aufruf
tol = 1e-5
x0 = np.array([3, -1]).T
[xn,n] = newton(f, Df, x0, tol)
print('x_{} + str(n) + ' = ' + str(np.reshape(xn,(2))))
```

Das obige Verfahren führt zwar zu einem Ergebnis, hat jedoch mehrere Nachteile. Das Fehlerfunktional und seine partiellen Ableitungen müssen explizit berechnet werden, und es konvergiert nur, wenn man genügend gute Anfangsnäherungen für die Parameter zur Verfügung hat, was häufig nicht der Fall ist. So würden wir mit einem Startwert $x_0 = [1, -1]$ im obigen Skript das Resultat $a = 3.00000\dots$, $b = -13.60235\dots$ erhalten. Das folgende iterative Verfahren hat diesbezüglich einige Vorteile.

6.3.4 Das Gauss-Newton-Verfahren

Wir definieren:

Definition 6.7: Quadratmittelproblem [1]

- Gegeben ist eine Funktion $\mathbf{g} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ und das zugehörige Fehlerfunktional $E : \mathbb{R}^m \rightarrow \mathbb{R}$, definiert durch $E(\mathbf{x}) := \|\mathbf{g}(\mathbf{x})\|_2^2$. Das Problem, einen Vektor $\mathbf{x} \in \mathbb{R}^m$ zu finden, für den $E(\mathbf{x})$ minimal wird, nennt man Quadratmittelproblem.

Nichtlineare Ausgleichsprobleme sind also Quadratmittelprobleme, wobei wir $\mathbf{g}(\mathbf{x})$ ersetzen mit $\mathbf{g}(\boldsymbol{\lambda})$:

$$\mathbf{g}(\boldsymbol{\lambda}) := \mathbf{y} - \mathbf{f}(\boldsymbol{\lambda}), \quad \mathbf{y} \in \mathbb{R}^n, \quad \boldsymbol{\lambda} \in \mathbb{R}^m$$

wie in Def. 6.6. Das Fehlerfunktional lautet entsprechend

$$E(\boldsymbol{\lambda}) := \|\mathbf{g}(\boldsymbol{\lambda})\|_2^2 = \|\mathbf{y} - \mathbf{f}(\boldsymbol{\lambda})\|_2^2$$

und der Vektor $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)^T$ beinhaltet die gesuchten Parameter.

Das folgende Gauss-Newton-Verfahren besteht aus einer Kombination von linearer Ausgleichrechnung und dem Newton-Verfahren. Wir ersetzen dabei in $\|\mathbf{g}(\boldsymbol{\lambda})\|_2^2$ den Vektor $\mathbf{g}(\boldsymbol{\lambda})$ durch die Linearkombination

$$\mathbf{g}(\boldsymbol{\lambda}) \approx \mathbf{g}(\boldsymbol{\lambda}^{(0)}) + \mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)}) \cdot (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(0)})$$

Dies entspricht der 'verallgemeinerten Tangentengleichung' aus Def. 5.3 mit der $n \times m$ Jacobi-Matrix $\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)})$ in der Umgebung eines Startwertes $\boldsymbol{\lambda}^{(0)}$:

$$\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)}) := \begin{pmatrix} \frac{\partial g_1}{\partial \lambda_1}(\boldsymbol{\lambda}^{(0)}) & \frac{\partial g_1}{\partial \lambda_2}(\boldsymbol{\lambda}^{(0)}) & \cdots & \frac{\partial g_1}{\partial \lambda_m}(\boldsymbol{\lambda}^{(0)}) \\ \frac{\partial g_2}{\partial \lambda_1}(\boldsymbol{\lambda}^{(0)}) & \frac{\partial g_2}{\partial \lambda_2}(\boldsymbol{\lambda}^{(0)}) & \cdots & \frac{\partial g_2}{\partial \lambda_m}(\boldsymbol{\lambda}^{(0)}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial \lambda_1}(\boldsymbol{\lambda}^{(0)}) & \frac{\partial g_n}{\partial \lambda_2}(\boldsymbol{\lambda}^{(0)}) & \cdots & \frac{\partial g_n}{\partial \lambda_m}(\boldsymbol{\lambda}^{(0)}) \end{pmatrix}$$

Das heisst, wir suchen statt des Minimums des ursprünglichen Fehlerfunktionalen

$$E(\boldsymbol{\lambda}) = \| \mathbf{g}(\boldsymbol{\lambda}) \|_2^2$$

nun das Minimum des 'linearisierten Fehlerfunktionalen'

$$\tilde{E}(\boldsymbol{\lambda}) = \| \underbrace{\mathbf{g}(\boldsymbol{\lambda}^{(0)})}_{\tilde{\mathbf{y}}} + \underbrace{\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)})}_{-\tilde{\mathbf{A}}} \cdot \underbrace{(\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(0)})}_{\boldsymbol{\delta}} \|_2^2$$

und haben so das nichtlineare Ausgleichproblem zurückgeführt auf ein lineares Ausgleichproblem. Mit den Substitutionen

$$\begin{aligned}\tilde{\mathbf{y}} &:= \mathbf{g}(\boldsymbol{\lambda}^{(0)}) \\ \tilde{\mathbf{A}} &:= -\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)}) \\ \boldsymbol{\delta} &:= \boldsymbol{\lambda} - \boldsymbol{\lambda}^{(0)}\end{aligned}$$

haben wir die gleiche Form wie in Def. 6.4 beim linearen Ausgleichsproblem

$$\tilde{E}(\boldsymbol{\lambda}) = \| \tilde{\mathbf{y}} - \tilde{\mathbf{A}}\boldsymbol{\delta} \|_2^2$$

und die Lösung berechnet sich gemäss Def. 6.5 über das Normalgleichungssystem

$$\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \boldsymbol{\delta} = \tilde{\mathbf{A}}^T \tilde{\mathbf{y}}$$

bzw.

$$\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)})^T \mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)}) \boldsymbol{\delta} = -\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)})^T \cdot \mathbf{g}(\boldsymbol{\lambda}^{(0)}).$$

Wegen der besseren Konditionierung lösen wir das Normalgleichungssystem nicht mit dem Gauss-Algorithmus sondern benutzen wieder die QR -Zerlegung, diesmal von $\mathbf{D}\mathbf{g}(\boldsymbol{\lambda}^{(0)}) = \mathbf{Q}\mathbf{R}$, und lösen das äquivalente Gleichungssystem

$$\mathbf{R}\boldsymbol{\delta} = -\mathbf{Q}^T \mathbf{g}(\boldsymbol{\lambda}^{(0)}).$$

Die Lösung dieses linearen Gleichungssystems liefert uns dann $\boldsymbol{\delta} = \boldsymbol{\lambda} - \boldsymbol{\lambda}^{(0)}$ und damit können wir nach der eigentlich gesuchten Grösse

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(0)} + \boldsymbol{\delta}$$

auflösen. Die Grösse $\boldsymbol{\delta}$ liefert uns in diesem Sinne eine Korrektur bzw. "Verbesserung" zum Startwert $\boldsymbol{\lambda}^{(0)}$. Unter der Annahme, dass $\boldsymbol{\lambda}$ eine bessere Näherung ist als der Startwert $\boldsymbol{\lambda}^{(0)}$, können wir $\boldsymbol{\lambda}^{(1)} := \boldsymbol{\lambda}$ setzen und diesen Schritt durch rekursives Einsetzen wiederholen. Damit erhalten wir das Gauss-Newton Verfahren:

Definition 6.8: Gauss-Newton-Verfahren

- Sei $\lambda^{(0)}$ ein Startvektor in der Nähe des Minimums des Fehlerfunktionalen E der Ausgleichsfunktion $f(\lambda)$. Das Gauss-Newton-Verfahren zur näherungsweisen Bestimmung des Minimums lautet:

- Berechne die Funktion

$$g(\lambda) := y - f(\lambda)$$

sowie deren Jacobi-Matrix

$$Dg(\lambda)$$

- Für $k = 0, 1, \dots$:

- Berechne $\delta^{(k)}$ als Lösung des linearen Ausgleichproblems

$$\min \| g(\lambda^{(k)}) + Dg(\lambda^{(k)}) \cdot \delta^{(k)} \|_2^2,$$

d.h. löse konkret das Normalgleichungssystem

$$Dg(\lambda^{(k)})^T Dg(\lambda^{(k)}) \delta^{(k)} = -Dg(\lambda^{(k)})^T \cdot g(\lambda^{(k)})$$

nach $\delta^{(k)}$ auf. Dies wird am stabilsten mit der QR-Zerlegung von $Dg(\lambda^{(k)})$ erreicht:

- Berechne die QR-Zerlegung

$$Dg(\lambda^{(k)}) = Q^{(k)} R^{(k)}$$

- Löse nach $\delta^{(k)}$ auf:

$$R^{(k)} \delta^{(k)} = -Q^{(k)T} g(\lambda^{(k)})$$

- Setze

$$\lambda^{(k+1)} = \lambda^{(k)} + \delta^{(k)}.$$

Die Grösse $\delta^{(k)}$ kann als 'Korrekturrichtung' der Näherung $\lambda^{(k)}$ aufgefasst werden. In der Praxis verwendet man besser das folgende gedämpfte Gauss-Newton-Verfahren. Analog zum gedämpften Newton-Verfahren aus Kapitel 5.4.3 akzeptieren wir die Korrektur $\delta^{(k)}$ nur, wenn sie wirklich zu einer Abnahme des Fehlerfunktionalen führt, also

$$E(\lambda^{(k+1)}) = \| g(\lambda^{(k+1)}) \|_2^2 < \| g(\lambda^{(k)}) \|_2^2 = E(\lambda^{(k)}).$$

erfüllt ist. Dies können wir durch die Dämpfung wie folgt anstreben:

Definition 6.9: Gedämpftes Gauss-Newton-Verfahren

- Sei $\lambda^{(0)}$ ein Startvektor in der Nähe des Minimums von E der Ausgleichsfunktion $f(\lambda)$. Das gedämpfte Gauss-Newton-Verfahren zur näherungsweisen Bestimmung des Minimums lautet:

- Berechne die Funktion

$$g(\lambda) := y - f(\lambda)$$

sowie deren Jacobi-Matrix

$$Dg(\lambda)$$

- Für $k = 0, 1, \dots$:

- Berechne $\delta^{(k)}$ als Lösung des linearen Ausgleichproblems

$$\min \| g(\lambda^{(k)}) + Dg(\lambda^{(k)}) \cdot \delta^{(k)} \|_2^2,$$

d.h. löse konkret das Normalgleichungssystem

$$Dg(\lambda^{(k)})^T Dg(\lambda^{(k)}) \delta^{(k)} = -Dg(\lambda^{(k)})^T g(\lambda^{(k)})$$

nach $\delta^{(k)}$ auf. Dies wird am stabilsten mit der QR-Zerlegung von $Dg(\lambda^{(k)})$ erreicht:

- Berechne die QR-Zerlegung

$$Dg(\lambda^{(k)}) = Q^{(k)} R^{(k)}$$

- Löse nach $\delta^{(k)}$ auf:

$$R^{(k)} \delta^{(k)} = -Q^{(k)T} g(\lambda^{(k)})$$

- Finde das minimale $p \in \{0, 1, \dots, p_{max}\}$ mit

$$\underbrace{\| g\left(\lambda^{(k)} + \frac{\delta^{(k)}}{2^p}\right) \|_2^2}_{\lambda^{(k+1)}} < \| g(\lambda^{(k)}) \|_2^2$$

- Falls kein minimales p gefunden werden kann, rechne mit $p = 0$ weiter

- Setze

$$\lambda^{(k+1)} = \lambda^{(k)} + \frac{\delta^{(k)}}{2^p}.$$

Bemerkungen [1]:

- Ein Vorteil des gedämpften Gauss-Newton-Verfahrens ist, dass es i.d.R. für einen grösseren Bereich von Startvektoren konvergiert als das ungedämpfte Gauss-Newton-Verfahren. Durch das fortlaufende Anpassen der 'Korrekturrichtung' kann es sozusagen noch mit Startvektoren umgehen, die weiter entfernt sind vom Optimum. Die Dämpfung ist aber keine Garantie für Konvergenz. Man benötigt noch weitere Bedingungen, auf die wir hier nicht eingehen werden.
- Als Abbruchkriterium des Algorithmus kann z.B.

$$\| \frac{\delta^{(k)}}{2^p} \|_2 < TOL$$

verwendet werden. Wiederum ist das aber keine Garantie, dass die berechnete Näherung einen maximalen Abstand von TOL zum gesuchten Minimum besitzt.

Beispiel 6.10 [1]:

- Wir nehmen nochmals das Problem aus Beispiel 6.8. Wir wollen also eine Ansatzfunktion $f(x) = ae^{bx}$ bestmöglich im Sinn der kleinsten Fehlerquadrate an die gegebenen Daten anpassen und vergleichen die Resultate des ungedämpften und des gedämpften Gauss-Newton-Verfahrens, einmal für den Startvektor $\lambda^{(0)} = (a^{(0)}, b^{(0)})^T = (1, -1.5)^T$ und dann nochmals für den Startvektor $\lambda^{(0)} = (a^{(0)}, b^{(0)})^T = (2, 2)^T$

x_i	0	1	2	3	4	
y_i	3	1	0.5	0.2	0.05	

- Lösung:

$$\mathbf{g}(a, b) := \mathbf{y} - \mathbf{f}(a, b) = \begin{pmatrix} g_1(a, b) \\ \vdots \\ g_5(a, b) \end{pmatrix} = \begin{pmatrix} y_1 - ae^{bx_1} \\ \vdots \\ y_5 - ae^{bx_5} \end{pmatrix} = \begin{pmatrix} 3 - ae^{b \cdot 0} \\ 1 - ae^{b \cdot 1} \\ 0.5 - ae^{b \cdot 2} \\ 0.2 - ae^{b \cdot 3} \\ 0.05 - ae^{b \cdot 4} \end{pmatrix}$$

$$\mathbf{D}\mathbf{g}(a, b) := \begin{pmatrix} \frac{\partial g_1}{\partial a}(a, b) & \frac{\partial g_1}{\partial b}(a, b) \\ \frac{\partial g_2}{\partial a}(a, b) & \frac{\partial g_2}{\partial b}(a, b) \\ \vdots & \vdots \\ \frac{\partial g_5}{\partial a}(a, b) & \frac{\partial g_5}{\partial b}(a, b) \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ -e^{b \cdot 1} & -ae^{b \cdot 1} \\ -e^{b \cdot 2} & -2ae^{b \cdot 2} \\ -e^{b \cdot 3} & -3ae^{b \cdot 3} \\ -e^{b \cdot 4} & -4ae^{b \cdot 4} \end{pmatrix}$$

Für den ersten Schritt des ungedämpften Verfahrens erhalten wir also

$$\mathbf{D}\mathbf{g}(1, -1.5) = \begin{pmatrix} -1 & 0 \\ -e^{-1.5} & -e^{-1.5} \\ -e^{-3} & -2e^{-3} \\ -e^{-4.5} & -3e^{-4.5} \\ -e^{-6} & -4e^{-6} \end{pmatrix}, \quad \mathbf{g}(1, -1.5) = \begin{pmatrix} 3 - 1 \\ 1 - e^{-1.5} \\ 0.5 - e^{-3} \\ 0.2 - e^{-4.5} \\ 0.05 - e^{-6} \end{pmatrix}$$

und das lineare Gleichungssystem

$$\begin{pmatrix} -1 & -e^{-1.5} & -e^{-3} & -e^{-4.5} & -e^{-6} \\ 0 & -e^{-1.5} & -2e^{-3} & -3e^{-4.5} & -4e^{-6} \end{pmatrix} \begin{pmatrix} -1 & 0 \\ -e^{-1.5} & -e^{-1.5} \\ -e^{-3} & -2e^{-3} \\ -e^{-4.5} & -3e^{-4.5} \\ -e^{-6} & -4e^{-6} \end{pmatrix} \begin{pmatrix} \delta_1^{(0)} \\ \delta_2^{(0)} \end{pmatrix}$$

$$= - \begin{pmatrix} -1 & -e^{-1.5} & -e^{-3} & -e^{-4.5} & -e^{-6} \\ 0 & -e^{-1.5} & -2e^{-3} & -3e^{-4.5} & -4e^{-6} \end{pmatrix} \cdot \begin{pmatrix} 3 - 1 \\ 1 - e^{-1.5} \\ 0.5 - e^{-3} \\ 0.2 - e^{-4.5} \\ 0.05 - e^{-6} \end{pmatrix}$$

bzw.

$$\begin{pmatrix} 1.0524 & 0.0551 \\ 0.0551 & 0.0609 \end{pmatrix} \begin{pmatrix} \delta_1^{(0)} \\ \delta_2^{(0)} \end{pmatrix} = \begin{pmatrix} 2.1980 \\ 0.2249 \end{pmatrix}$$

mit der Lösung

$$\begin{pmatrix} \delta_1^{(0)} \\ \delta_2^{(0)} \end{pmatrix} = \begin{pmatrix} 1.9894 \\ 1.8920 \end{pmatrix}$$

und damit

$$\begin{aligned} \boldsymbol{\lambda}^{(1)} &= \boldsymbol{\lambda}^{(0)} + \boldsymbol{\delta}^{(0)} \\ &= \begin{pmatrix} 1 \\ -1.5 \end{pmatrix} + \begin{pmatrix} 1.9894 \\ 1.8920 \end{pmatrix} = \begin{pmatrix} 2.9894 \\ 0.3920 \end{pmatrix} \end{aligned}$$

- Die weiteren Lösungen des ungedämpften Verfahrens sind

i	0	1	2	5	10
$\boldsymbol{\lambda}^{(i)}$	$\begin{pmatrix} 1 \\ -1.5 \end{pmatrix}$	$\begin{pmatrix} 2.99 \\ 0.392 \end{pmatrix}$	$\begin{pmatrix} 1.26 \\ 0.279 \end{pmatrix}$	$\begin{pmatrix} 2.91 \\ -0.856 \end{pmatrix}$	$\begin{pmatrix} 2.981658705 \\ -1.003280776 \end{pmatrix}$

Ab $\boldsymbol{\lambda}^{(13)} = (2.981658972, -1.003281352)^T$ tritt keine Veränderung der Ziffern mehr ein. Dies stimmt mit dem Ergebnis aus Bsp. 6.8 überein. Für den Startvektor $\boldsymbol{\lambda}^{(0)} = (2, 2)^T$ tritt hingegen keine Konvergenz ein, so erhalten wir z.B. $\boldsymbol{\lambda}^{(13)} = (0.3698, 33.6868)^T$.

- Dagegen sind die Ergebnisse des gedämpften Gauss-Newton-Verfahrens:

i	0	1	2	3	4
$\lambda^{(i)}$	$\begin{pmatrix} 1 \\ -1.5 \end{pmatrix}$	$\begin{pmatrix} 1.99 \\ -0.554 \end{pmatrix}$	$\begin{pmatrix} 2.919 \\ -0.951 \end{pmatrix}$	$\begin{pmatrix} 2.980 \\ -0.999 \end{pmatrix}$	$\begin{pmatrix} 2.981516868 \\ -1.002965939 \end{pmatrix}$

Ab $\lambda^{(7)} = (2.981658324, -1.003279952)^T$ tritt keine Veränderung der Ziffern mehr ein. Man sieht dass das gedämpfte Gauss-Newton Verfahren bei gleichem Startvektor wesentlich schneller konvergiert. Für den Startvektor $\lambda^{(0)} = (2, 2)^T$ tritt ebenfalls Konvergenz ein:

i	0	1	2	5	10
$\lambda^{(i)}$	$\begin{pmatrix} 2 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 0.00384 \\ 2.00 \end{pmatrix}$	$\begin{pmatrix} 0.00384 \\ 1.75 \end{pmatrix}$	$\begin{pmatrix} 0.207 \\ -0.752 \end{pmatrix}$	$\begin{pmatrix} 2.981652024 \\ -1.003266310 \end{pmatrix}$

Ab $\lambda^{(14)} = (2.981658971, -1.003281352)^T$ tritt keine Veränderung der Ziffern mehr auf. Man sieht, dass das gedämpfte Gauss-Newton-Verfahren auch für Startvektoren konvergiert, für die das ungedämpfte Gauss-Newton-Verfahren nicht konvergiert.

Aufgabe 6.7 [1]:

- Bestimmen Sie mit dem Newton-Gauss Algorithmus für die folgenden Daten die Funktion der Form $f(x) = a \ln(x + b)$, die diese Daten bestmöglich im Sinne der kleinsten Fehlerquadrate approximiert bis auf eine Toleranz von 1e-5:

x_i	1	2	3	4
y_i	7.1	7.9	8.3	8.8

Kapitel 7

Numerische Integration

Aus der Analysis sind uns die Problemstellungen der Differential- bzw. Integralrechnungen bereits bekannt. In diesem Kapitel werden wir nun auf einige Verfahren eingehen, mit denen einige Problemstellungen der numerischen Integration nicht analytisch, sondern numerisch angegangen werden können.

Lernziele:

- Sie kennen die wichtigsten Verfahren der numerischen Integration und können damit bestimmte Integrale berechnen. Sie können die dabei auftretenden Fehler bestimmen.
- Sie können die Romberg-Extrapolation anwenden.
- Sie können die hier vorgestellten Verfahren in Python implementieren.

7.1 Zur historischen Entwicklung¹

Die Problemstellungen, Flächen zu berechnen oder Tangenten an geometrische Kurven zu legen, wurden bereits in der Antike untersucht. Archimedes (287 – 212 v.Chr.), einer der bedeutendsten Mathematiker der Antike, berechnete unendliche Reihen und ihm gelang die exakte Bestimmung des Flächeninhalts einer von einem Parabelbogen und einer Sekante begrenzten Fläche sowie die Konstruktion von Tangenten an die nach ihm benannte Archimedische Spirale. Das Konzept infinitesimal kleiner Größen blieb aber über lange Zeit unvollständig und unverstanden.

Bis ins 16. Jahrhundert gab es denn auf diesem Gebiet auch nur wenig Fortschritte. Der deutsche Astronom Johannes Kepler (1571 – 1630) benutzte in seinem Werk *Astronomia Nova* (1609) bei der Berechnung der Marsbahn Methoden, die heute als numerische Integration bezeichnet würden. Er versuchte ab 1612, den Rauminhalt von Weinfässern zu berechnen. Die eigentlichen Anfänge der Differentialrechnung gehen aber auf den französische Mathematiker Pierre de Fermat (1601 – 1665) zurück. Er entwickelte 1628 die auch noch heute verwendete Methode, Extremstellen durch Nullsetzen der Tangentensteigung zu berechnen und konnte bereits Tangenten an Kegelschnitte und andere Kurven bestimmen. Sein Landsmann René Descartes (1596 – 1650) entwickelte in *La Géometrie* 1637 eine Methode, Normalen zu berechnen.

Ende des 17. Jahrhunderts, zur Zeit der Frühaufklärung², gelang es dann dem englischen Physiker Isaac Newton (1643 – 1727) und dem deutschen Mathematiker Gottfried Wilhelm Leibniz (1646 – 1716) unabhängig voneinander, widerspruchsfrei funktionierende Kalküle der Infinitesimalrechnung zu entwickeln. Ihre Arbeiten erlaubten das Abstrahieren von einer rein geometrischen Vorstellung. Leibniz entwickelte die heute gebräuchliche Notation mit dem Symbol \int und dx . Newton benutzte sein Kalkül für bahnbrechende Berechnungen in der Mechanik. Im Jahr 1687 erschien sein Hauptwerk *Philosophia Naturalis Principia Mathematica*. Noch heute sprechen wir in der Physik in der klassischen Mechanik von den drei Newtonschen Gesetzen.

¹Hauptsächlich gemäß Wikipedia und http://www-history.mcs.st-andrews.ac.uk/HistTopics/The_rise_of_calculus.html

²Der Begriff Aufklärung bezeichnet seit etwa 1700 das gesamte Vorhaben, durch rationales Denken alle den Fortschritt behindernde Strukturen zu überwinden. Als wichtige Kennzeichen der Aufklärung gelten die Berufung auf die Vernunft als universelle Urteilstinstanz, der Kampf gegen Vorurteile, die Hinwendung zu den Naturwissenschaften, das Plädoyer für religiöse Toleranz und die Orientierung am Naturrecht.

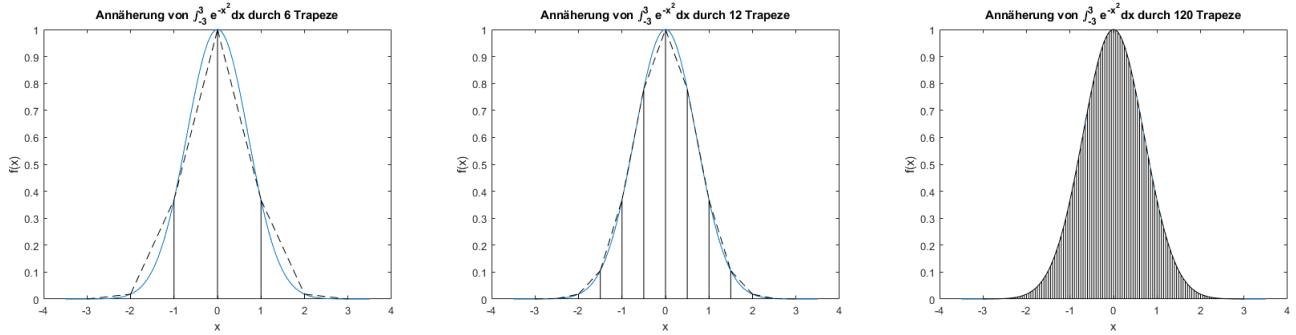


Abbildung 7.1: Ein Beispiel für die Trapezregel. Das bestimmte Integral $\int_{-3}^3 f(x)dx$ mit $f(x) = e^{-x^2}$ (deren Stammfunktion nicht existiert), wird angenähert durch die Summe der Trapezflächen für $n = 6$ (links), $n = 12$ (Mitte) und $n = 120$ (rechts).

Nach Newton und Leibniz wurde die Infinitesimalrechnung durch die Schweizer Mathematiker und Brüder Jakob Bernoulli (1654 – 1705) und Johann Bernoulli (1667 – 1748) weiterentwickelt. Der Begriff des Integrals geht auf Johann Bernoulli zurück. Ebenfalls auf den Werken von Leibniz und Newton setzte der bedeutende Schweizer Mathematiker und Physiker Leonhard Euler (1707 in Basel geboren, 1783 in Petersburg gestorben) auf³, der wegen seiner Beiträge in der Analysis und Zahlentheorie und weiteren Teilgebieten der Mathematik Berühmtheit erlangte.

Im 19. Jahrhundert wurde die gesamte Analysis auf ein solideres Fundament gestellt. 1823 entwickelte der französische Mathematiker Augustin-Louis Cauchy (1789 – 1857) erstmals einen Integralbegriff, der den heutigen Ansprüchen an Stringenz genügt.

7.2 Numerische Integration

Im Gegensatz zur Ableitung, die für alle differenzierbaren Funktionen analytisch berechnet werden kann, können Integrale für eine Vielzahl von Funktionen nicht analytisch gelöst werden. Verfahren zur numerischen Integration (man spricht auch von Quadratur) spielen daher eine wichtige Rolle, wie z.B. bei der numerischen Lösung von Differentialgleichungen in Kap. 8.

7.2.1 Problemstellung

Für eine Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ soll das bestimmte Integral

$$I(f) = \int_a^b f(x)dx$$

auf einem Intervall $[a, b]$ numerisch berechnet werden.

Wir beschränken uns hier auf einige “Klassiker” unter den Quadratur- bzw. Integrationsverfahren, nämlich die Newton-Cotes Formeln im Rahmen der Rechtecks-, Trapez-, und Simpsonregel sowie die Gauss-Formeln und die Romberg-Extrapolation.

Quadraturverfahren haben dabei im Allgemeinen die Form

$$I(f) = \sum_{i=1}^n a_i f(x_i).$$

Dabei nennt man die x_i die *Stützstellen* oder *Knoten* der Quadraturformel und die a_i die *Gewichte*. Die Funktion selbst kann als Funktionsgleichung $y = f(x)$ oder als Wertetabelle $(x_i, f(x_i))$ mit $i = 1, \dots, m$ vorliegen. Ein Beispiel für $f(x) = e^{-x^2}$ ist in Abb. 7.1 dargestellt.

7.2.2 Rechteck- und Trapezregel

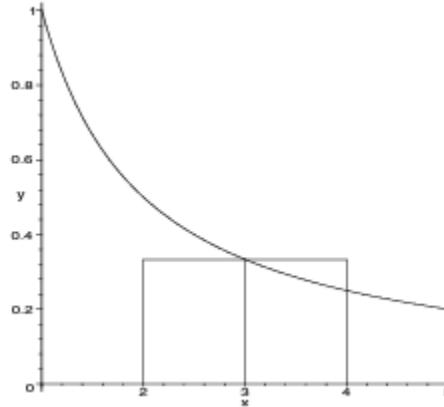
Wir betrachten nun ein Beispiel, wie ein bestimmtes Integral einfach angenähert werden kann.

³Von 1976-1995 auf der Schweizer 10 Franken Note abgebildet.

Beispiel 7.1 [1]

- Es soll $\int_2^4 \frac{1}{x} dx$ angenähert werden, indem $f(x) = \frac{1}{x}$ auf dem Intervall $[2, 4]$ durch eine Konstante ersetzt wird.

Lösung: Wir wählen den Funktionswert in der Mitte des Intervalls $[2, 4]$, also $f(3) = \frac{1}{3}$ und erhalten damit als Näherungswert $\int_2^4 \frac{1}{x} dx \approx (4 - 2) \cdot \frac{1}{3} = \frac{2}{3}$ (vgl. Skizze unten). Zum Vergleich: der exakte Wert ist $\int_2^4 \frac{1}{x} dx = \ln(4) - \ln(2) = 0.6931\dots$.

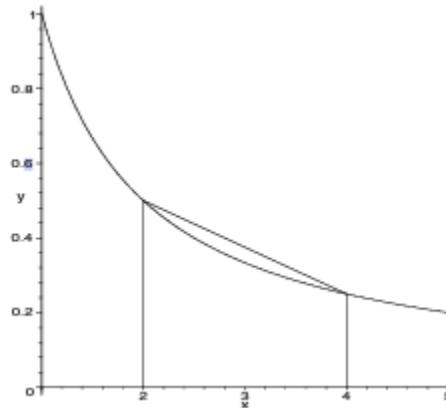


- Jetzt soll $\int_2^4 \frac{1}{x} dx$ durch ein Trapez angenähert werden. Zur Erinnerung: die Fläche eines Trapezes berechnet sich als

$$A = \frac{a + c}{2} \cdot h,$$

wobei a und c die parallelen Grundseiten des Trapezes sind und h seine Höhe.

Lösung: $\int_2^4 \frac{1}{x} dx \approx \frac{f(2) + f(4)}{2} (4 - 2) = 0.75$



Wir haben dabei die einfachste Form der Rechtecks- bzw. Trapezregel verwendet. Ihre Definition ist:

Definition 7.1 [1]: Rechteckregel / Trapezregel

- Die **Rechteckregel** (bzw. Mittelpunktsregel) Rf und die **Trapezregel** Tf zur Approximation von

$$\int_a^b f(x) dx$$

sind definiert als

$$\begin{aligned} Rf &= f\left(\frac{a+b}{2}\right) \cdot (b-a) \\ Tf &= \frac{f(a) + f(b)}{2} \cdot (b-a) \end{aligned}$$

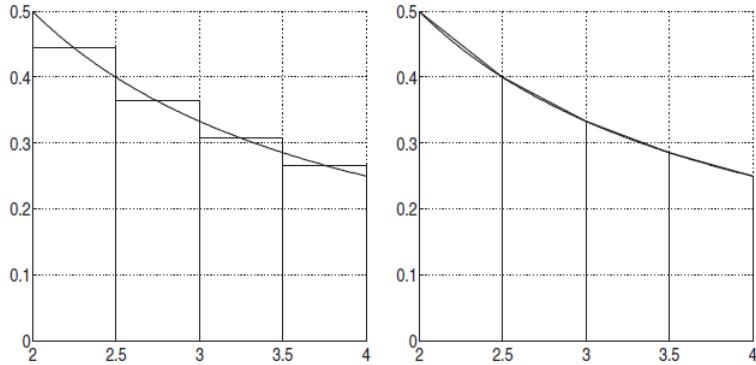


Abbildung 7.2: Summierte Rechteckregel (links) und summierte Trapezregel (rechts) für $n = 4$ zur Berechnung von $\int_2^4 \frac{1}{x} dx$.

Offensichtlich lohnt es sich, zur Steigerung der Genauigkeit das Intervall $[a, b]$ in n Subintervalle zu unterteilen mit der Schrittweite $h = \frac{b-a}{n}$ und anschliessend aufzusummieren (vgl. Abb. 7.2). Dann erhalten wir die summierte Rechtecks- bzw. Trapezregel.

Definition 7.2 [1]: summierte Rechteckregel / summierte Trapezregel

- Sei $f : [a, b] \rightarrow \mathbb{R}$ stetig, $n \in \mathbb{N}$ die Anzahl Subintervalle $[x_i, x_{i+1}]$ auf $[a, b]$ mit der konstanten Breite $h = x_{i+1} - x_i = \frac{b-a}{n}$ und $x_i = a + i \cdot h$ für $i = 0, \dots, n-1$ und $x_n = b$.

Die **summierte Rechteckregel** (bzw. summierte Mittelpunktsregel) $Rf(h)$ und die **summierte Trapezregel** $Tf(h)$ zur Approximation von

$$\int_a^b f(x) dx$$

sind gegeben durch

$$Rf(h) = h \cdot \sum_{i=0}^{n-1} f(x_i + \frac{h}{2})$$

$$Tf(h) = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right)$$

Aufgabe 7.1:

- Berechnen Sie $\int_2^4 \frac{1}{x} dx$ näherungsweise mit der summierten Mittelpunkts- bzw. Trapezregel mit $n = 4$.

7.2.3 Die Simpson-Regel

Wir erhalten die Mittelpunkts- bzw. Rechtecksregel, wenn wir $f(x)$ in $\int_a^b f(x) dx$ durch eine Konstante (Polynom 0. Grades) ersetzen. Wenn wir $f(x)$ durch eine Gerade (Polynom 1. Grades) ersetzen, erhalten wir analog die Trapezregel. Nun können wir noch einen Schritt weitergehen und $f(x)$ durch ein Polynom $p(x)$ 2. Grades ersetzen (vgl. Abb. 7.3).

Wir machen für $p(x)$ und $x \in [a, b]$ den Ansatz

$$p(x) = \alpha + \beta(x - a) + \gamma(x - a)(x - b)$$

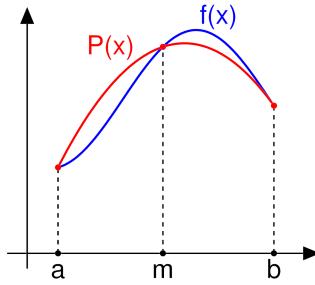


Abbildung 7.3: Die Funktion $f(x)$ wird auf dem Intervall $[a, b]$ durch ein Polynom $P(x)$ an den Stellen $x_1 = a$, $x_2 = m = \frac{b+a}{2}$ und $x_3 = b$ angenähert.

und fordern, dass $p(x)$ an den Stellen $x_1 = a$, $x_2 = \frac{b+a}{2}$ und $x_3 = b$ exakt mit $f(x)$ übereinstimmt, also:

$$\begin{aligned} p(a) &= \alpha \stackrel{!}{=} f(a) \\ p\left(\frac{b+a}{2}\right) &= \alpha + \beta\left(\frac{b+a}{2} - a\right) + \gamma\left(\frac{b+a}{2} - a\right)\left(\frac{b+a}{2} - b\right) = \alpha + \beta\left(\frac{b-a}{2}\right) + \gamma\left(\frac{b-a}{2}\right)\left(\frac{a-b}{2}\right) \stackrel{!}{=} f\left(\frac{b+a}{2}\right) \\ p(b) &= \alpha + \beta(b-a) \stackrel{!}{=} f(b) \end{aligned}$$

Dies ist ein einfache lösbares lineares Gleichungssystem mit den unbekannten α, β, γ . Die Lösung ist dann

$$\begin{aligned} \alpha &= f(a) \\ \beta &= \frac{f(b) - f(a)}{b - a} \\ \gamma &= \frac{f\left(\frac{b+a}{2}\right) - f(a) - \frac{f(b) - f(a)}{b-a} \cdot \left(\frac{b-a}{2}\right)}{-\left(\frac{b-a}{2}\right)^2} = \frac{f(a) - 2f\left(\frac{b+a}{2}\right) + f(b)}{2\left(\frac{b-a}{2}\right)^2}. \end{aligned}$$

Damit ist das Näherungspolynom $p(x)$ eindeutig bestimmt und wir können es mittels der Potenzregel einfach integrieren. Wegen

$$f(x) \approx p(x)$$

gilt also

$$\int_a^b f(x) dx \approx \int_a^b p(x) dx = \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right)$$

Dies ist die *Simpson-Regel*. Nun können wir natürlich wie bei der Rechtecks- und Trapez-Regel die Genauigkeit erhöhen, indem wir das Intervall $[a, b]$ statt nur in eins in n Intervalle mit der Schrittweite $h = \frac{b-a}{n}$ unterteilen und aufsummieren. Wir erhalten dann die *summierte Simpsonregel*:

Definition 7.3 [1]: summierte Simpsonregel

- Sei $f : [a, b] \rightarrow \mathbb{R}$ stetig, $n \in \mathbb{N}$ die Anzahl Subintervalle $[x_i, x_{i+1}]$ auf $[a, b]$ mit der konstanten Breite $h = x_{i+1} - x_i = \frac{b-a}{n}$ und $x_i = a + i \cdot h$ für $i = 0, \dots, n-1$ und $x_n = b$.

Die **summierte Simpsonregel** $Sf(h)$ zur Approximation von

$$\int_a^b f(x) dx$$

ist gegeben durch

$$Sf(h) \equiv \frac{h}{3} \left(\frac{1}{2}f(a) + \sum_{i=1}^{n-1} f(x_i) + 2 \sum_{i=1}^n f\left(\frac{x_{i-1} + x_i}{2}\right) + \frac{1}{2}f(b) \right)$$

Aufgabe 7.2:

- Zeigen Sie, dass die summierte Simpsonregel als gewichtetes Mittel der summierten Trapez- und Rechteckregel interpretiert werden kann:

$$Sf(h) = \frac{1}{3} (Tf(h) + 2Rf(h))$$

- Berechnen Sie $\int_2^4 \frac{1}{x} dx$ näherungsweise mit der summierten Simpsonregel mit $n = 4$.

Aufgabe 7.3:

- Implementieren Sie die summierte Rechteck-, Trapez- und Simpsonregel in Python und überprüfen Sie Ihre manuellen Resultate aus Aufgaben 7.1 und 7.2.

7.2.4 Der Fehler der summierten Quadraturformeln

Der Fehler einer Näherung ist wie immer definiert als der Betrag der Differenz zwischen dem exakten Wert und der Näherung. Für die summierte Rechteckregel $Rf(h)$, die summierte Trapezregel $Tf(h)$ und die summierte Simpsonregel $Sf(h)$ gelten die folgenden Fehlerabschätzungen (ohne Beweis):

Satz 7.1 [1]: Fehlerabschätzung für summierte Quadraturformeln

$$\begin{aligned} \left| \int_a^b f(x) dx - Rf(h) \right| &\leq \frac{h^2}{24} (b-a) \cdot \max_{x \in [a,b]} |f''(x)| \\ \left| \int_a^b f(x) dx - Tf(h) \right| &\leq \frac{h^2}{12} (b-a) \cdot \max_{x \in [a,b]} |f''(x)| \\ \left| \int_a^b f(x) dx - Sf(h) \right| &\leq \frac{h^4}{2880} (b-a) \cdot \max_{x \in [a,b]} |f^{(4)}(x)| \end{aligned}$$

Aufgabe 7.4:

- Sie wollen

$$I = \int_0^{0.5} e^{-x^2} dx$$

näherungsweise mit der summierten Trapezregel auf einen absoluten Fehler von maximal 10^{-5} genau berechnen. Bestimmen Sie eine geeignete Schrittweite h und berechnen Sie entsprechend den Wert der summierten Trapezregel.

7.2.5 Gauss-Formeln

Bisher haben wir die Stützstellen x_i äquidistant gewählt (d.h. die Schrittweite $h = \frac{b-a}{n}$ war konstant) und damit die Rechtecks-, die Trapez- und die Simpson-Formel hergeleitet. Diese drei Formeln werden auch als Newton-Cotes Formeln der Ordnung $N = 0, 1$ und 2 bezeichnet (entspricht dem Grad des verwendeten Polynoms).

Die Stützstellen x_i müssen jedoch nicht zwingend äquidistant sein, sondern können so gewählt werden, dass sie das Integral $\int_a^b f(x) dx$ 'optimal' approximieren. Man erhält dann die sogenannten Gauss-Formeln. Dafür werden die Stützstellen x_i und die Gewichte a_i in der generellen Quadraturformel

$$I(f) = \sum_{i=1}^n a_i f(x_i)$$

so gewählt, dass die 'Fehlerordnung' möglichst hoch wird bzw. der Fehler

$$| \int_a^b f(x)dx - I(f) |$$

möglichst klein.

Wir wollen an dieser Stelle auf die genaue Herleitung verzichten und begnügen uns damit, die Gaussformeln für $n = 1, 2$ und 3 anzugeben (also für eine, zwei und drei Stützstellen). Wir haben dann

Satz 7.2 [1]: Gauss Formeln für $n = 1, 2, 3$:

- Die Gauss Formeln für $n = 1, 2, 3$ für $\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i=1}^n a_i f(x_i)$ lauten:
 - $n = 1: G_1 f = (b-a) \cdot f(\frac{b+a}{2})$
 - $n = 2: G_2 f = \frac{b-a}{2} \left[f\left(-\frac{1}{\sqrt{3}} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) + f\left(\frac{1}{\sqrt{3}} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) \right]$
 - $n = 3: G_3 f = \frac{b-a}{2} \left[\frac{5}{9} \cdot f\left(-\sqrt{0.6} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) + \frac{8}{9} \cdot f\left(\frac{b+a}{2}\right) \right] + \frac{b-a}{2} \left[\frac{5}{9} \cdot f\left(\sqrt{0.6} \cdot \frac{b-a}{2} + \frac{b+a}{2}\right) \right]$

Aufgabe 7.5:

- Berechnen Sie das Integral

$$I = \int_0^{0.5} e^{-x^2} dx$$

mit der summierten Rechtecks-, Trapez-, und Simpsonformel für $n = 3$ und vergleichen Sie den Wert mit der Gaussformel $G_3 f$.

7.2.6 Romberg-Extrapolation

Wir können für die Integration ein Extrapolationsschema verwenden, welches erlaubt, aus einigen Anfangsnäherungen für ein bestimmtes Integral einen genaueren Wert zu extrapolieren. Es so möglich, die mit der summierten Trapezformel berechneten Werte auf einfache Weise zu verbessern:

Satz 7.3 [1]: Romberg-Extrapolation

- Für die summierte Trapezregel $Tf(h)$ zur näherungsweisen Berechnung von $I(f) = \int_a^b f(x) dx$ gilt:

Sei $T_{j0} = Tf\left(\frac{b-a}{2^j}\right)$ für $j = 0, 1, \dots, m$. Dann sind durch die Rekursion

$$T_{jk} = \frac{4^k \cdot T_{j+1,k-1} - T_{j,k-1}}{4^k - 1}$$

für $k = 1, 2, \dots, m$ und $j = 0, 1, \dots, m-k$ Näherungen der Fehlerordnung $2k+2$ gegeben. Diese Methode heisst **Romberg-Extrapolation**. Die verwendete Schrittweitenfolge $h_j = \frac{b-a}{2^j}$ heisst auch Romberg-Folge.

Wir müssen also zuerst für $k = 0$ die T_{j0} mit der summierten Trapezformel für die fortlaufend halbierten Schrittweiten $h_j = \frac{b-a}{2^j}$ ($j = 0, 1, 2, \dots, m$) berechnen, z.B. $h, \frac{h}{2}, \frac{h}{4}, \frac{h}{8}$ für $m = 3$. Damit erhalten wir $T_{00}, T_{10}, T_{20}, T_{30}$ in der ersten Spalte des Extrapolationsschemas (s.u.). Anschliessend können wir diese Werte extrapolieren und erhalten die T_{jk} (für $k = 1, 2, \dots, m$) ohne grossen Rechenaufwand, wie im Schema dargestellt. Natürlich muss jeweils die Anzahl $n_j = 2^j$ der Intervalle für die Berechnung der ersten Spalte T_{j0} mit der summierten Trapezregel

angepasst werden. Das heisst, für jede Erhöhung von j müssen mehr Funktionsauswertungen gemacht werden, denn es gilt

$$T_{j0} = Tf(h_j) = h_j \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n_j-1} f(x_i) \right) \text{ mit } h_j = \frac{b-a}{2^j} \text{ und } n_j = 2^j \text{ und } x_i = a + ih_j.$$

Man erhält für die Extrapolation folgendes Schema (für z.B. $m = 3$):

T_{j0}	T_{j1}	T_{j2}	T_{j3}
T_{00}			
	$T_{01} = \frac{4T_{10}-T_{00}}{3}$		
T_{10}		$T_{02} = \frac{16T_{11}-T_{01}}{15}$	
	$T_{11} = \frac{4T_{20}-T_{10}}{3}$		$T_{03} = \frac{64T_{12}-T_{02}}{63}$
T_{20}		$T_{12} = \frac{16T_{21}-T_{11}}{15}$	
	$T_{21} = \frac{4T_{30}-T_{20}}{3}$		
T_{30}			

Das Extrapolationsschema lässt sich durch die Erhöhung von m beliebig nach unten (zusätzliche Zeilen) und demzufolge nach rechts (zusätzliche Spalten) erweitern.

Beispiel 7.2:

- Berechnen Sie

$$I = \int_2^4 \frac{1}{x} dx$$

näherungsweise mit der summierten Trapezregel und anschliessender Extrapolation ausgehend von den Schrittweiten $h_j = \frac{b-a}{n_j} = \frac{4-2}{2^j}$ für $j = 0, 1, 2, 3$. Denken Sie daran, dass sich die jeweilige Anzahl $n_j = 2^j$ der Intervalle in $T_{j0} = Tf(h_j)$ jeweils verdoppelt, und damit auch die Funktionsauswertungen $f(x_i)$ für die $x_i = a + ih_j$ zunehmen. Für die grafische Darstellung siehe Abb. 7.4.

- Lösung:

- Zuerst berechnen wir die erste Spalte des Schemas mittels der summierten Trapezformel:

$$\begin{aligned}
j = 0, h_0 &= \frac{4-2}{2^0} = 2, n_0 = 2^0 = 1 \Rightarrow T_{00} &= h_0 \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^0 f(x_i) \right) \\
&= h_0 \cdot \frac{f(a) + f(b)}{2} \\
&= h_0 \cdot \frac{f(2) + f(4)}{2} \\
&= 2 \cdot \frac{\frac{1}{2} + \frac{1}{4}}{2} \\
&= 0.75
\end{aligned}$$

$$\begin{aligned}
j = 1, h_1 = \frac{4-2}{2^1} = 1, n_1 = 2^1 = 2 \Rightarrow T_{10} &= h_1 \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^1 f(x_i) \right) \\
&= h_1 \cdot \left(\frac{f(a) + f(b)}{2} + f(x_1) \right) \\
&= h_1 \cdot \left(\frac{f(2) + f(4)}{2} + f(3) \right) \\
&= 1 \cdot \left(\frac{\frac{1}{2} + \frac{1}{4}}{2} + \frac{1}{3} \right) \\
&= 0.7083333... \\
j = 2, h_2 = \frac{4-2}{2^2} = \frac{1}{2}, n_2 = 2^2 = 4 \Rightarrow T_{20} &= h_2 \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^3 f(x_i) \right) = \\
&= h_2 \cdot \left(\frac{f(a) + f(b)}{2} + f(x_1) + f(x_2) + f(x_3) \right) \\
&= h_2 \cdot \left(\frac{f(2) + f(4)}{2} + f(2.5) + f(3) + f(3.5) \right) \\
&= \frac{1}{2} \cdot \left(\frac{\frac{1}{2} + \frac{1}{4}}{2} + \frac{1}{2.5} + \frac{1}{3} + \frac{1}{3.5} \right) \\
&= 0.6970238... \\
j = 3, h_3 = \frac{4-2}{2^3} = \frac{1}{4}, n_3 = 2^3 = 8 \Rightarrow T_{30} &= h_3 \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^7 f(x_i) \right) = \\
&= h_3 \cdot \left(\frac{f(a) + f(b)}{2} + f(x_1) + f(x_2) + \dots + f(x_7) \right) \\
&= h_3 \cdot \left(\frac{f(2) + f(4)}{2} + f(2.25) + f(2.5) + \dots + f(3.75) \right) \\
&= \frac{1}{4} \cdot \left(\frac{\frac{1}{2} + \frac{1}{4}}{2} + \frac{1}{2.25} + \frac{1}{2.5} + \dots + \frac{1}{3.75} \right) \\
&= 0.6941218...
\end{aligned}$$

– Die weiteren Spalten erhalten wir mit der Romberg-Extrapolation:

h	T_{j0}	T_{j1}	T_{j2}	T_{j3}
2	0.75000000			
1	0.70833333	0.69444444		
0.5	0.69702381	0.69325397	0.6931746	0.69314748
0.25	0.69412185	0.69315453	0.6931479	

– Die zugehörigen Fehler $E_{jk} = |T_{jk} - I|$ sind:

h	E_{j0}	E_{j1}	E_{j2}	E_{j3}
2	5.68528194e - 02			
1	1.51861528e - 02	1.29726388e - 03		
0.5	3.87662896e - 03	1.06787694e - 04	2.74226147e - 05	2.97084887e - 07
0.25	9.74669812e - 04	7.35009459e - 06	7.20921290e - 07	

Die Fehler nehmen wie erwartet von oben nach unten respektive von links nach rechts ab. Sind in der ersten Spalte die T_{j0} berechnet, sind keine weiteren Auswertung der zu integrierenden Funktion $f(x)$ mehr nötig.

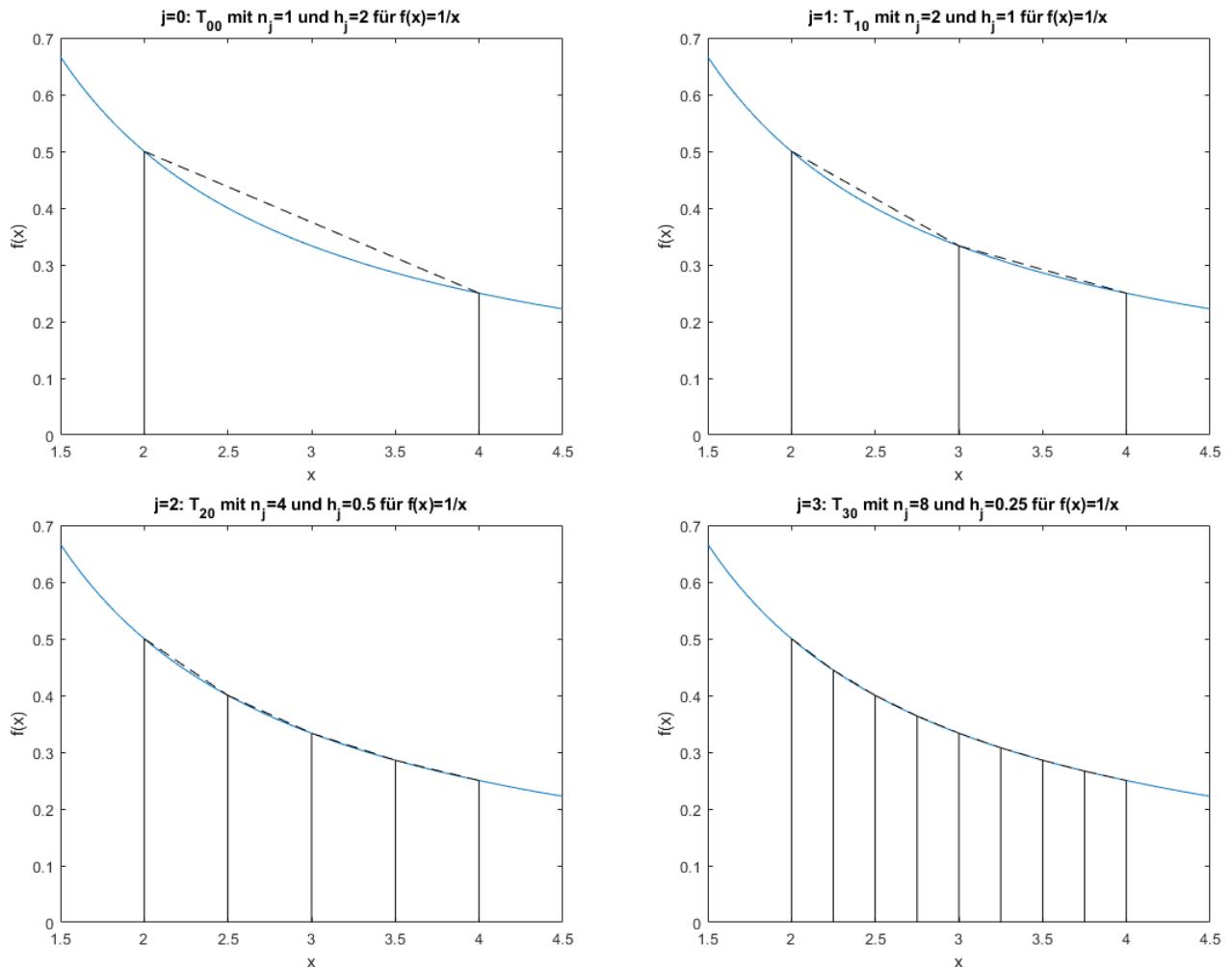
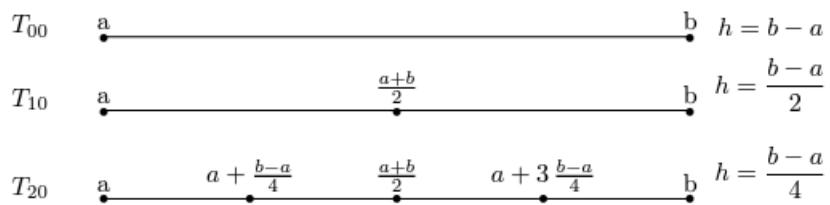


Abbildung 7.4: Abbildung der Summe der Trapezflächen T_{00} (oben links) bis T_{30} (unten rechts) zu Bsp. 7.2.

Bemerkungen:

1. Bei der Berechnung der T_{j0} in der ersten Spalte kann man Auswertungen der zu integrierenden Funktion einsparen, wie im Folgenden erläutert:
 - (a) für T_{00} werden nur die Funktionsauswertungen $f(a)$ und $f(b)$ benötigt
 - (b) für T_{10} wird zusätzlich zu $f(a)$ und $f(b)$ noch $f\left(\frac{a+b}{2}\right)$ benötigt
 - (c) für T_{20} kommen zusätzlich zu $f(a)$, $f(b)$, $f\left(\frac{a+b}{2}\right)$ noch $f\left(a + \frac{b-a}{4}\right)$ und $f\left(a + 3\frac{b-a}{4}\right)$
 - (d) etc.

Dies lässt sich folgendermassen illustrieren:



Darauf basierend lässt sich die summierte Trapezregel zu den halbierten Schrittweiten rekursiv formulieren, d.h. man berechnet die T_{j0} für $j \geq 1$ allein aus dem vorhergehenden Wert von $T_{j-1,0}$ und zwar über die folgende Rekursionsformel (ohne Beweis):

$$T_{j0} = \frac{1}{2}T_{j-1,0} + h_j \sum_{i=1}^{n_{j-1}} f(a + (2i-1)h_j)$$

wobei gilt $n_{j-1} = 2^{j-1}$ und $h_j = h_{j-1}/2$.

2. Betrachtet man den Extrapolationsschritt von T_{00} und T_{10} nach T_{01} nochmal genauer, so findet man:

$$\begin{aligned} T_{01} &= \frac{4T_{10} - T_{00}}{3} \\ &= \frac{1}{3} \left(4 \left(\frac{b-a}{2} \cdot \frac{f(a) + f(b)}{2} + f\left(\frac{a+b}{2}\right) \right) - \frac{b-a}{2}(f(a) + f(b)) \right) \\ &= \frac{b-a}{6} (f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)) = Sf. \end{aligned}$$

Das bedeutet, dass in der zweiten Spalte des Schemas nichts anderes steht als die Simpsonregel Sf steht (die analogen Resultate erhält man für T_{02} etc.).

Aufgabe 7.6:

- Berechnen Sie das Integral

$$I = \int_0^{0.5} e^{-x^2} dx$$

mit der summierten Trapezformel und Extrapolation für die Schrittweiten $h_j = \frac{b-a}{2^j}$ ($j = 0, 1, 2, 3$).

Kapitel 8

Einführung in gewöhnliche Differentialgleichungen

Prozesse in Natur und Technik lassen sich häufig nur mit Differentialgleichungen beschreiben. Die Lösung solcher Gleichungen ist deshalb eminent wichtig. Dabei lassen sich nur wenige Differentialgleichungen mit analytischen Verfahren lösen, die grosse Mehrzahl muss mit numerischen Lösungsverfahren angegangen werden. Wir wollen in diesem Kapitel einige einfache Lösungsverfahren betrachten und beschränken uns hierbei auf Anfangswertprobleme. Angesichts der Komplexität und Vielfalt von Differentialgleichungen kann dieses Kapitel nur als Einführung in die Problematik gesehen werden.

Lernziele:

- Sie kennen die Definitionen einer gewöhnlichen Differentialgleichung n -ter Ordnung und eines Anfangswertproblems.
- Sie verstehen das Prinzip eines Richtungsfeldes und können Richtungsfelder interpretieren und zeichnen.
- Sie können gewöhnliche Differentialgleichungen lösen basierend auf dem Euler-Verfahren, dem modifizierten Euler-Verfahren, dem Mittelpunkt-Verfahren und den Runge-Kutta Verfahren.
- Sie können Differentialgleichungen n -ter Ordnung in ein System von n Differentialgleichungen 1. Ordnung umwandeln und diese Systeme lösen.
- Sie kennen den Begriff der Stabilität.

8.1 Zur historischen Entwicklung

Die Geschichte der Differentialgleichungen ist Teil der Geschichte der Differentialrechnung, so erstaunt es nicht, dass die Anfänge wieder auf Isaac Newton (1643 – 1727) und Gottfried Wilhelm Leibniz (1646 – 1716) zurück gehen. Vor ihnen beschäftigte sich aber bereits Galileo Galilei (1541 – 1642) mit der Beschreibung des freien Falls (siehe Kap. 8.3) und war um 1609 in der Lage, diesen mathematisch korrekt zu beschreiben. Er erkannte, dass im Vakuum alle Körper gleich schnell fallen und ihre Bewegung gleichmäßig beschleunigt ist (ein Umstand, den allerdings bereits der römische Dichter und Philosoph Lukrez um 55 v. Chr. in seinem Werk “De rerum natura” postulierte: “Deshalb müssen die Körper mit gleicher Geschwindigkeit alle trotz ungleichem Gewicht durch das ruhende Leere sich stürzen”).

In seinem Werk “Methodus fluxionum et Serierum Infinitarum” beschäftigte sich Newton 1671 mit drei Arten von Differentialgleichungen und löste als Beispiele

$$\begin{aligned} (\dot{y})^2 &= \dot{x}\dot{y} + (\dot{x})^2 x^2 \\ \dot{y}ax - \dot{x}xy - aax &= 0 \\ 2\dot{x} - \dot{z} + \dot{y}x &= 0 \end{aligned}$$

Im Jahr 1687 veröffentlichte Newton dann in *Philosophiae Naturalis Principia* seine drei Gesetze der klassischen Mechanik, welche die Bewegung von Körpern letztlich in der Form von Differentialgleichungen beschreiben. In einer modernen Formulierung lauten die ersten beiden Gesetze

- 1. Newtonsches Gesetz: $\sum \mathbf{F} = 0 \Leftrightarrow \frac{d\mathbf{v}}{dt} = 0$
- 2. Newtonsche Gesetz: $\mathbf{F} = m \frac{d\mathbf{v}}{dt} = m\mathbf{a}$

Der französische Physiker und Aufklärer Jean-Baptiste le Rond, genannt D'Alembert, (1717 - 1783) beschrieb die Schwingung einer Saite bereits anhand der eindimensionalen homogenen Wellengleichung

$$\frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0.$$

Leonhard Euler (1707 - 1783), ebenfalls involviert in der Diskussion um die Ausbreitung von Wellen, entwickelte unter anderem in der Strömungsmechanik die Euler-Gleichungen zur Beschreibung der Strömung von reibungsfreien Fluiden. Der Impulssatz lautet in der heutigen Form

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} + \frac{1}{\rho} \nabla p = \mathbf{0},$$

wobei $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$ der Geschwindigkeitsvektor, ρ die Dichte, $p = p(\mathbf{x}, t)$ der Druck, \mathbf{x} der Ortsvektor, t die Zeit und $\nabla = \left(\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n} \right)$ der Nabla-Operator ist. In 1768 publizierte Euler das nach ihm benannte Lösungsverfahren für gewöhnliche Differentialgleichungen (siehe Kap. 8.5). Augustin-Louis Cauchy (1789 – 1857) beweist dessen Konvergenz 1824. Der französische Mathematiker Claude Louis Marie Henri Navier (1785-1836) und der irische Mathematiker Sir George Gabriel Stokes (1819 - 1903) entwickeln unabhängig voneinander in der ersten Hälfte des 19. Jahrhunderts die Navier-Stokes Gleichungen, eine Erweiterung der Euler-Gleichungen unter Einbezug der Viskosität (Reibung) von Fluiden. Um 1855 werden die Mehrschritt-Lösungsverfahren des englischen Mathematikers und Astronoms John Couch Adams (1819 - 1892) in einem Brief des Ballistikers Francis Bashforth (1819 - 1912) erstmals erwähnt. Die deutschen Mathematiker Carl David Tolmé Runge (1856 - 1927) und Martin Wilhelm Kutta (1867 - 1944) entwickeln anfangs des 20. Jahrhunderts in Göttingen die Runge-Kutta-Lösungsverfahren für Anfangswertprobleme.

Die amerikanischen Chemiker Charles Francis Curtiss (1921 - 2007) und Joseph Oakland Hirschfelder (1911 - 1990) publizieren 1952 ihre BDF (Backward Difference Formulas) Mehrschrittverfahren zur Lösung von sogen. steifen Differentialgleichungen. Der schwedische Mathematiker Germund Dahlquist (1925 - 2005) leistet bahnbrechende Beiträge zur Theorie der numerischen Lösung von Anfangswertproblemen, insbesondere zum Thema Stabilität.

8.2 Problemstellung

In vielen Prozessen in der Natur oder Technik spielen zeitliche Änderungen von beobachtbaren Größen ('Observablen'), nennen wir sie z.B. $y(t)$, eine wichtige Rolle. Häufig ist die zeitliche Änderungen einer solchen Größe, also $\frac{dy}{dt}$, abhängig von der Größe $y(t)$ selbst, d.h. wir können $\frac{dy}{dt}$ durch eine Funktion f ausdrücken, die von der Zeit t aber eben auch von der eigentlichen Größe $y(t)$ abhängt:

$$\frac{dy}{dt} = f(t, y(t)). \quad (8.1)$$

Eine solche Gleichung nennt man auch 'gewöhnliche Differentialgleichung 1. Ordnung'¹. Erster Ordnung deshalb, weil in diesem Fall nur die erste Ableitung auftritt.

Beispiel 8.1:

- Radioaktiver Zerfall: die Abnahme $\frac{dn}{dt}$ der Anzahl $n = n(t)$ der Atome eines instabilen Isotops über die Zeit hängt einerseits von n selbst ab (d.h. je weniger Atome vorhanden sind, um so weniger zerfallen pro Zeiteinheit) und andererseits von einer (konstanten) Zerfallsrate λ ab gemäß:

$$\frac{dn}{dt} = -\lambda n$$

mit der Lösung $n(t) = n_0 e^{-\lambda t}$ für die Anfangswertbedingung $n(t=0) = n_0$.

¹Im Englischen spricht man von 'Ordinary Differential Equation', kurz ODE.

- Beliebige weitere Beispiele von DGL 1. Ordnung gemäss Gl. 8.1:

$$\begin{aligned}\frac{dy}{dt} &= t \cdot y^2 \\ \frac{dy}{dt} &= \cos(t^2) \cdot \sin(ay) \\ &\vdots\end{aligned}$$

Natürlich können aber auch Ableitungen beliebiger, d.h. n -ter Ordnung in solchen Gleichungen auftreten, analog spricht man dann von gewöhnlichen Differentialgleichungen n -ter Ordnung. Ausserdem muss die Variable nicht die Zeit t sein, sondern es kann sich auch um eine beliebige Variable, nennen wir sie x , handeln.

Definition 8.1: Gewöhnliche Differentialgleichung n -ter Ordnung

- Eine Gleichung, in der Ableitungen einer unbekannten Funktion $y = y(x)$ bis zur n -ten Ordnung auftreten, heisst eine *gewöhnliche Differentialgleichung n -ter Ordnung*. Sie hat die explizite Form

$$y^{(n)}(x) = f(x, y(x), y'(x), \dots, y^{(n-1)}(x)) \quad (8.2)$$

Gesucht sind die Lösungen $y = y(x)$ dieser Gleichung, wobei die Lösungen y auf einem Intervall $[a, b]$ definiert sein sollen, $y : [a, b] \rightarrow \mathbb{R}$.

Bemerkung:

- Für die Ableitungen höherer Ordnung von $y(x)$ gibt es verschiedene Notationen:

– Lagrange-Notation (mit Strichen, resp. Hochzahlen in Klammern):

$$y'(x), y''(x), y'''(x), y^{(4)}(x), y^{(5)}(x), \dots, y^{(n)}(x)$$

– Newton-Notation (mit Punkten)

$$\dot{y}(x), \ddot{y}(x), \dddot{y}(x), \ddot{\ddot{y}}(x), \dots$$

– Leibniz-Notation (mit Differentialoperator $\frac{d}{dx}$)

$$\frac{dy(x)}{dx}, \frac{d^2y(x)}{dx^2}, \frac{d^3y(x)}{dx^3}, \dots, \frac{d^n y(x)}{dx^n}$$

oder auch

$$\frac{d}{dx}y(x), \frac{d^2}{dx^2}y(x), \frac{d^3}{dx^3}y(x), \dots, \frac{d^n}{dx^n}y(x)$$

Beispiele:

- DGL 2. Ordnung

$$y''(x) = y'(x)^3 \cdot y(x) + \cos(x)$$

oder

$$\frac{d^2y(x)}{dx^2} = \left(\frac{dy(x)}{dx} \right)^3 \cdot y(x) + \cos(x)$$

- DGL 4. Ordnung (da es sich hier um eine Linearkombination der Ableitungen mit den Koeffizienten a_i handelt, spricht man auch von einer “linearen” DGL)

$$a_5 y^{(4)}(x) + a_4 y'''(x) + a_3 y''(x) + a_2 y'(x) + a_1 y(x) + a_0 = 0$$

oder

$$a_5 \frac{d^4y(x)}{dx^4} + a_4 \frac{d^3y(x)}{dx^3} + a_3 \frac{d^2y(x)}{dx^2} + a_2 \frac{dy(x)}{dx} + a_1 y(x) + a_0 = 0$$

Bemerkungen:

1. Neben den *gewöhnlichen* Differentialgleichungen gibt es noch die *partiellen* Differentialgleichungen. Diese enthalten die partiellen Ableitungen einer unbekannten Funktion mehrerer Variablen. In dieser Vorlesung behandeln wir ausschliesslich gewöhnliche Differentialgleichungen und lassen den Ausdruck 'gewöhnlich' im Folgenden weg und verwenden die Abkürzung DGL. Außerdem beschränken wir uns vorwiegend auf die erste und zweite Ordnung (also $n = 1$ und $n = 2$). Wie wir später sehen werden, kann jede Differentialgleichung höherer Ordnung auf ein System von Differentialgleichungen erster Ordnung zurückgeführt werden.
2. Entsprechend gibt es auch Systeme von Differentialgleichungen, die unter Umständen miteinander gekoppelt sind. Ein Beispiel sind gekoppelte Schwingungen (z.B. zwei durch eine Feder miteinander gekoppelte Fadenpendel) oder das Räuber-Beute Modell, bei dem die zeitliche Entwicklung der Anzahl Räuber die Anzahl Beutetiere beeinflusst und umgekehrt.
3. Die allgemeine Lösung einer Differentialgleichung n -ter Ordnung enthält noch n unabhängige Parameter (denken Sie an die Integrationskonstante bei einem unbestimmten Integral). In den Anwendungen ist man jedoch häufig nur an einer spezifischen Lösung näher interessiert. Insbesondere müssen die n Parameter für die Anwendung numerischer Verfahren bekannt sein. Das heisst, um die n Parameter festlegen zu können, werden noch n zusätzliche Informationen über die gesuchte Lösung benötigt. Sie können z.B. bei einem physikalischen Prozess aus den zum Zeitpunkt $t = 0$ herrschenden Bedingungen bestehen. Je nach Art der Vorgabe unterscheidet man zwischen einem Anfangs- oder Randwertproblem. Wir konzentrieren uns hier auf Anfangswertprobleme.

Definition 8.2: Anfangswertproblem

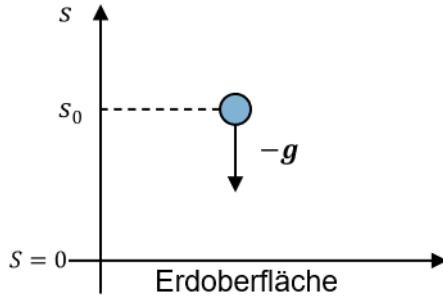
- Bei einem Anfangswertproblem (AWP) für eine Differentialgleichung n -ter Ordnung werden der Lösungsfunktion $y = y(x)$ noch n Werte vorgeschrieben, nämlich der Funktionswert an einer bestimmten Stelle x_0 sowie die Werte der ersten $n - 1$ Ableitungen an der gleichen Stelle.
- Für die hier betrachteten Differentialgleichungen 1. und 2. Ordnung heisst das:
 - Differentialgleichung 1. Ordnung: Gesucht ist diejenige *spezifische* Lösungskurve $y = y(x)$, die durch den vorgegebenen Punkt $P = (x_0, y(x_0))$ verläuft. Gegeben ist beim AWP also die DGL 1. Ordnung $y'(x) = f(x, y(x))$ und der Anfangswert $y(x_0)$.
 - Differentialgleichung 2. Ordnung: Gesucht ist diejenige *spezifische* Lösungskurve $y = y(x)$, die durch den vorgegebenen Punkt $P = (x_0, y(x_0))$ verläuft und im Punkt x_0 die vorgegebene Steigung $y'(x_0) = m$ besitzt. Gegeben ist beim AWP also die DGL 2. Ordnung $y''(x) = f(x, y(x), y'(x))$ und die Anfangswerte $y(x_0)$ und $y'(x_0)$.

8.3 Beispiele aus den Naturwissenschaften

Nach der allgemeinen Formulierung des Problems wollen wir uns nun einige Beispiele aus den Naturwissenschaften anschauen.

8.3.1 Der Freie Fall (aus Papula)

Wir betrachten einen frei fallenden Körper im luftleeren Raum und wollen den Weg bestimmen, den der Körper nach einer gewissen Zeit zurückgelegt hat. Dazu führen wir die von der Erdoberfläche senkrecht nach *oben* gerichtete Koordinatenachse s für den Weg ein. Der Körper unterliegt dann ausschliesslich der Schwerkraft und erfährt in der Nähe der Erdoberfläche die konstante Fallbeschleunigung $a = -g$ (das Minuszeichen bedeutet, dass die Bewegung des Körpers entgegen der Richtung der Koordinatenachse erfolgt).



Wir wissen aus der Physik, dass die Beschleunigung $a(t)$ die Ableitung der Geschwindigkeit $v(t)$ nach der Zeit ist, und die Geschwindigkeit ist die Ableitung des zurückgelegten Weges $s(t)$ nach der Zeit, also:

$$v(t) = \frac{d}{dt}s(t) = \dot{s}(t) \text{ und } a(t) = \frac{d}{dt}v(t) = \dot{v}(t) \text{ bzw. } a(t) = \frac{d}{dt}\left(\underbrace{\frac{d}{dt}s(t)}_{v(t)}\right) = \frac{d^2}{dt^2}s(t) = \ddot{s}(t).$$

Das bedeutet also, wir haben eine Differentialgleichung zweiter Ordnung die wir lösen müssen, nämlich

$$\ddot{s}(t) = -g$$

Die Punkte bedeuten die (erste oder zweite) Ableitung nach der Zeit. Offensichtlich hängt die zweite Ableitung $\ddot{s}(t)$ nicht von $s(t)$ selbst ab sondern nur von der Konstanten g . Das bedeutet, dass wir analytisch integrieren können, um das Problem zu lösen. Wir erhalten dann nach einmal integrieren die Geschwindigkeit

$$v(t) = \int a(t) dt = \int (-g) dt = - \int g dt = -gt + v_0.$$

Hierbei ist die Integrationskonstante v_0 die Geschwindigkeit zum Zeitpunkt $t = 0$. Nochmals integrieren liefert

$$s(t) = \int v(t) dt = \int (-gt + v_0) dt = -\frac{1}{2}gt^2 + v_0 t + s_0.$$

Wieder ist s_0 die Integrationskonstante und gibt den Weg zum Zeitpunkt $t = 0$ an. Die Integrationskonstanten (Parameter) sind somit Anfangshöhe s_0 und Anfangsgeschwindigkeit v_0 . Die oben hergeleitete Beziehung

$$s(t) = -\frac{1}{2}gt^2 + v_0 t + s_0$$

gibt also die allgemeine Lösung der Differentialgleichung zweiter Ordnung

$$\ddot{s}(t) = -g$$

an. Durch Festlegen der Anfangshöhe s_0 und Anfangsgeschwindigkeit v_0 erhalten wir *eine* spezielle Lösung.

Beispiel 8.2:

- Nehmen wir an, die Anfangshöhe sei $s_0 = 1000$ m, die Anfangsgeschwindigkeit sei $v_0 = 0$ und die Fallbeschleunigung $g \approx 10 \frac{\text{m}}{\text{s}^2}$. Was für eine Strecke hat ein aus dieser Höhe fallengelassener Stein nach 3 s zurückgelegt? Wie lange dauert es, bis er den Boden bei $s = 0$ erreicht hat? Welche Geschwindigkeit hat er beim Aufprall?
- Lösung: Wir haben also

$$s(t) = -\frac{1}{2} \cdot 10 \cdot t^2 + 1000 \text{ [m]} \text{ bzw. } s(t = 3\text{s}) = -\frac{1}{2} \cdot 10 \cdot 3^2 + 1000 = 955 \text{ m} \Rightarrow 45 \text{ m}$$

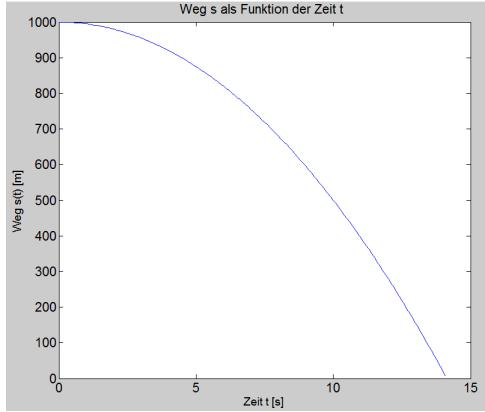
Beim Aufprall gilt

$$s(t) = -\frac{1}{2} \cdot 10 \cdot t^2 + 1000 = 0 \Rightarrow t = \sqrt{\frac{2 \cdot 1000}{10}} = 14.14 \text{ s}$$

und für die Geschwindigkeit haben wir

$$v(t) = -gt + v_0 = -10 \cdot 14.14 = -141.42 \frac{m}{s} = -509.12 \frac{km}{h}.$$

Das Minuszeichen bedeutet wieder, dass die Geschwindigkeit entgegen der Koordinatenachse, also auf die Erdoberfläche hin weist. Der hohe Wert der Geschwindigkeit überrascht und gilt nur unter Vernachlässigung des Luftwiderstandes. In Realität führt die Luftreibung dazu, dass die maximale Fallgeschwindigkeit tiefer als der hier errechnete Wert liegt (ungefähr bei der Hälfte, natürlich noch abhängig von der Fläche des Objekts). Die Fallparabel für diese Beispiele sieht folgendermassen aus:

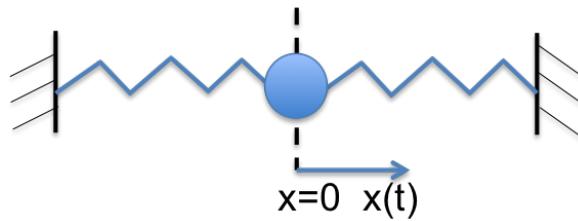


8.3.2 Harmonische Schwingung eines Federpendels (aus Papula)

Von Newton wissen wir: die Beschleunigung, die eine Masse m erfährt, ist proportional zur angreifenden Kraft. Außerdem wissen wir, die Beschleunigung ist die zweite Ableitung der Ortskoordinate, also

$$F = m \cdot a = m \cdot \ddot{x} \quad (\text{2. Newtonsches Gesetz}).$$

Wenn wir also die angreifende Kraft kennen, dann können wir dieses Gesetz dazu verwenden, die Ortskoordinate $x(t)$ als Funktion der Zeit zu bestimmen (wie wir es bereits beim Fall im luftleeren Raum getan haben). Nun wollen wir dies verwenden, um die Bewegungsgleichung eines ungedämpften Federpendels, also die Auslenkung $x(t)$ einer an einer Feder aufgehängten Masse m , zu berechnen:



Welche Kräfte müssen wir berücksichtigen? Wenn die Feder um x ausgelenkt wird, wirkt eine Rückstellkraft, die proportional zur Auslenkung ist (*Hooke'sches Gesetz*):

$$F_1 = -cx \quad (c: \text{Federkonstante}).$$

Unter Vernachlässigung der Gravitation und Reibung können wir also schreiben

$$m\ddot{x} = -cx.$$

Wir haben also wieder eine Differentialgleichung zweiter Ordnung. Anders als beim Fall im luftleeren Raum hängt nun die zweite Ableitung der Auslenkung nach der Zeit \ddot{x} nicht nur von einer Konstanten, sondern auch von x selber ab. Anschaulich bedeutet dies, dass je grösser die Auslenkung x ist, umso grösser ist die rückstellende Kraft bzw. die Beschleunigung (welche, wie durch das Minuszeichen ersichtlich ist, in entgegengesetzter Richtung wirkt). Natürlich ist dies nur für genügend kleine Auslenkungen der Fall. Überschreitet x eine kritische Grenze, wird die Feder überdehnt und das Hookesche Gesetz verliert seine Gültigkeit.

Um die obige Differentialgleichung zu lösen, benötigen wir also eine Funktion, die zweimal abgeleitet wieder sich selbst ergibt (bis auf eine Konstante und ein Vorzeichenwechsel). Wie sieht dieses $x(t)$ also aus? Aus der Analysis wissen wir, dass

$$\frac{d}{dt} \sin t = \cos t \text{ und } \frac{d}{dt} \cos t = -\sin t \text{ bzw. } \frac{d^2}{dt^2} \sin t = -\sin t$$

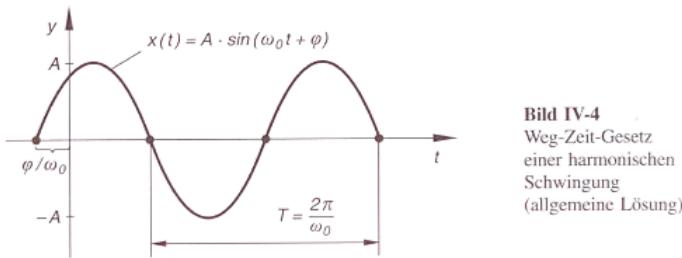
also ist $x(t)$ wahrscheinlich eine Sinus- bzw. Cosinus-Funktion. Tatsächlich, wenn wir den Ansatz machen

$$x(t) = A \cdot \sin(\omega_0 t + \varphi)$$

sehen wir durch einsetzen, dass diese Funktion die Differentialgleichung löst, wenn wir die sogenannte Kreisfrequenz $\omega_0 = \sqrt{\frac{c}{m}}$ setzen:

$$\begin{aligned}\dot{x}(t) &= A \cdot \omega_0 \cdot \cos(\omega_0 t + \varphi) \\ \ddot{x}(t) &= -A \cdot \omega_0^2 \cdot \sin(\omega_0 t + \varphi) = -\frac{c}{m} \cdot x(t).\end{aligned}$$

Dabei ist der Parameter A die Amplitude (Auslenkung) zum Zeitpunkt $t = 0$ (also $x(0) = A$) und der Parameter φ ist die sogenannte Phasenverschiebung. Die (allgemeine) Lösung sieht also folgendermassen aus:



8.4 Richtungsfelder für Differentialgleichungen 1. Ordnung

Differentialgleichungen 1. Ordnung und ihre Lösungen lassen sich mit Hilfe von sogenannten Richtungsfeldern veranschaulichen. Ausgangspunkt ist die geometrische Interpretation der Gleichung

$$y'(x) = f(x, y(x)).$$

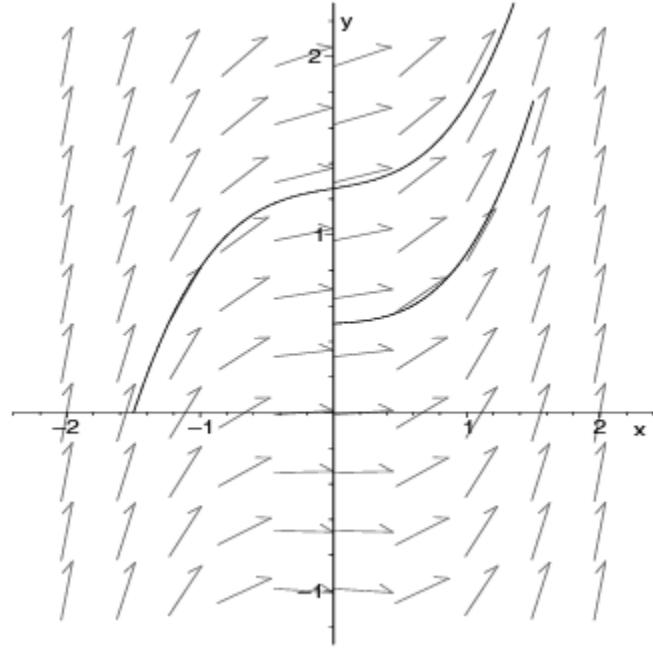
Offensichtlich gibt sie uns einen Zusammenhang zwischen der Steigung $y'(x)$ der gesuchten Funktion $y(x)$ für ein gegebenes x und dem Punkt $(x, y(x))$. In der (x, y) -Ebene bedeutet dies also: Wenn der Graph einer Lösung $y(x)$ durch einen Punkt (x_0, y_0) verläuft, so muss er dort die Steigung $y'(x_0)$ haben. Das heisst, wir können in der (x, y) -Ebene an einem beliebigen (x, y) -Punkt die Steigung $y'(x) = f(x, y(x))$ ausrechnen und können also die Tangente durch einen kleinen Pfeil graphisch anzeigen. Dann geben diese Pfeile in jedem Punkt die Richtung der Lösungskurve an. Auf diese Weise erhält man das Richtungsfeld der Differentialgleichung und die Lösungskurven der Gleichung verlaufen stets tangential zu den Pfeilen im Richtungsfeld.

Beispiel 8.3 [1]:

- Gegeben ist die Differentialgleichung

$$y'(x) = f(x, y(x)) = x^2 + 0.1 \cdot y(x).$$

Wir erhalten das Richtungsfeld, indem wir für jeden Punkt in der (x, y) -Ebene die zugehörige Steigung einzeichnen. Zum Beispiel haben wir für den Punkt $(-1, 1)$ die Steigung $y'(-1) = (-1)^2 + 0.1 \cdot 1 = 1.1$, oder für den Punkt $(0.5, 1)$ entsprechend $y'(0.5) = (0.5)^2 + 0.1 \cdot 1 = 0.35$. Die Lösungskurven ergeben sich, wenn man von einem gegebenen Anfangspunkt den Richtungspfeilen folgt. Gezeigt sind die Lösungen für die Anfangswerte $y(-1.5) = 0$ und $y(0) = 0.5$:



Aufgabe 8.1:

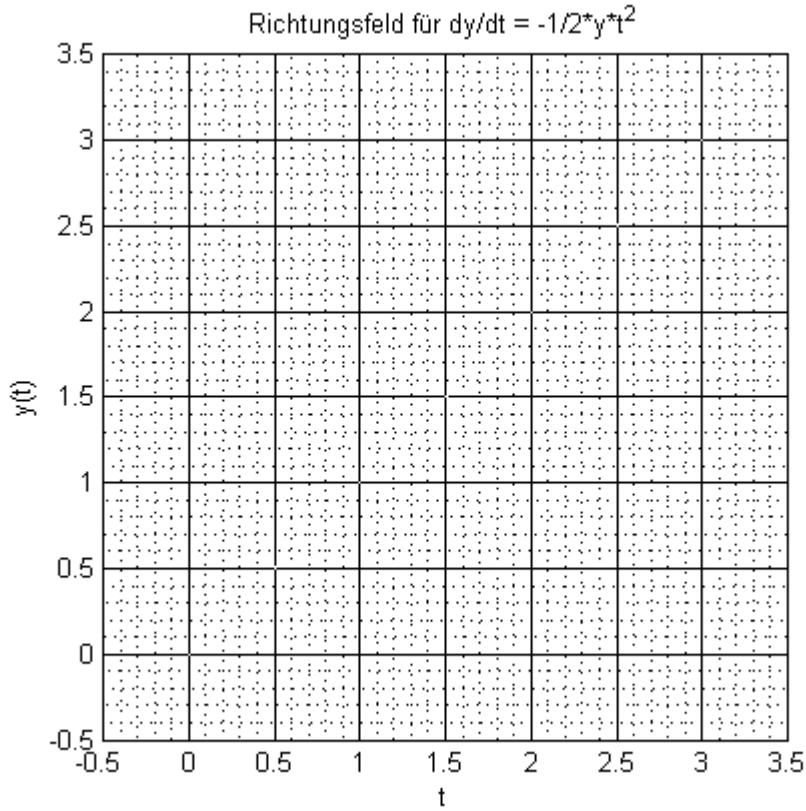
- Tragen Sie für die DGL

$$\frac{dy}{dt} = -\frac{1}{2} \cdot y \cdot t^2$$

die Steigung $\frac{dy}{dt}$ für die Punkte $(t_i, y_j) = (i, j)$ für $i, j = 0, 1, 2, 3$ in die untenstehende Tabelle ein und zeichnen Sie damit von Hand das entsprechende Richtungsfeld in untenstehendes Koordinatensystem. Zeichnen Sie anschliessend die (ungefähre) Lösungskurve für den Anfangswert $y(0) = 3$ ein.

- Lösung:

$\frac{dy}{dt}$	$t_0 = 0$	$t_1 = 1$	$t_2 = 2$	$t_3 = 3$
$y_0 = 0$				
$y_1 = 1$				
$y_2 = 2$				
$y_3 = 3$				



Die Idee vieler numerischer Verfahren zur Lösung von Anfangswertproblemen ist es nun, dem Richtungsfeld möglichst genau zu folgen. Allerdings besteht eine Lösungskurve aus unendlich vielen Punkten, die nicht alle berechnet werden können. Man benötigt also eine Diskretisierung, vergleichbar zur Diskretisierung bei der numerischen Integration.

Lösungsidee:

Gesucht ist die Funktion $y : x \mapsto y(x)$, $[a, b] \rightarrow \mathbb{R}$ mit der Anfangsbedingung $y(a) = y_0$ für die DGL 1. Ordnung $y'(x) = f(x, y(x))$. Wir wählen für ein vorgegebenes $n \in \mathbb{N}$ die (äquidistante) Schrittweite $h = \frac{b-a}{n}$ und diskretisieren das Intervall $[a, b]$ mit $n+1$ Gitterstellen $x_i = a + ih$, wobei $i = 0, \dots, n$. Ziel ist es jetzt also, für die Gitterstellen x_i Näherungen y_i für die exakte Lösung $y(x_i)$ des Anfangswertproblems zu berechnen. Es gibt dafür sogen. Einzelschritt- und Mehrschrittverfahren. In einem Einzelschrittverfahren wird die Näherung an der nächsten Stelle x_{i+1} basierend auf der bereits bekannten Näherung an der Stelle x_i berechnet gemäß (vgl. Abb. 8.1):

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + \text{Steigung} \cdot h \end{aligned}$$

Dabei ist der Wert für *Steigung* eine numerische Näherung für $\frac{dy}{dx}$ auf dem *gesamten* Intervall $[x_i, x_{i+1}]$. Der Unterschied der verschiedenen Einschrittverfahren, die wir kennen lernen werden, liegt in der Art und Weise, wie diese Steigung berechnet wird. Bei Mehrschrittverfahren wird die Näherung an der Stelle x_{i+1} basierend auf den bekannten Näherung an mehreren vorherigen Stellen berechnet. Auf Mehrschrittverfahren werden wir nur am Rand eingehen.

8.5 Das Euler-Verfahren

Das einfachste numerische Einschrittverfahren zur Lösung von Anfangswertproblemen ist das klassische Euler-Verfahren. Obwohl es in der Praxis mehrere Einschränkungen für die Anwendung dieses Verfahrens gibt und es deshalb keine grosse praktische Bedeutung hat, wollen wir es hier verwenden, um die grundlegenden Prinzipien zu demonstrieren. Verbesserte Verfahren sind das modifizierte Euler-Verfahren (manchmal auch Heun-Verfahren genannt) oder das Mittelpunkt-Verfahren.

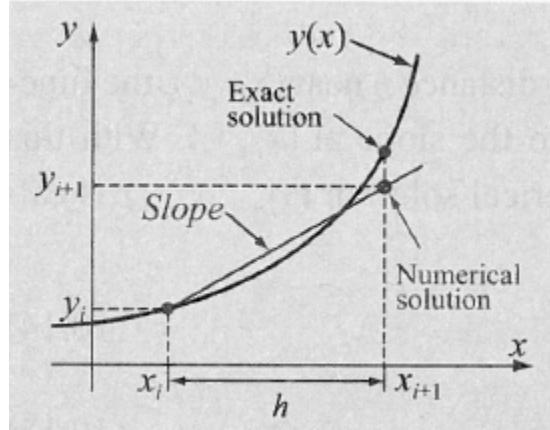


Abbildung 8.1: Einschrittverfahren zur Berechnung von y_{i+1} basierend auf y_i . Slope bedeutet Steigung (aus [9]).

8.5.1 Das klassische Euler-Verfahren

Wir gehen aus vom Anfangswertproblem

$$\frac{dy}{dx} = f(x, y(x)) \quad \text{mit } y(a) = y_0.$$

Das bedeutet also, $x_0 = a$ ist die einzige Stelle, bei der wir den Wert von $y(x)$ exakt kennen (da ja $y(x_0) = y_0$ vorgegeben ist) und auch die einzige Stelle, wo wir y' exakt kennen (wegen $y'(x_0) = f(x_0, y(x_0)) = f(x_0, y_0)$). Dem Startpunkt $(x_0, y(x_0))$ kommt also eine besondere Bedeutung zu. Alle weiteren berechneten Werte werden Fehler enthalten, da wir dem Richtungsfeld nicht exakt folgen können. Wir nehmen aber an, dass die Steigung in den Punkten um $(x_0, y(x_0))$ nur wenig von $y'(x_0)$ abweicht. Wir können dann die Steigung der Sekanten durch die Punkte $(x_0, y(x_0))$ und $(x_1, y(x_1))$ mit der Steigung $y'(x_0)$ gleichsetzen und dann nach $y(x_1)$ auflösen:

$$y'(x_0) = f(x_0, y(x_0)) \approx \frac{y(x_1) - y(x_0)}{\underbrace{x_1 - x_0}_h} \Rightarrow y(x_1) \approx y(x_0) + h \cdot f(x_0, y(x_0)) = y_1.$$

Wir folgen sozusagen der Tangente im Punkt $(x_0, y(x_0))$ um die Schrittweite h in der Hoffnung, dass wir dann genügend nahe an den Punkt $y(x_1)$ rankommen. Wenn die Schrittweite h genügend klein ist, sollte auch der Fehler entsprechend klein sein und umgekehrt (siehe Abb. 8.2).

Dieser Schritt lässt sich nun ausgehend von der Näherung $y_1 \approx y(x_1)$ wiederholen, wir erhalten dann $y(x_2) \approx y_1 + h \cdot f(x_1, y_1) = y_2$ usw. Dies ergibt das Euler-Verfahren:

Algorithmus: Euler-Verfahren [1]:

- Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \quad \text{mit } y(a) = y_0.$$

- Das Euler-Verfahren zur numerischen Lösung lautet

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot f(x_i, y_i) \end{aligned}$$

wobei $x_0 = a$, $x_i = a + ih$ für $i = 0, \dots, n - 1$ ($n \in \mathbb{N}$) und $h = \frac{b-a}{n}$.

Bemerkung:

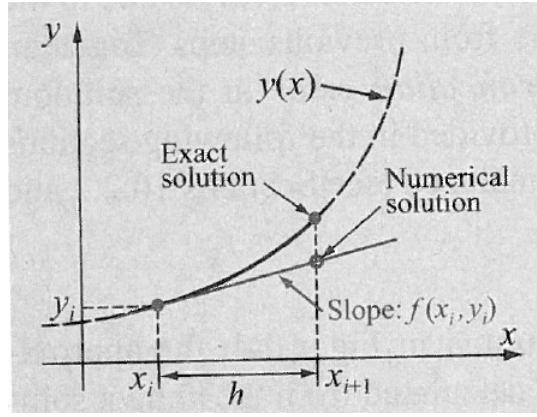


Abbildung 8.2: Euler-Verfahren zur Berechnung von y_{i+1} an der Stelle x_{i+1} basierend auf der Tangente mit der Steigung $f(x_i, y_i)$ an der Stelle x_i (aus [9]).

- Das Euler-Verfahren lässt sie sich auch mittels numerischer Integration herleiten, sofern man annimmt, dass $f(x, y(x))$ auf dem Intervall $[x_i, x_{i+1}]$ konstant ist (vgl. Kap. 7.2.2, Rechteckregel):

$$\begin{aligned}
 \frac{dy}{dx} &= f(x, y(x)) \\
 dy &= f(x, y(x)) dx \\
 \int_{y_i}^{y_{i+1}} dy &= \int_{x_i}^{x_{i+1}} f(x, y(x)) dx \\
 [y]_{y_i}^{y_{i+1}} &\approx f(x_i, y_i) \cdot (x_{i+1} - x_i) \\
 y_{i+1} - y_i &\approx f(x_i, y_i) \cdot h \\
 y_{i+1} &\approx y_i + h \cdot f(x_i, y_i)
 \end{aligned}$$

Aufgabe 8.2 [1]:

- Berechnen Sie mit dem Euler-Verfahren die numerische Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)) = t^2 + 0.1 \cdot y(t)$$

mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$.

- Zeichnen Sie Ihre Lösung mit Python in ein Koordinatensystem und vergleichen Sie mit der exakten Lösung

$$y(t) = -10t^2 - 200t - 2000 + 1722.5 \cdot e^{0.05(2t+3)}$$

- Offensichtlich weicht das Euler-Verfahren stark von der exakten Lösung ab. Woran liegt das?

8.5.2 Das Mittelpunkt-Verfahren

Das klassische Euler-Verfahren beruht darauf, die Steigung $y'(x_i)$ am Punkt (x_i, y_i) zu berechnen und der Tangente in diesem Punkt eine Schrittweite h zu folgen. Beim Mittelpunktverfahren folgt man der Tangente nur die halbe Schrittweite $h/2$ und berechnet beim Mittelpunkt des Intervalls $x_{h/2} = x_i + \frac{h}{2}$ die neue Steigung $y_{h/2}$:

$$y_{h/2} = y_i + \frac{h}{2} \cdot f(x_i, y_i)$$

Anschliessend geht man zurück an den Ausgangspunkt (x_i, y_i) und benutzt die berechnete Steigung $y_{h/2}$ für einen ganzen Schritt der Schrittweite h . Siehe Abb. 8.3

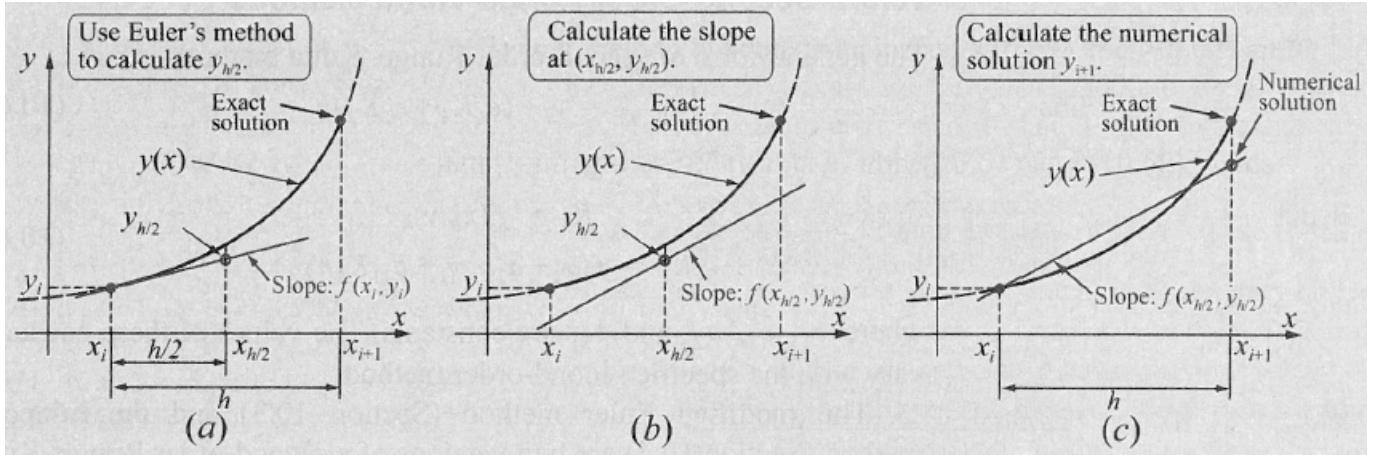


Abbildung 8.3: Mittelpunkt -Verfahren zur Berechnung von y_{i+1} an der Stelle x_{i+1} basierend auf der Tangente mit der Steigung $f(x_{h/2}, y_{h/2})$ an der Stelle $x_{h/2}$ (aus [9]).

Algorithmus: Mittelpunkt-Verfahren

- Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \quad \text{mit } y(a) = y_0.$$

- Das Mittelpunkt-Verfahren zur numerischen Lösung lautet

$$\begin{aligned} x_{h/2} &= x_i + \frac{h}{2} \\ y_{h/2} &= y_i + \frac{h}{2} \cdot f(x_i, y_i) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot f(x_{h/2}, y_{h/2}) \end{aligned}$$

wobei $x_0 = a$, $x_i = a + ih$ für $i = 0, \dots, n-1$ ($n \in \mathbb{N}$) und $h = \frac{b-a}{n}$.

Beispiel 8.4:

- Wir berechnen wie in Aufgabe 8.2 die numerische Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)) = t^2 + 0.1 \cdot y(t),$$

mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$, diesmal aber mit dem Mittelpunkt-Verfahren.

- Mittelpunkt-Verfahren:

$$\begin{aligned} n &= 5 \Rightarrow h = \frac{b-a}{n} = \frac{1.5 - (-1.5)}{5} = 0.6 \\ t_i &= a + ih = -1.5 + i \cdot 0.6 \quad (i = 0, 1, \dots, 5) \\ y_{i+1} &= y_i + h f(t_{h/2}, y_{h/2}) = y_i + h(t_{h/2}^2 + 0.1 y_{h/2}) \end{aligned}$$

i	t_i	y_i	$t_{h/2} = t_i + \frac{h}{2}$	$y_{h/2} = y_i + \frac{h}{2} \cdot (t_i^2 + 0.1y_i)$	$t_{i+1} = t_i + h$	$y_{i+1} = y_i + h \cdot (t_{h/2}^2 + 0.1y_{h/2})$
0	-1.5	0	-1.2	0.6750	-0.9	0.9045
1	-0.9	0.9045	-0.6	1.1746	-0.3	1.1910
2	-0.3	1.1910	0.0	1.2537	0.3	1.2662
3	0.3	1.2662	0.6	1.3312	0.9	1.5621
4	0.9	1.5621	1.2	1.8519	1.5	2.5372
5	1.5	2.5372	-	-	-	-

Aufgabe 8.3:

- Implementieren Sie das obige Beispiel 8.4 als Skript mit dem Mittelpunkt-Verfahren in Python und stellen Sie die Lösung grafisch dar (ohne Richtungsfeld).

8.5.3 Das modifizierte Euler-Verfahren

Beim modifizierten Euler-Verfahren (manchmal auch als Heun-Verfahren bezeichnet, wobei es hier in der Literatur Inkonsistenzen gibt) verwendet man den Durchschnitt zweier Steigungen. Zuerst folgt man wie beim klassischen Euler-Verfahren ausgehend vom Punkt (x_i, y_i) der Tangente der Steigung $f(x_i, y_i)$ einen ganzen Schritt und erhält den neuen Punkt $(x_{i+1}, y_{i+1}^{Euler})$ wobei $y_{i+1}^{Euler} = y_i + h \cdot f(x_i, y_i)$. Dies ist in Abb. 8.4 (a) dargestellt. Anschliessend berechnet man im Punkt $(x_{i+1}, y_{i+1}^{Euler})$ die nächste Steigung $f(x_{i+1}, y_{i+1}^{Euler})$, siehe Abb. 8.4 (b). Anschliessend wird der Durchschnitt der beiden Steigungen genommen und vom Ausgangspunkt (x_i, y_i) ein Schritt mit Schrittänge h gemacht zum neuen Punkt (x_{i+1}, y_{i+1}) , siehe Abb. 8.4 (c).

Algorithmus: Modifiziertes Euler-Verfahren

- Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \quad \text{mit } y(a) = y_0.$$

- Das modifizierte Euler-Verfahren zur numerischen Lösung lautet

(a) Führe das klassische Euler-Verfahren durch und speichere die erste Tangentensteigung in der Variable k_1 :

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1}^{Euler} &= y_i + h \cdot f(x_i, y_i) \\ k_1 &= f(x_i, y_i) \end{aligned}$$

(b) Berechne die zweite Tangentensteigung am Punkt $(x_{i+1}, y_{i+1}^{Euler})$ und speichere sie in der Variablen k_2 :

$$k_2 = f(x_{i+1}, y_{i+1}^{Euler})$$

(c) Bilde den Durchschnitt der beiden Steigungen $(k_1 + k_2)/2$ und mache einen Schritt h ausgehend vom ursprünglichen Punkt (x_i, y_i) zur Berechnung der Näherung (x_{i+1}, y_{i+1}) :

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot \frac{(k_1 + k_2)}{2} \end{aligned}$$

(d) Wiederhole diese Schritte ausgehend von $x_0 = a$ für $x_i = a + ih$ mit $i = 0, \dots, n - 1$ ($n \in \mathbb{N}$) und $h = \frac{b-a}{n}$.

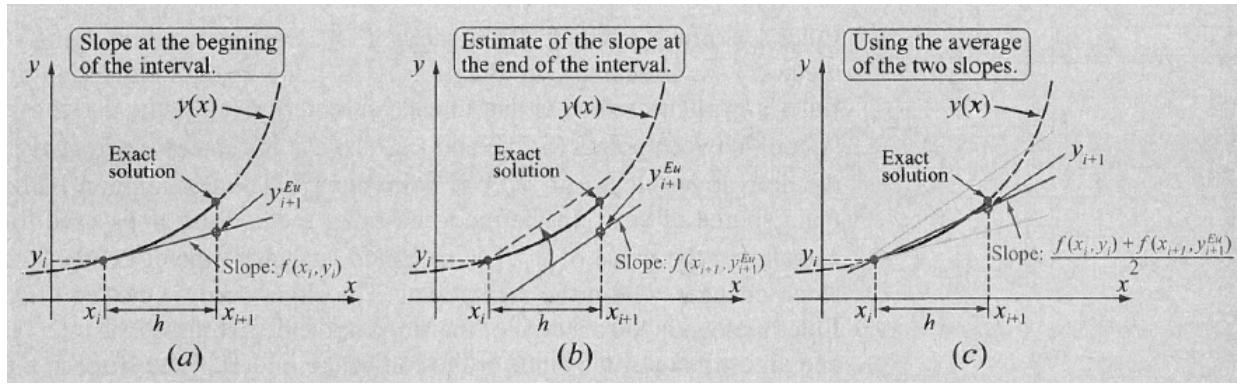


Abbildung 8.4: Modifiziertes Euler-Verfahren zur Berechnung von y_{i+1} an der Stelle x_{i+1} basierend auf dem Mittelwert der Steigungen $f(x_i, y_i)$ und $f(x_{i+1}, y_{i+1}^{Euler})$ an den Stellen x_i bzw. x_{i+1} (aus [9]).

Beispiel 8.5:

- Wir berechnen wieder wie in Aufgabe 8.2 und Beispiel 8.4 die numerische Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)) = t^2 + 0.1 \cdot y(t),$$

mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$, diesmal aber mit dem modifizierten Euler-Verfahren. Anschliessend vergleichen wir die Werte mit der exakten Lösung

$$y(t) = -10t^2 - 200t - 2000 + 1722.5 \cdot e^{0.05(2t+3)}$$

- modifiziertes Euler-Verfahren:

$$\begin{aligned} n &= 5 \Rightarrow h = \frac{b-a}{n} = \frac{1.5 - (-1.5)}{5} = 0.6 \\ t_i &= a + ih = -1.5 + i \cdot 0.6 \quad (i = 0, 1, \dots, 5) \\ y_{i+1} &= y_i + h \cdot \frac{(k_1 + k_2)}{2} \end{aligned}$$

i	t_i	y_i	$k_1 = f(t_i, y_i)$	$y_{i+1}^{Euler} = y_i + h \cdot f(t_i, y_i)$	$t_{i+1} = t_i + h$	$k_2 = f(t_{i+1}, y_{i+1}^{Euler})$	$y_{i+1} = y_i + h \cdot \frac{(k_1 + k_2)}{2}$
0	-1.5	0	2.2500	1.3500	-0.9	0.9450	0.9585
1	-0.9	0.9585	0.9059	1.5020	-0.3	0.2402	1.3023
2	-0.3	1.3023	0.2202	1.4344	0.3	0.23345	1.4384
3	0.3	1.4384	0.2338	1.5787	0.9	0.9678	1.7989
4	0.9	1.7989	0.9899	2.3929	1.5	2.4893	2.8426
5	1.5	2.8426	-	-	-	-	-

- Vergleich:

i	t	y_{exakt}	y_{euler}	$y_{\text{mittelpunkt}}$	$y_{\text{mod. euler}}$
0	-1.5	0	0	0	0
1	-0.9	0.9135	1.3500	0.9045	0.9585
2	-0.3	1.2133	1.9170	1.1910	1.3023
3	0.3	1.3069	2.0860	1.2662	1.4384
4	0.9	1.6267	2.2652	1.5621	1.7989
5	1.5	2.6318	2.8871	2.5372	2.8426

Aufgabe 8.4:

- Implementieren Sie das obige Beispiel 8.5 als Skript mit dem modifizierten Euler-Verfahren in Python und stellen Sie die Lösung grafisch dar (ohne Richtungsfeld).

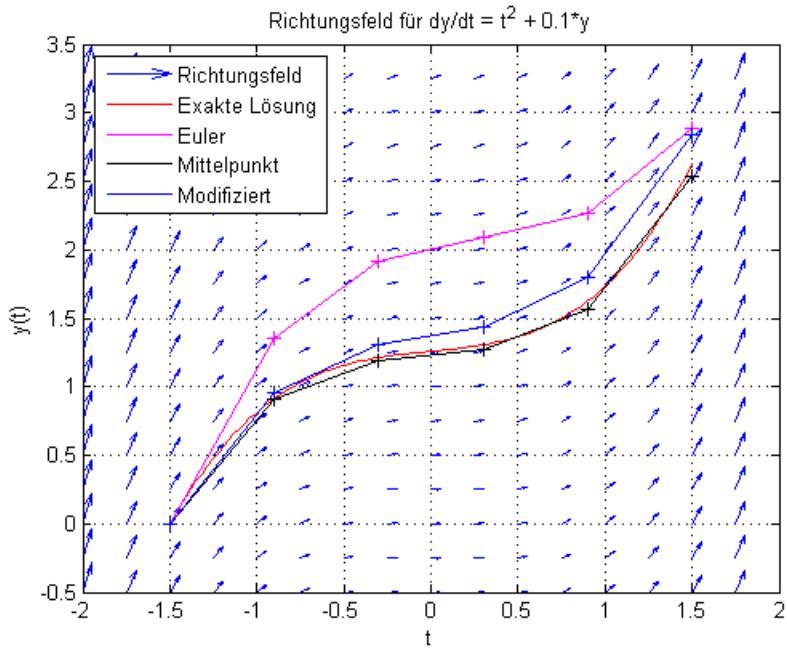


Abbildung 8.5: Die exakte Lösung der DGL $y'(t) = t^2 + 0.1 \cdot y(t)$ im Vergleich mit den Näherungen durch das Euler-Verfahren, das Mittelpunkt-Verfahren und das modifizierte Euler-Verfahren.

8.6 Die Fehlerordnung eines Verfahrens

In der obigen Aufgabe haben wir gesehen, dass sich die numerische Lösung weit von der exakten Lösung entfernen kann. Nun wollen wir dies etwas genauer untersuchen. Dafür müssen wir den Begriff des “lokalen” und “globalen” Fehlers sowie der jeweiligen Fehlerordnung definieren.

Definition 8.3 [1]: lokaler Fehler / Konsistenzordnung

- Sei $y'(x) = f(x, y(x))$ eine Differentialgleichung mit der Anfangsbedingung $y(x_i) = y_i$ und der exakten Lösung $y(x)$. Sei y_{i+1} der mit einem numerischen Näherungsverfahren mit der Schrittweite h berechnete Näherungswert für $y(x_{i+1})$, wobei $x_{i+1} = x_i + h$. Dann ist der **lokale Fehler** (also nach einer Iteration) definiert als die Differenz zwischen dem exakten Wert und der Näherung:

$$\varphi(x_i, h) := y(x_{i+1}) - y_{i+1}$$

- Ein numerisches Verfahren hat die **Konsistenzordnung p** falls gilt:

$$|\varphi(x_i, h)| \leq C \cdot h^{p+1}$$

für genügend kleine h und einer Konstante $C > 0$, die von der Differentialgleichung abhängt.

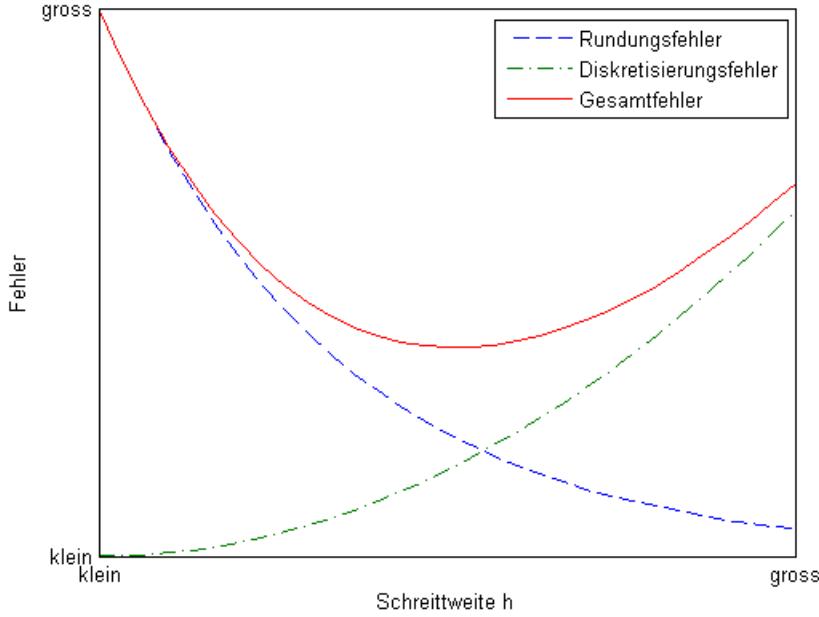


Abbildung 8.6: Verhalten des Rundungsfehlers und des Diskretisierungsfehlers.

Definition 8.4 [1]: globaler Fehler / Konvergenzordnung

- Sei $y'(x) = f(x, y(x))$ eine Differentialgleichung mit der Anfangsbedingung $y(x_0) = y_0$ und der exakten Lösung $y(x)$. Sei y_n der mit einem numerischen Näherungsverfahren mit der Schrittweite h berechnete Näherungswert für $y(x_n)$, wobei $x_n = x_0 + nh$. Dann ist der Gesamtfehler (also nach n Iterationen) bzw. der **globale Fehler** definiert als die Differenz zwischen dem exakten Wert und der Näherung:

$$y(x_n) - y_n$$

- Ein numerisches Verfahren hat die **Konvergenzordnung p** falls gilt:

$$|y(x_n) - y_n| \leq C \cdot h^p$$

für genügend kleine h und einer Konstante $C > 0$, die von der Differenzialgleichung abhängt.

Bemerkungen:

- Der lokale Fehler ist also ein Mass ist für die Abweichung von der exakten Lösung nach einer Iteration, während der globale Fehler die Abweichung von der exakten Lösung nach n Iterationen misst.
- Die hier gemachten Definitionen beziehen sich nur auf den Diskretisierungsfehler. Nicht berücksichtigt wird der Rundungsfehler, der für abnehmendes h hinzu kommt (siehe auch Abb. 8.6).
- Für die hier betrachteten Verfahren ist die Konsistenz- und Konvergenzordnung jeweils identisch.
- Verwendbar sind nur Verfahren mit der Konvergenzordnung $p \geq 1$, da dann der globale Fehler gegen Null strebt:

$$\lim_{h \rightarrow 0} |y(x_n) - y_n| \leq \lim_{h \rightarrow 0} C \cdot h^p = 0.$$

Das bedeutet, der Diskretisierungsfehler wird theoretisch beliebig klein, wenn die Schrittweite h beliebig klein wird. In der Praxis verhindern natürlich Rundungsfehler, dass eine beliebige Genauigkeit erzielt werden kann.

Beispiel 8.6:

- Für das Euler-Verfahren kann man mittels Taylorentwicklung zeigen, dass für den lokalen Fehler gilt:

$$\varphi(x_n, h) = \frac{h^2}{2} y''(z)$$

wobei $z \in [x_n, x_n + h]$ eine unbekannte Zwischenstelle im Intervall ist. Die Konsistenzordnung ist also $p = 1$.

- Für den globalen Fehler gilt:

$$|y(x_n) - y_n| \leq \frac{h}{2} \max_{x \in [x_0, x_n]} |y''(x)| \cdot \tilde{C}$$

wobei wir auf die Konstante \tilde{C} hier nicht weiter eingehen möchten. Sie lässt sich aber unter gewissen Bedingungen berechnen. Die Abschätzung zeigt, dass damit auch die Konvergenzordnung des Euler-Verfahrens 1 ist.

- Die Mittelpunktsregel und das modifizierte Euler-Verfahren haben die Konsistenz- und Konvergenzordnung $p = 2$ (ohne Beweis).

8.7 Runge-Kutta Verfahren

Wie bereits gesehen, wird in Einzelschrittverfahren die Steigung im Richtungsfeld verwendet gemäss (vgl. Abb. 8.1):

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + \text{Steigung} \cdot h \end{aligned}$$

Das klassische Euler-Verfahren verwendet einen Punkt, um die Steigung zu berechnen und hat die Konsistenz- und Konvergenzordnung $p = 1$. Das Mittelpunkt-Verfahren und das modifizierte Euler-Verfahren verwenden dazu zwei Punkte und haben die Konsistenz- und Konvergenzordnung $p = 2$. Offenbar kann durch die Hinzunahme von zusätzlichen Punkten zur Berechnung eines Durchschnitt-Werts für die Steigung die Genauigkeit eines Einschrittverfahrens verbessert werden.

8.7.1 Das klassische vierstufige Runge-Kutta Verfahren

Ein Beispiel ist das folgende Verfahren:

Algorithmus: klassisches vierstufiges Runge-Kutta Verfahren [1]

- Gegeben sei für $x \in [a, b]$ das Anfangswertproblem

$$y' = f(x, y) \quad \text{mit } y(a) = y_0.$$

- Das klassische Runge-Kutta zur numerischen Lösung lautet

$$\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right) \\ k_3 &= f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right) \\ k_4 &= f(x_i + h, y_i + hk_3) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

wobei $x_0 = a$, $x_i = a + ih$ für $i = 0, \dots, n - 1$ ($n \in \mathbb{N}$) und $h = \frac{b-a}{n}$.

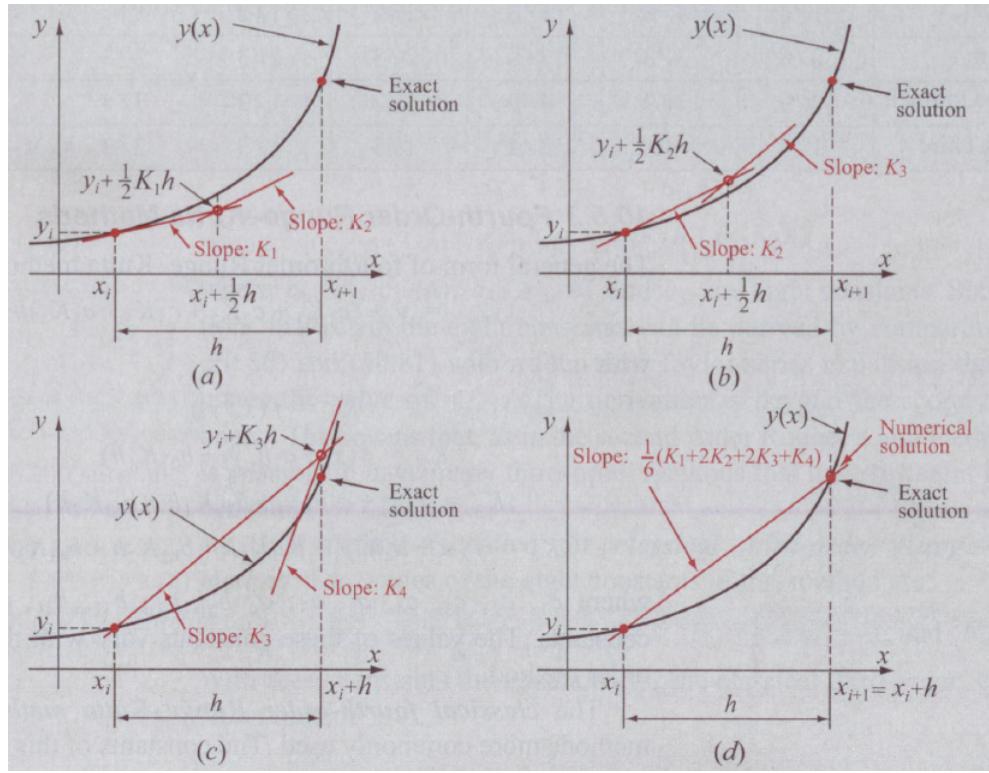


Abbildung 8.7: Klassisches vierstufiges Runge-Kutta-Verfahren zur Berechnung von y_{i+1} an der Stelle x_{i+1} basierend auf dem gewichteten Mittelwert aus den Steigungen k_1, k_2, k_3, k_4 . (aus [9]).

Bemerkung:

- Das klassische vierstufige Runge-Kutta Verfahren hat die Konsistenz- und Konvergenzordnung $p = 4$. Siehe dazu das nächste Beispiel.

Beispiel 8.7:

- Wir berechnen wieder wie in Beispiel 8.5 die numerische Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)) = t^2 + 0.1 \cdot y(t),$$

mit $y(-1.5) = 0$ auf dem Intervall $[a, b] = [-1.5, 1.5]$ und $n = 5$, diesmal aber mit klassischen Runge-Kutta Verfahren.

- Klassisches Runge-Kutta Verfahren:

i	t_i	y_{exakt}	y_i	k_1	k_2	k_3	k_4	y_{i+1}
0	-1.5	0	0	2.2500	1.5075	1.485	0.8991	0.9135
1	-0.9	0.9135	0.9135	0.9013	0.4784	0.4657	0.2093	1.2133
2	-0.3	1.2133	1.2133	0.2113	0.1277	0.1252	0.2188	1.3069
3	0.3	1.3069	1.3069	0.2207	0.4973	0.5056	0.9710	1.6267
4	0.9	1.6267	1.6267	0.9727	1.6318	0.1651	2.512	2.6318
5	1.5	2.6318	2.6318	-	-	-	-	-

- Der Vergleich mit den Werten der exakten Lösung y_{exakt} aus Bsp. 8.5 zeigt bei dieser Genauigkeit keinen Unterschied mehr.
- In Abb. 8.8 ist die Lösung grafisch gezeigt (linke Figur). Ebenfalls gezeigt ist die Abweichung $|y_{\text{exakt}}(t_i = 0) - y_i|$ für das fixierte $t_i = 0$ in Abhängigkeit von unterschiedlichen Schrittweite h (rechte Figur).

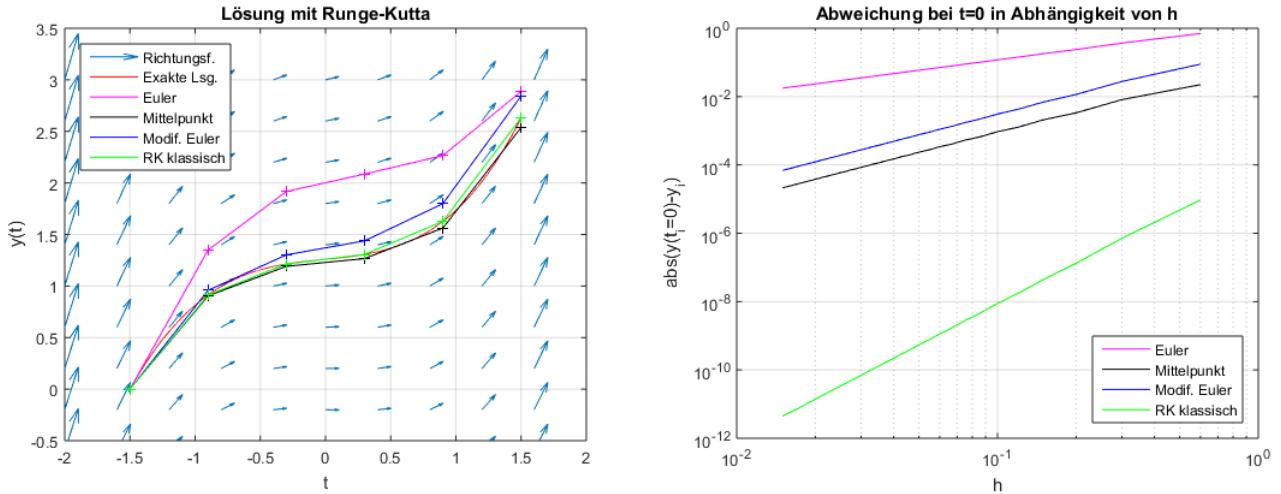


Abbildung 8.8: Lösung von Bsp. 8.7 für die verschiedenen Verfahren inkl. Runge-Kutta (links) sowie die grafische Darstellung der Konvergenzordnung (rechts).

Wir wissen bereits, dass das klassische Euler-Verfahren die Konvergenzordnung $p = 1$ hat, das Mittelpunkts- und das modifizierte Euler-Verfahren $p = 2$ und das klassische Runge-Kutta Verfahren $p = 4$. Im log-log Koordinatensystem (rechte Figur) entspricht das gerade der Steigung der entsprechenden Geraden. Der Achsenabschnitt (Ordinate) korreliert mit der Proportionalitätskonstante C der theoretischen Fehlerabschätzung aus Def. 8.4.

Aufgabe 8.5 [1]:

- a) Implementieren Sie das obige Beispiel 8.7 als Skript mit dem klassischen vierstufigen Runge-Kutta Verfahren in Python und stellen Sie die Lösung grafisch dar (ohne Richtungsfeld).
- b) Berechnen Sie für die Differentialgleichung $y'(x) = y(x)$ mit der Anfangsbedingung $y(0) = 1$ die Lösung auf dem Intervall $x \in [0, 1]$ mit dem klassischen vierstufigen Verfahren von Runge-Kutta und den Schrittweiten $h = 0.1$ und $h = 0.05$ in Python und vergleichen Sie die numerische Lösung mit dem exakten Wert.

8.7.2 Das allgemeine s -stufige Runge-Kutta Verfahren

Nun kann man das verallgemeinern und das s -stufige Runge-Kutta Verfahren definieren:

Definition 8.5 [1]: Allgemeines s -stufiges Runge-Kutta Verfahren

- Ein allgemeines (explizites) s -stufiges Runge-Kutta Verfahren ist gegeben durch die Formeln

$$\begin{aligned} k_n &= f\left(x_i + c_n h, y_i + h \sum_{m=1}^{n-1} a_{nm} k_m\right) \quad \text{für } n = 1, \dots, s \\ y_{i+1} &= y_i + h \sum_{n=1}^s b_n k_n \end{aligned}$$

- Hierbei ist $s \in \mathbb{N}$ die Stufenzahl und a_{nm}, b_n, c_n sind Konstanten. Die Konsistenz- und Konvergenzordnung hängt von der Wahl dieser Konstanten ab.

Bemerkung:

- Man notiert die Koeffizienten meist in der Form

c_1					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
\vdots					
c_n	a_{n1}	a_{n2}	\dots	$a_{n,n-1}$	
\vdots					
c_s	a_{s1}	a_{s2}	\dots	$a_{s,s-1}$	
	b_1	b_2	\dots	b_{s-1}	b_s

Beispiel 8.8:

- Alle bisher vorgestellten Verfahren entsprechen diesem Schema:

– Euler-Verfahren, $s = 1$:
$$\begin{array}{c|c} 0 \\ \hline 1 \end{array}$$

– Mittelpunkt-Verfahren, $s = 2$:
$$\begin{array}{cc|c} 0 & & 0.5 \\ 0.5 & & \\ \hline 0 & & 1 \end{array}$$

– Modifiziertes Euler-Verfahren, $s = 2$:
$$\begin{array}{c|c} 0 & 1 \\ 1 & \\ \hline 0.5 & 0.5 \end{array}$$

– klass. Runge-Kutta Verfahren, $s = 4$:
$$\begin{array}{cc|ccc} 0 & & 0.5 & & \\ 0.5 & & 0 & 0.5 & \\ 1 & & 0 & 0 & 1 \\ \hline & & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

8.8 Erweiterung auf Systeme von Differentialgleichungen

Die bisher vorgestellten Lösungsverfahren sind nur anwendbar auf Differentialgleichungen 1. Ordnung. Was aber, wenn eine Differentialgleichung höherer Ordnung gemäss Def. 8.2 vorliegt? Wir zeigen in diesem Abschnitt, (i) wie aus einer Differentialgleichung k -ter Ordnung ein System von k Differentialgleichungen 1. Ordnung gemacht und (ii) wie dieses System anschliessend mit den uns bereits bekannten Verfahren gelöst werden kann.

8.8.1 Zurückführen einer DGL k -ter Ordnung auf k DGL 1. Ordnung

Beginnen wir mit einem Beispiel:

Beispiel 8.9 [2]:

- Wir betrachten die Differentialgleichung 3. Ordnung

$$y''' + 5y'' + 8y' + 6y = 10e^{-x}$$

mit der Anfangsbedingung

$$y(0) = 2, \quad y'(0) = y''(0) = 0$$

– 1. Schritt: wir lösen nach der höchsten Ableitung auf:

$$y''' = 10e^{-x} - 5y'' - 8y' - 6y$$

– 2. Schritt: wir führen Hilfsfunktionen z_1, z_2, z_3 bis zur zweiten Ableitung ein:

$$\begin{aligned} z_1(x) &= y(x) \\ z_2(x) &= y'(x) \\ z_3(x) &= y''(x) \end{aligned}$$

- 3. Schritt: wir leiten die Hilfsfunktionen ab und setzen sie in $z'_3 = y'''$ ein

$$\begin{aligned} z'_1 &= y' (= z_2) \\ z'_2 &= y'' (= z_3) \\ z'_3 &= y''' \\ &= 10e^{-x} - 5y'' - 8y' - 6y \\ &= 10e^{-x} - 5z_3 - 8z_2 - 6z_1 \end{aligned}$$

- 4. Schritt: wir schreiben die DGL in vektorieller Form

$$\mathbf{z}' = \begin{pmatrix} z'_1 \\ z'_2 \\ z'_3 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ 10e^{-x} - 5z_3 - 8z_2 - 6z_1 \end{pmatrix} = \mathbf{f}(x, \mathbf{z}) \text{ und } \mathbf{z}(0) = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}$$

und erhalten so ein Anfangswertproblem 1. Ordnung (wobei Vektoren und vektorwertige Funktionen fett gedruckt sind):

$$\mathbf{z}' = \mathbf{f}(x, \mathbf{z}) \text{ mit } \mathbf{z}(0) = \begin{pmatrix} y(0) \\ y'(0) \\ y''(0) \end{pmatrix}$$

- Damit haben wir die DGL 3. Ordnung zurückgeführt auf 3 DGL 1. Ordnung und können die uns bekannten Verfahren zur Lösung anwenden.
- Bemerkung: da es sich um eine lineare DGL 3. Ordnung gehandelt hat, lässt sich \mathbf{z}' in diesem Beispiel als lineares Gleichungssystem schreiben

$$\mathbf{z}' = \mathbf{A}\mathbf{z} + \mathbf{b}$$

wobei

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -8 & -5 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ 10e^{-x} \end{pmatrix}$$

Dies ist aber nicht generell der Fall, deshalb schreiben wir \mathbf{z}' i.d.R. in der vektoriellen Form.

Rezept für das Zurückführen auf ein System erster Ordnung

1. Die Differentialgleichung nach der höchsten vorkommenden Ableitung der unbekannten Funktionen auflösen.
2. Neue Hilfsfunktionen für die unbekannte Funktion und deren Ableitungen bis Ordnung der höchsten Ableitung minus 1 einführen.
3. Das System erster Ordnung durch Ersetzen der höheren Ableitungen durch die neuen Funktionen aufstellen.
4. Das entsprechende Anfangswertproblem in vektorieller Form aufschreiben.

Aufgabe 8.6 [2]:

- Führen Sie die folgenden linearen Differentialgleichungen auf ein System erster Ordnung zurück:
 1. $y^{(4)} + 1.1y''' - 0.1y'' - 0.3y = \sin x + 5$ mit $y(0) = y''(0) = y'''(0) = 0$ und $y'(0) = 2$
 2. $x^2y'' + xy' + (x^2 - n^2)y = 0$ mit $y(1) = y'(1) = 2$

8.8.2 Lösen eines Systems von k DGL 1. Ordnung

Wie wir im Beispiel 8.9 gesehen haben, kann ein Anfangswertproblem vorliegen als System von DGL 1. Ordnung in der Form

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \text{ mit } \mathbf{y}(x_0) = \mathbf{y}^{(0)},$$

(wobei wie in den früheren Kapiteln ein hochgestellter Index in Klammern $\mathbf{y}^{(i)}$ einen Vektor aus \mathbb{R}^n nach der i -ten Iteration bezeichnet). Es wird also die Lösung $\mathbf{y}(x)$ gesucht, welche sich als vektorwertige Funktion $\mathbf{y} : \mathbb{R} \rightarrow \mathbb{R}^k$ interpretieren lässt mit

$$\mathbf{y} : x \mapsto \mathbf{y}(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_k(x) \end{pmatrix},$$

wobei die Komponenten $y_i(x)$ skalarwertige Funktionen $y_i : \mathbb{R} \rightarrow \mathbb{R}$ sind. Analog ist dann \mathbf{f} eine vektorwertige Funktion $\mathbf{f} : \mathbb{R} \times \mathbb{R}^k \rightarrow \mathbb{R}^k$ mit

$$\mathbf{f} : (x, \mathbf{y}(x)) \mapsto \mathbf{f}(x, \mathbf{y}(x)) = \begin{pmatrix} f_1(x, \mathbf{y}(x)) \\ f_2(x, \mathbf{y}(x)) \\ \vdots \\ f_k(x, \mathbf{y}(x)) \end{pmatrix}$$

und den Komponenten $f_i : \mathbb{R} \times \mathbb{R}^k \rightarrow \mathbb{R}$. In Komponentenschreibweise lautet das ganze System

$$\begin{aligned} y'_1(x) &= f_1(x, y_1(x), \dots, y_k(x)) \\ y'_2(x) &= f_2(x, y_1(x), \dots, y_k(x)) \\ &\vdots && \vdots \\ y'_k(x) &= f_k(x, y_1(x), \dots, y_k(x)) \end{aligned}$$

Rezept für das Lösen eines Systems von k DGL 1. Ordnung

Ist ein Lösungs-Verfahren

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + \text{Steigung} \cdot h \end{aligned}$$

für die eindimensionale Gleichung

$$y'(x) = f(x, y(x)), \quad y(x_0) = y_0$$

definiert, so kann es völlig analog erweitert werden als

$$\begin{aligned} x_i &= x_i + h \\ \mathbf{y}^{(i+1)} &= \mathbf{y}^{(i)} + \text{Steigung} \cdot h \end{aligned}$$

für ein System

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}(x)) \text{ mit } \mathbf{y}(x_0) = \mathbf{y}^{(0)},$$

(wobei wie üblich ein hochgestellter Index in Klammern $\mathbf{y}^{(i)}$ einen Vektor aus \mathbb{R}^n nach der i -ten Iteration bezeichnet).

Dabei werden ersetzt:

- y' durch den Vektor \mathbf{y}' der Ableitungen der einzelnen Komponenten,
- $f(x, y(x))$ durch die vektorwertige Funktion $\mathbf{f}(x, \mathbf{y}(x))$ und
- die skalare Anfangsbedingung $y(x_0) = y_0$ durch die Anfangsbedingung $\mathbf{y}(x_0) = \mathbf{y}^{(0)}$.

Es ist dann also

$$\mathbf{y}(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{pmatrix}, \mathbf{y}' = \begin{pmatrix} y'_1(x) \\ y'_2(x) \\ \vdots \\ y'_n(x) \end{pmatrix}, \mathbf{f}(x, \mathbf{y}(x)) = \begin{pmatrix} f_1(x, \mathbf{y}(x)) \\ f_2(x, \mathbf{y}(x)) \\ \vdots \\ f_n(x, \mathbf{y}(x)) \end{pmatrix}, \mathbf{y}(x_0) = \mathbf{y}^{(0)} = \begin{pmatrix} y_1(x_0) \\ y_2(x_0) \\ \vdots \\ y_n(x_0) \end{pmatrix}.$$

Bemerkung:

- Wurde das System erstellt, um eine DGL k -ter Ordnung zu lösen, so finden sich die gesuchte Lösung $y(x)$ in der ersten Komponente des Vektors $\mathbf{y}(x)$, also $y(x) \approx y_1(x)$.

Beispiel 8.10:

- Wir lösen das System aus Beispiel 8.9 mit dem Euler-Verfahren in vektorieller Form

$$\begin{aligned} x_{i+1} &= x_i + h \\ \mathbf{y}^{(i+1)} &= \mathbf{y}^{(i)} + h \cdot \mathbf{f}(x_i, \mathbf{y}^{(i)}) \end{aligned}$$

auf dem Intervall $[0, 1]$ und $h = 0.5$

- Das System lautet

$$\mathbf{z}' = \begin{pmatrix} z'_1 \\ z'_2 \\ z'_3 \end{pmatrix} = \begin{pmatrix} z_2 \\ z_3 \\ 10e^{-x} - 5z_3 - 8z_2 - 6z_1 \end{pmatrix} = \mathbf{f}(x, \mathbf{z}) \text{ und } \mathbf{z}(0) = \mathbf{z}^{(0)} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} z_1^{(0)} \\ z_2^{(0)} \\ z_3^{(0)} \end{pmatrix}$$

– $i = 0$: für den ersten Schritt erhalten wir

$$\begin{aligned} \mathbf{f}(x_0, \mathbf{z}^{(0)}) &= \begin{pmatrix} z_2^{(0)} \\ z_3^{(0)} \\ 10e^{-x_0} - 5z_3^{(0)} - 8z_2^{(0)} - 6z_1^{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 10e^{-0} - 5 \cdot 0 - 8 \cdot 0 - 6 \cdot 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix} \\ \mathbf{z}^{(1)} &= \mathbf{z}^{(0)} + h \cdot \mathbf{f}(x_0, \mathbf{z}^{(0)}) = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} 0 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} \\ x_1 &= x_0 + h = 0.5 \end{aligned}$$

– $i = 1$: für den zweiten Schritt erhalten wir

$$\begin{aligned} \mathbf{f}(x_1, \mathbf{z}^{(1)}) &= \begin{pmatrix} z_2^{(1)} \\ z_3^{(1)} \\ 10e^{-x_1} - 5z_3^{(1)} - 8z_2^{(1)} - 6z_1^{(1)} \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 10e^{-0.5} - 5 \cdot (-1) - 8 \cdot 0 - 6 \cdot 2 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ -0.9347 \end{pmatrix} \\ \mathbf{z}^{(2)} &= \mathbf{z}^{(1)} + h \cdot \mathbf{f}(x_1, \mathbf{z}^{(1)}) = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} 0 \\ -1 \\ -0.9347 \end{pmatrix} = \begin{pmatrix} 2 \\ -0.5 \\ -1.4673 \end{pmatrix} \\ x_2 &= x_1 + h = 1.0 \end{aligned}$$

Aufgabe 8.7:

- Benutzen Sie die Gleichungssysteme der Aufgabe 8.6 und berechnen Sie manuell je den ersten Schritt des Euler-Verfahrens und des klassischen vierstufigen Verfahrens von Runge-Kutta mit $h = 0.1$.

8.9 Stabilität

Wie wir bereits gesehen haben, setzt sich der Fehler bei der Lösung einer DGL zusammen aus dem vom benutzten Verfahren abhängigen Diskretisierungsfehler und dem vom Rechner abhängigen Rundungsfehler. Auf die Begriffe der Konsistenz bzw. Konvergenz sind wir in Kap. 8.6 eingegangen.

In gewissen Situationen kann es vorkommen, dass der numerische Fehler im Verlauf der Iteration unbeschränkt gross werden kann, unabhängig von der Schrittweite h . Man spricht dann von Instabilität bzw. einer instabilen Lösung. Die Stabilität einer Lösung hängt von drei Faktoren ab: (i) dem benutzten Verfahren, (ii) der Schrittweite h und (iii) dem spezifischen Anfangswertproblem. Es gibt verschiedene Methoden, um das Problem zu untersuchen, z.B.

- Durchrechnen der Lösung einmal mit single- und einmal mit double-precision mit anschliessendem Vergleich. Dies kann Aussagen möglich machen zur Akkumulation von Rundungsfehlern.
- Variation der Schrittweite h . Dies erlaubt Aussagen zum Diskretisierungsfehler.
- Vergleich der Resultate erzielt mit einem Lösungsverfahren höherer Ordnung und einem Lösungsverfahren niederer Ordnung.
- Vergleich der Resultate eines numerischen Lösungsverfahrens mit der analytischen Lösung, sofern diese bekannt ist.

Wir wollen den letzten Punkt etwas ausführen und betrachten das Anfangswertproblem

$$y' = -\alpha y, \quad y(0) = y_0 = 1 \quad (\alpha > 0)$$

für welches wir die exakte Lösung kennen:

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

Das Euler-Verfahren liefert die numerische Lösung

$$y_{i+1} = y_i - h \cdot \alpha y_i = y_i (1 - h\alpha).$$

Durch rekursives Einsetzen von $y_i = y_{i-1} (1 - h\alpha)$ erhalten wir

$$y_{i+1} = y_i \cdot (1 - h\alpha) = y_{i-1} (1 - h\alpha)^2 = y_{i-2} (1 - h\alpha)^3 = \dots = \underbrace{y_0}_{=1} (1 - h\alpha)^{i+1}.$$

Da die exakte Lösung $y = e^{-\alpha x}$ streng monoton fallend ist, sollte auch die Näherungslösung streng monoton fallend sein, d.h. in diesem Fall

$$|1 - h\alpha| < 1$$

und es folgt

$$0 < h\alpha < 2 \Rightarrow 0 < h < \frac{2}{\alpha}.$$

Wir haben also eine obere Schranke für h hergeleitet bzw. eine Schrittweitenobergrenze, für die die numerische Lösung stabil bleibt.

Definition 8.6: Stabilitätsfunktion / Stabilitätsintervall

- Kann bei der Anwendung eines Verfahrens auf die DGL $y' = -\alpha y$ die numerische Lösung in der Form

$$y_{i+1} = g(h\alpha) \cdot y_i$$

geschrieben werden, so nennt man $g(z)$ die Stabilitätsfunktion des Verfahrens (mit $z = h\alpha$).

- Das offene Intervall $z \in (0, \alpha)$, in dem $|g(z)| < 1$ gilt, bezeichnet man als das Stabilitätsintervall des Verfahrens.

Bemerkungen:

- Wie wir oben gesehen haben, ist die Stabilitätsfunktion für das Euler-verfahren $g(z) = 1 + z$
- Man kann zeigen, dass die Stabilitätsfunktion eine s -stufigen (expliziten) Runge-Kutta Verfahrens ein Polynom vom Grad s ist.

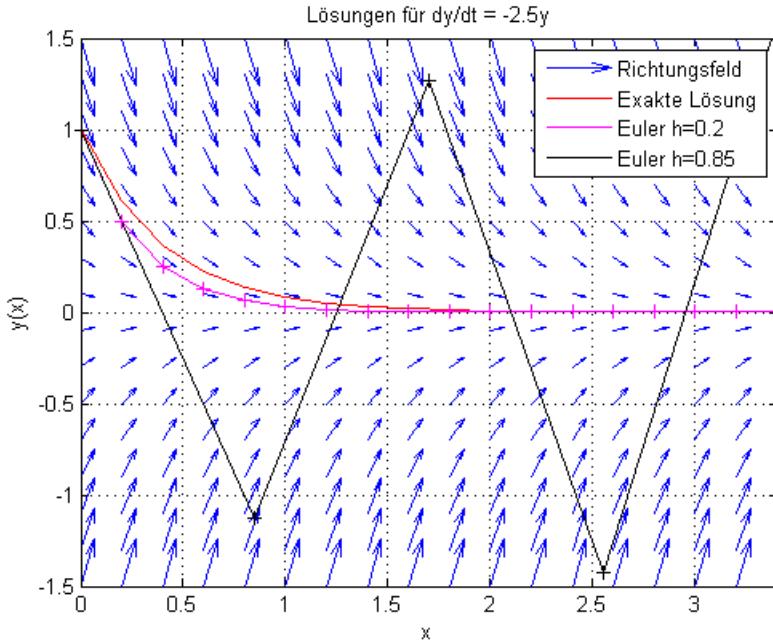


Abbildung 8.9: Euler-Verfahren für die Lösung der DGL $y' = -2.5y$ mit den Schrittweiten $h_1 = 0.2$ (stabil) und $h_2 = 0.85$ (instabil).

Beispiel 8.11 [9]:

- Wir betrachten die DGL

$$y' = -2.5y, \quad y(0) = 1, \quad x \in [0, 3.4].$$

Mit $\alpha = 2.5$ gilt $\frac{2}{\alpha} = 0.8$ und damit für eine stabile Lösung

$$0 < h < 0.8.$$

In Abb. 8.9 ist die stabile Lösung für $h_1 = 0.2$ gezeigt im Vergleich zur instabilen Lösung mit $h_2 = 0.85$.

8.10 Weitere Punkte

Wir haben in diesem Kapitel einige wichtige Verfahren zur Lösung von Anfangswertproblemen für gewöhnliche Differentialgleichungen kennengelernt und konnten doch nur an der Oberfläche der ganzen Thematik kratzen. Im Folgenden gehen wir zur Vollständigkeit noch kurz auf Mehrschrittverfahren und implizite Verfahren ein, erläutern die Begriffe der Schrittweitensteuerung und Steifigkeit von DGL und schliessen das Kapitel mit einer kurzen Übersicht über die in Python bereits implementierten Funktionen zur Lösung von Anfangswertproblemen ab.

8.10.1 Einschritt- vs. Mehrschrittverfahren

Wir haben bisher nur Einschrittverfahren kennengelernt, d.h. zur Berechnung von y_{i+1} wird nur der vorhergehende Punkt (x_i, y_i) benötigt. Bei Mehrschrittverfahren werden im Gegensatz dazu für die Berechnung von y_{i+1} nicht nur (x_i, y_i) sondern noch weiter zurückliegende Punkte wie $(x_{i-1}, y_{i-1}), (x_{i-2}, y_{i-2})$ etc. benötigt. Ein Beispiel aus der Klasse der Mehrschrittverfahren sind die Adams-Bashforth Methoden. Die Adams-Bashforth Methode 3. Ordnung lautet z.B.

$$y_{i+1} = y_i + \frac{h}{12} (23f(x_i, y_i) - 16f(x_{i-1}, y_{i-1}) + 5f(x_{i-2}, y_{i-2})).$$

Mehrschrittverfahren können effizienter sein bei der Berechnung der numerischen Lösung einer DGL.

8.10.2 Implizite vs. explizite Verfahren

Wir haben uns in diesem Kapitel auf die Lösung von expliziten DGL der Form

$$y^{(n)}(x) = f(x, y(x), y'(x), \dots, y^{(n-1)}(x))$$

beschränkt, d.h. die DGL ist nach der höchsten Ableitung aufgelöst. Wir haben demnach auch nur explizite Verfahren angewendet. Im Gegensatz dazu gibt es die impliziten DGL der Form

$$F(x, y(x), y'(x), \dots, y^{(n-1)}(x), y^{(n)}(x)) = 0,$$

die nicht nach der höchsten Ableitung aufgelöst sind. Bei den zugehörigen impliziten Lösungsverfahren muss jeweils noch ein Nullstellenproblem gelöst werden (z.B. mit dem Newton-Verfahren), wie beim impliziten Euler-Verfahren

$$\begin{aligned} x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + h \cdot f(x_{i+1}, y_{i+1}) \end{aligned}$$

da hier die gesuchte Lösung y_{i+1} auf beiden Seiten der Gleichung auftaucht. Diese Verfahren sind deshalb aufwendiger, aber auch stabiler als die expliziten Verfahren.

8.10.3 Steife DGL

Im Zusammenhang mit Stabilitätsüberlegungen (vgl. Kap. 8.10) redet man auch von sogen. 'steifen' DGL (stiff ODE). Diese sind DGL mit stark schwankenden Zeitskalen t (oder Längenskalen x) der Lösung y . Zur Lösung solcher steifen DGL benötigen explizite Lösungsverfahren eine verschwindend kleine Schrittweite h , so dass die Lösung wegen langer Laufzeiten unpraktikabel oder wegen zunehmenden Rundungsfehlern instabil wird. Steife DGL werden deshalb mit impliziten Verfahren gelöst.

8.10.4 Schrittweitensteuerung

Für DGL mit stark schwankenden Lösungsfunktionen $y(x)$ ist es effizienter, statt einer konstanten (ausreichend kleinen) Schrittweite $h = \frac{b-a}{n} = const.$ eine Variable Schrittweite h_i anzuwenden, die klein ist in Bereichen grosser Krümmung der Lösung $y(x)$ und gross in Bereichen mit kleiner Krümmung. Hierzu sind Fehlerschätzungen für den lokalen Fehler nötig. Auf die dazu benötigten Verfahren gehen wir nicht ein.

8.10.5 Python-Funktionen zur Lösung von Anfangswertproblemen

Python bietet mehrere Funktionen an, um Anfangswertprobleme zu lösen. Eine häufig benutzte ist

- `scipy.integrate.solve_ivp()` (siehe Dokumentation).

Die zu verwendenden Methoden (z.B. Runge-Kutta) werden als Parameter spezifiziert.

method	Description
'RK45'	Default. Explicit Runge-Kutta method of order 5(4). The error is controlled assuming accuracy of the fourth-order method, but steps are taken using the fifth-order accurate formula (local extrapolation is done).
'RK23'	Explicit Runge-Kutta method of order 3(2). The error is controlled assuming accuracy of the second-order method, but steps are taken using the third-order accurate formula (local extrapolation is done).
'DOP853'	Explicit Runge-Kutta method of order 8. Python implementation of the "DOP853" algorithm originally written in Fortran.
'Radau'	Implicit Runge-Kutta method of the Radau IIA family of order 5 [4]. The error is controlled with a third-order accurate embedded formula.
'BDF'	Implicit multi-step variable-order (1 to 5) method based on a backward differentiation formula for the derivative approximation.
'LSODA'	Adams/BDF method with automatic stiffness detection and switching.