
Mathematisches Seminar
Numerik

Leitung: Andreas Müller

Benjamin Bouhafs-Keller, Daniel Bucher, Manuel Cattaneo
Patrick Elsener, Reto Fritzsche, Niccolò Galliani, Tobias Grab
Thomas Kistler, Fabio Marti, Joël Rechsteiner, Cédric Renda
Michael Schmid, Mike Schmid, Michael Schneeberger
Martin Stypinski, Manuel Tischhauser, Nicolas Tobler
Raphael Unterer, Severin Weiss

Inhaltsverzeichnis

I Grundlagen	3
Einleitung	5
1 Berechnung	7
1.1 Zahlensysteme	8
1.1.1 Zahlendarstellung bezüglich verschiedener Basen	8
1.1.2 Festkommazahlen	10
1.1.3 Gleitkommazahlen	12
1.1.4 Hochpräzisionsbibliotheken	15
1.2 Numerische Effekte	20
1.2.1 Auslöschnung	20
1.2.2 Verschmierung	22
1.3 Iteration	26
1.3.1 Beispiele	26
1.3.2 Graphische Analyse	30
1.3.3 Konvergenzbedingung	30
1.3.4 Die logistische Gleichung	32
1.3.5 Dritte Ableitung im Fixpunkt	33
1.4 Konvergenzgeschwindigkeit	35
1.4.1 Lineare und quadratische Konvergenz	35
1.4.2 Konvergenzbeschleunigung	37
1.5 Numerische Instabilität	40
1.5.1 Eine instabile Quadratwurzel	40
1.5.2 Numerische Instabilität	41
1.6 Kondition	43
2 Gleichungen lösen	47
2.1 Nullstellen von Funktionen	48
2.1.1 Intervallhalbierung	48
2.1.2 Sekantenverfahren	51
2.2 Newton-Verfahren	53
2.2.1 Analytischer Ansatz für ein quadratisch konvergentes Verfahren	54
2.2.2 Geometrische Interpretation des Newton-Verfahrens	54
2.2.3 Wurzeln	55
2.2.4 Newton-Verfahren in \mathbb{R}^n	55
2.2.5 Der Fall $f'(x^*) = 0$	57

2.2.6	Vergleich mit dem Sekantenverfahren	58
2.2.7	Nullstellen von Polynomen	58
2.2.8	Inverse der Normalverteilungsfunktion	62
2.3	Homotopie-Verfahren	63
3	Interpolation	69
3.1	Lineare Interpolation und Polygonzüge	69
3.1.1	Lineare Interpolation	69
3.1.2	Polygonzüge	70
3.2	Interpolationspolynom	72
3.2.1	Bestimmung des Interpolationspolynoms	72
3.2.2	Fehler von Approximationenpolynomen	75
3.2.3	Runges Phänomen	84
3.2.4	Wahl der Stützstellen und Tschebyscheff-Interpolationspolynom	84
3.3	Hermite-Interpolation	89
3.3.1	Aufgabenstellung	89
3.3.2	Bestimmung des Hermite-Interpolationspolynom	92
3.3.3	Zwei Stützstellen	93
3.4	Baryzentrische Formeln für Interpolationspolynome	96
3.5	Spline-Interpolation	97
3.5.1	Anforderungen an die interpolierende Funktion	98
3.5.2	Das Optimierungsproblem	98
3.5.3	Lösung des Optimierungsproblems	102
3.5.4	Bézier-Kurven und Splines in der Ebene	105
4	Integration	119
4.1	Riemann-Integral und Trapezregel	119
4.1.1	Das Riemann-Integral	120
4.1.2	Mittelpunktsregel	121
4.1.3	Trapezregel	122
4.1.4	Fehler von Trapez- und Mittelpunktsregel	125
4.2	Romberg-Algorithmus	132
4.3	Weitere Integrationsverfahren	135
5	Gewöhnliche Differentialgleichungen	139
5.1	Problemstellung	139
5.1.1	Reduktion der Ordnung	140
5.1.2	Anfangswertprobleme	141
5.1.3	Randwertprobleme	142
5.1.4	Höhere Ableitungen	142
5.1.5	Ableitung nach der Anfangsbedingung	143
5.1.6	Numerische Lösung der Differentialgleichung für $y(x)$ und $J(x)$	145
5.1.7	Abhängigkeit von Parametern	146
5.2	Grundprinzip numerischer Lösungsverfahren	147
5.3	Fehler-Entwicklung numerischer Lösungen	150
5.4	Einschrittverfahren	151
5.4.1	Quadratische Verfahren	152
5.4.2	Runge-Kutta-Verfahren	154

5.5	Mehrschrittverfahren	156
5.5.1	Ein Verfahren zweiter Ordnung	157
5.5.2	Vergleich mit der Taylor-Reihe	158
5.5.3	Höhere Ordnung	158
5.5.4	Fehleranalyse für das Adams-Bashford-Verfahren zweiter Ordnung	158
5.6	Software	160
5.6.1	Octave	161
5.6.2	GNU Scientific Library	161
5.7	Randwertprobleme	163
5.7.1	Einführende Beispiele	163
5.7.2	Schiessverfahren	169
6	Lineare Gleichungssysteme	179
6.1	Direkte Verfahren	180
6.1.1	Thomas-Algorithmus für tridiagonale Matrizen	180
6.1.2	Zyklisch tridiagonale Matrix	183
6.1.3	Sherman-Morrison-Woodbury-Formel	183
6.1.4	Bandmatrizen	185
6.2	Iterative Gleichungslösung	186
6.2.1	Iterative Lösung nach Gauss-Seidel	186
6.2.2	Matrixformulierung	187
6.2.3	Konvergenzbedingung	188
6.2.4	Zerlegungsverfahren	189
6.3	Kondition	192
6.3.1	Gelfand-Radius und Eigenwerte	193
6.3.2	Konditionszahl	196
6.4	QR-Zerlegung mit Spiegelungen	197
6.4.1	Gram-Schmidt-Orthonormalisierung	198
6.4.2	Spiegelungen	198
6.4.3	QR-Zerlegung mit Spiegelungen	199
6.5	Diagonalisierung mit dem Jacobi-Verfahren	201
6.5.1	Jacobi-Verfahren in zwei Dimensionen	201
6.5.2	Beliebige Dimension	202
7	Partielle Differentialgleichungen	207
7.1	Problemstellung	207
7.1.1	Gebiet und Rand	208
7.1.2	Klassifikation der partiellen Differentialgleichungen	209
7.1.3	Randbedingungen	213
7.1.4	Lösungen	214
7.2	Finite Differenzen	214
7.2.1	Gitter und Ableitungen	215
7.2.2	Wärmeleitungsgleichung	221
7.2.3	Wärmeleitungsgleichung mit Dirichlet-Randbedingungen	227
7.2.4	Wärmeleitungsgleichung mit Neumann-Randbedingungen	230
7.2.5	Stabilität und computational mode	234
7.3	Finite Volumina	238

7.3.1	Die Kontinuitätsgleichung	239
7.3.2	Integralform	240
7.3.3	Diskretisation	241
7.4	Finite Elemente	242
7.4.1	Das äquivalente Minimalproblem	242
7.4.2	Approximation	247
7.4.3	Quadratische Minimalprobleme	252
II	Anwendungen und weiterführende Themen	261
8	Iteration der logistischen Gleichung	265
8.1	Modellierung von Populationen	265
8.2	Iteration der logistischen Gleichung	266
8.3	Stabilität der logistischen Gleichung	269
8.4	Beispiele von verwandten Funktionen	271
8.4.1	Mandelbrotmenge	271
8.4.2	Universelle Eigenschaft	273
9	Stabile Berechnung von Legendre-Polynomen	275
9.1	Einleitung	275
9.2	Problemstellung	276
9.3	Lösung	277
9.3.1	Anfangswerte	277
9.3.2	m -Rekursion	279
9.3.3	l -Rekursion	280
9.3.4	Welche Rekursionsformel soll nun verwendet werden?	280
9.4	Folgerungen	280
10	Gauss-Quadratur	283
10.1	Einleitung	283
10.2	Problemstellung	283
10.2.1	Überlegung hinter der Gauss-Quadratur	283
10.2.2	Anwendungsbereiche der numerischen Integration	285
10.3	Lösung	286
10.3.1	Anwendung der Gauss-Legendre-Formel	286
10.3.2	Berechnung der Position der Stützstellen	287
10.3.3	Orthogonale Polynome	291
10.3.4	Berechnung der Gewichte an den Stützstellen	293
10.3.5	Formen der Gauss-Quadratur	295
10.3.6	Fehler der Gauss-Quadratur	296
10.4	Folgerungen	299
10.4.1	Vergleich Gauss-Quadratur mit Trapezregel	299
10.4.2	Programm für Gauss-Quadratur in Python	299
10.4.3	Fazit	300

11 Van der Pol-Differentialgleichung	301
11.1 Einleitung	301
11.2 Die Van der Pol Gleichung	301
11.2.1 Van der Pol Gleichung als Oszillator mit nichtlinearem Widerstand	301
11.2.2 Homogene Gleichung	303
11.2.3 Inhomogene Gleichung	304
11.3 Numerische Lösung der Van der Pol Gleichung	304
11.3.1 Implementation des Runge-Kutta-Verfahrens	305
11.3.2 Implementation der Differentialgleichung	305
11.4 Chaos im Van der Pol System	306
11.4.1 Anfangsbedingung	308
11.4.2 Ergebnisse	310
11.5 Folgerungen	311
12 Taylor-Reihe und Differentialgleichungen	315
12.1 Einleitung	315
12.2 Problemstellung	315
12.3 Das Taylor-Verfahren	316
12.3.1 Auswertung des Vergleiches	317
12.4 Probleme mit der logistischen Funktion	319
12.4.1 Konvergenz	319
12.4.2 Gefährliche Umgebung der Anfangsbedingung	319
12.5 Lösung	320
12.6 Folgerungen aus dem Beispiel	320
12.7 Analytischer Vergleich der Inkrementfunktionen	320
13 Schrittängensteuerung	323
13.1 Einleitung	323
13.2 Problemstellung	323
13.3 Realisierung	324
13.3.1 Simple Schrittweitensteuerung mit konstanter Testschrittweite	324
13.3.2 Schrittweitensteuerung nach Fehlberg	326
13.4 Folgerungen	329
14 Numerische Laplace-Inversion	331
14.1 Motivation: Lösung einer linearen Differentialgleichung	332
14.2 Lösung	333
14.3 Folgerungen	335
14.4 Programmcode	336
15 Störungstheorie	343
15.1 Einleitung	343
15.2 Problemstellung	343
15.3 Numerische Lösung	344
15.4 Erster Lösungsansatz	345
15.5 Verbesserte Lösung	348
15.6 Möglichkeiten zur Genauigkeitssteigerung	348
15.6.1 Reduktion der Intervalllänge	349

15.6.2 Erhöhung der Ordnung	349
15.7 Fazit	353
16 Störungstheorie für das Eigenwertproblem	355
16.1 Einleitung	355
16.2 Problemstellung	356
16.3 Anwendungen	356
16.4 Idee	356
16.5 Herleitung	357
16.5.1 Nicht entarteter Fall	358
16.5.2 Entartung	359
16.6 Folgerungen	361
17 Kettenbrüche	363
17.1 Einleitung	363
17.1.1 Definition	363
17.2 Rationale Zahlen	364
17.2.1 Endliche Kettenbrüche	364
17.2.2 Euklidischer Algorithmus	364
17.2.3 Rekursionsformel	365
17.3 Irrationale Zahlen	368
17.3.1 Definition	368
17.3.2 Periodische Kettenbrüche	368
17.3.3 Nicht periodische Kettenbrüche	370
17.4 Approximation	374
17.4.1 Definition	374
17.4.2 Näherungsgesetz	374
17.4.3 Approximation einer Funktion	375
17.5 Folgerungen	376
18 Padé-Approximation	377
18.1 Warum die Taylor-Reihe nicht gut genug ist	377
18.2 Berechnung der Padé-Approximation	379
18.2.1 Potenzreihen von analytischen Funktionen	379
18.2.2 Padé-Approximation erstellen	381
18.3 Anwendungen	384
18.3.1 Totzeit Approximation	384
18.3.2 Digitale Signalmodellierung	390
18.4 Wann nun eine Padé-Approximation?	395
19 Kettenbrüche und Padé-Approximation	397
19.1 Kettenbrüche und Matrizen	397
19.2 Padé-Approximation von $\arctan x$	398
19.2.1 Approximation bis zur Ordnung 3	399
19.2.2 Approximation bis zur Ordnung 5	400
19.2.3 Der Grad von Zähler und Nenner	401

20 Die Gleichung von Burgers	403
20.1 Einleitung	403
20.1.1 Beziehung zu den Navier-Stokes-Gleichungen	403
20.2 Problemstellung	404
20.2.1 Numerische Wettervorhersage	404
20.2.2 Notation	405
20.3 Lösungsmethoden	405
20.3.1 Explizit	405
20.3.2 Implizit	411
20.4 Resultate	413
20.4.1 Explizit	413
20.4.2 Implizit	415
21 Finite Elemente in der Ebene	417
21.1 Einleitung	417
21.2 Problemstellung	417
21.2.1 Was sind Ansatzfunktionen?	418
21.2.2 Herausforderung FEM in der Ebene	418
21.3 Lösung	421
21.3.1 Vorbereitung	421
21.3.2 Schritt 1: Minimalproblem bilden	423
21.3.3 Schritt 2: Lösung durch Ansatzfunktionen approximieren	423
21.3.4 Schritt 3: Minimalprinzip anwenden auf Approximation	428
21.3.5 Schritt 4: Gleichungssystem aufstellen	429
21.3.6 Andere lineare Ansatzfunktionen	429
21.4 Folgerungen	430
22 QR-Zerlegung mit Givens-Rotationen	431
22.1 Einleitung	431
22.1.1 Anwendungsbeispiel Least-Squares	431
22.2 Numerische Probleme des Gram-Schmidt-Verfahrens	432
22.3 Lösung mit Givens-Rotationen	434
22.4 Folgerungen	436
23 Francis-Algorithmus	439
23.1 Grundlagen	440
23.1.1 Ähnlichkeitstransformation	440
23.1.2 Spezielle Matrizen	441
23.1.3 Eigenwerte	441
23.1.4 Givens-Rotationen	442
23.1.5 Householder-Transformation	442
23.2 Strategie	443
23.3 Vorbereitung der Matrix	444
23.4 Francis-Iteration erster Ordnung	445
23.4.1 Wahl der Shifts	446
23.5 Francis-Iteration der Ordnung n	447
23.6 Nachweis	448
23.6.1 Implementation	448

23.6.2 Resultate	449
24 Die Methode der konjugierten Gradienten	451
24.1 Voraussetzungen	451
24.2 Gradient Descent	452
24.2.1 Minimierungsproblem	452
24.2.2 Berechnung der Schrittweite	453
24.2.3 Berechnung des Gradienten	454
24.2.4 Probleme beim Gradient Descent	454
24.2.5 Beispiel zur Motivation der A-Orthogonalität	454
24.3 Herleitung des Algorithmus	455
24.3.1 Intuition	455
24.3.2 Optimale Suchrichtung	456
24.3.3 Wieso genügt Orthogonalisierung auf der letzten Richtung?	457
24.4 Algorithmus	459
24.5 Ergebnisse	460
24.5.1 Fazit	462
25 Numerische Ableitung	463
25.1 Einleitung	463
25.2 Lernen in neuronalen Netzwerken	463
25.2.1 Was ist Lernen?	464
25.2.2 Der Parameterraum	465
25.2.3 Das Optimierungsproblem	466
25.2.4 Der Backpropagation-Algorithmus	467
25.3 Ein alternativer Lösungsansatz	468
25.3.1 Finite Differenzen Methode	469
25.3.2 Taylor-Methode	469
25.4 Experiment	472
25.5 Folgerungen	474
26 Interpolation und numerische Ableitung	477
26.1 Das Problem der numerischen Ableitung	477
26.2 Die Ableitung des Interpolationspolynoms	478
26.3 Ableitungsverfahren	479
26.4 Fourier-Spektrum der Ableitung	480
Index	483

Vorwort

Dieses Buch entstand im Rahmen des Mathematischen Seminars im Frühjahrssemester 2020 an der Hochschule für Technik Rapperswil. Die Teilnehmer, Studierende der Abteilungen für Elektrotechnik, Informatik, Bauingenieurwesen und Erneuerbare Energie und Umwelttechnik der HSR, erarbeiteten nach einer Einführung in das Themengebiet jeweils einzelne Aspekte des Gebietes in Form einer Seminararbeit, über deren Resultate sie auch in einem Vortrag informierten.

Im Frühjahr 2020 war das Thema des Seminars die Numerik, also die Berechnung von mathematischen Resultaten mit Hilfe von Computern.

In einigen Arbeiten wurde auch Code zur Demonstration der besprochenen Methoden und Resultate geschrieben, soweit möglich und sinnvoll wurde dieser Code im Github-Repository dieses Kurses¹ [4] abgelegt. Im genannten Repository findet sich auch der Source-Code dieses Skriptes, es wird hier unter einer Creative Commons Lizenz zur Verfügung gestellt.

Das Umschlagbild zeigt die Resultate einer numerischen Simulation der Überschallströmung durch die Düse eines Raketenmotors mit Hilfe der Software OpenFOAM. Rechnergestützte Fluidodynamik bringt viele der in diesem Buch besprochenen Techniken zusammen. Die Lösungsverfahren für die partiellen Differentialgleichungen der Strömungsdynamik führen zum Beispiel auf sehr grosse lineare Gleichungssysteme. Sie führen aber oft auch numerische Instabilitäten ein, was nicht weiter verwunderlich ist, da auch die Strömungen selbst oft instabil sind. Es illustriert damit die besondere Bedeutung, die eine sorgfältige numerische Analyse auf der Basis der in diesem Buch vorgestellten Techniken hat.

¹<https://github.com/AndreasFMueller/SeminarNumerik.git>

Teil I

Grundlagen

Einleitung

Die Mathematik stellt Werkzeuge zur Verfügung, mit denen natürliche Prozesse exakt beschrieben werden können. Die Ingenieurwissenschaften haben über die Jahrhunderte deren Anwendbarkeit immer wieder mit unglaublichen Leistungen demonstriert. Riesige Bauwerke, Ozeanriesen, Flugzeuge, das Internet mit seiner unüberschaubaren Transportkapazität für Datenströme, die Landung auf dem Mond und auf anderen Planeten, die Messung winziger Elementarteilchen, die nur wenige Zeptosekunden überleben im Large Hadron Collider, die Detektion von Gravitationswellen, die die Raumzeit um nur einen Tausendstel des Durchmessers eines Protons verzerren, zeugen von der unvorstellbaren Präzision, mit der mathematische Modelle die Realität modellieren können.

Die Realität spielt aber trotzdem oft nicht mit. Messungen sind in der Realität nur mit eingeschränkter Genauigkeit möglich. Wie verändert sich die Qualität einer mathematischen Vorhersage, wenn die Eingabedaten nicht perfekt sind? Auch stehen nicht für jede mathematische Problemlösung analytische Formeln zur Verfügung. Es bleibt dann nichts anderes übrig, als die Berechnungen auf einem Computer durchzuführen, der unvermeidliche Rundungsfehler einführt, die die Resultate ebenfalls beeinträchtigen können. Wie kann man auch unter diesen erschwerten Bedingungen die Präzision der Ergebnisse sicherstellen, die die genannten technischen Meisterleistungen erst möglich gemacht haben?

Die numerische Mathematik befasst sich mit der Konstruktion und Analyse von Algorithmen für kontinuierliche mathematische Probleme. Die von einem Computer zur Verfügung gestellten Zahlensysteme sind nur Approximationen für die reellen Zahlen. Sie besser zu verstehen und die Auswirkungen ihre Unzulänglichkeiten besser in den Griff zu bekommen ist der Inhalt von Kapitel 1. Das wichtigste Werkzeug, die Iteration, wird darin einer gründlichen Analyse unterzogen und es werden Bedingungen ermittelt, unter denen der Erfolg garantiert werden kann. Es lassen sich Gesetzmäßigkeiten für die Fehler finden, die einerseits instabiles Verhalten vorhersagen können, aber andererseits auch dazu dienen können, den Fehler zu reduzieren und die Konvergenz eines Verfahrens zu beschleunigen.

Kapitel 2 kümmert sich um eine der ältesten Aufgaben der Numerik, nämlich die Lösung von Gleichungen. Da es sich dabei um eine sehr grundlegende Aufgabenstellung handelt, die immer wieder als Baustein umfassender Problemlösungen auftritt, ist hohe Genauigkeit und Geschwindigkeit besonders wichtig. Es wird gezeigt, wie der Newton-Algorithmus sich durch sogenannte *quadratische Konvergenz* auszeichnet.

Die Mathematik stellt eine grosse Zahl von nützlichen Funktionen zur Verfügung, deren exakte Berechnung jedoch oft umständlich und zeitaufwendig ist. Kapitel 3 zeigt, wie solche Funktionen mit hoher Genauigkeit durch Polynome approximiert werden können, die meist viel einfacher zu berechnen sind. Besonderes Augenmerk wird dabei darauf gelegt, die Approximationsfehler abzuschätzen, so dass ein Interpolationsverfahren immer den Genauigkeitsanforderungen der Anwendung angepasst werden kann. Spline-Interpolation und Bézier-Kurven runden als in der Praxis be-

sonders erfolgreiches Approximationsverfahren das Kapitel ab.

In der Analysis lernt man unter anderem, Integrale in geschlossener Form auszuwerten. Dabei entsteht oft der falsche Eindruck, mit genügend Ausdauer sei praktisch jedes Integral lösbar. Nichts könnte weiter von der Realität entfernt sein. Tatsächlich ist die Berechnung von Integralen eine weitere Grundaufgabe, für die die Numerik effiziente Methoden bereitstellen soll. Kapitel 4 diskutiert die Mittelpunktsformel und die Trapezregel und zeigt, wie sich die Genauigkeit dieser Verfahren sehr erfolgreich dank einer sorgfältigen Fehleranalyse steigern lässt.

Zu den wichtigsten mathematischen Modellen für natürliche Prozesse gehören gewöhnliche Differentialgleichungen. Hier ist die Verfügbarkeit analytischer Lösungen noch prekärer als bei den Integralen. Glücklicherweise stehen Verfahren sehr hoher Präzision in weit verbreiteten Softwarepaketen für die numerische Analyse zur Verfügung. Die Techniken, mit denen sich diese Präzision erreichen lässt, werden in Kapitel 5 dargestellt. Neben dem universell einsetzbaren Runge-Kutta-Verfahren, welches zur Klasse der Einschrittverfahren gehört, werden auch Mehrschrittverfahren vorgestellt sowie Software zur Lösung gewöhnlicher Differentialgleichungen. Es wird auch gezeigt, wie man die Ableitung der Lösung einer Differentialgleichung nach Anfangsbedingungen und nach Parametern modellieren kann. Die Lösung von Randwertproblemen wird damit effizient möglich.

Im Kapitel 2 wurde bewusst auf die Behandlung linearer Gleichungssysteme verzichtet. Zwar kennt man den Gauss-Algorithmus aus den Grundvorlesungen, er benötigt aber für n Unbekannte eine Laufzeit von der Größenordnung $O(n^3)$, was für die sehr grosse Zahl von Unbekannten, mit der man es zum Beispiel bei der numerischen Lösung von partiellen Differentialgleichungen zu tun hat, zu langsam ist. Die besonderen Eigenschaften linearer Gleichungssysteme ermöglichen aber alternative Ansätze zu deren Lösung. Das Kapitel 6 beginnt mit einer Diskussion von in der Praxis häufig vorkommenden Spezialfällen von Gleichungssystemen, die sich schneller als in $O(n^3)$ lösen lässt. Es diskutiert dann eine Reihe von iterativen Verfahren zur Gleichungslösung und zeigt, wie der Spektralradius einer Matrix die zentrale Masszahl für den Erfolg oder Misserfolg solcher Verfahren ist. Es schliesst mit einer Diskussion zweier einfacher Verfahren für die QR-Zerlegung und für das Eigenwertproblem, die im zweiten Teil vertieft wird.

Partielle Differentialgleichungen stellen mathematische Modelle für ausgedehnte Systeme wie elektromagnetische Felder, Temperaturverteilung, Dichte etc. bereit. Für deren Lösung sind die Methoden für gewöhnliche Differentialgleichungen nicht geeignet. Kapitel 7 führt die grundlegenden Definitionen aus der Theorie der partiellen Differentialgleichungen ein und vermittelt die Grundideen der wichtigsten Diskretisations- und Lösungsverfahren. Zur Sprache kommen finite Differenzen, finite Volumina und finite Elemente.

1

Berechnung

Die numerischen Datentypen eines digitalen Computers sind Approximationen für die abstrakten Zahlensysteme \mathbb{N} , \mathbb{Z} , \mathbb{Q} und \mathbb{R} . Man kann sie daher verwenden, die in der Analysis und der linearen Algebra definierten Konzepte wie Grenzwerte, Integrale oder inverse Matrizen zu berechnen. Da sie jedoch nur Annäherungen sind, werden die gewohnten Rechenregeln nicht immer gültig sein. In \mathbb{R} kann die Addition dreier Zahlen zum Beispiel in beliebiger Reihenfolge durchgeführt werden, für `double`-Zahlen auf einem Computer ist dies nicht der Fall. Zum Beispiel kann man in Octave die folgende Rechnung durchführen¹:

```
octave:1> a = 1;
octave:2> b = 0.0000000000000001;
octave:3> x = a+(b+b)
x = 1.0000
octave:4> y = (a+b)+b
y = 1
octave:5> x-y
ans = 2.2204e-16
octave:6> x-1
ans = 2.2204e-16
octave:7> y-1
ans = 0
```

Das Assoziativgesetz verlangt, dass $a + (b + b) = (a + b) + b$ ist und zunächst will es auch den Anschein haben, dass x und y tatsächlich gleich sind. Ungewöhnlich ist nur, dass der Wert von x mit auf den ersten Blick unnötigen Nachkommastellen angezeigt wird. Diese Nullen deuten jedoch an, dass $x \neq 1$ aber $y = 1$, wie man in Schritt 6 und 7 sieht, wenn man die Differenz zu 1 bestimmt.

Das Beispiel verdeutlicht, dass auf einem Computer zur Verfügung stehende numerische Datentypen die gewohnten Rechengesetze verletzen und neue Unzulänglichkeiten wie Rundungsfehler in die Berechnung injizieren. Die Numerik muss sich daher mit der Frage befassen, welcher Art diese neuen Effekte sind und wie ihnen effektiv begegnet werden kann. Nur so kann die Zuverlässigkeit numerisch gefundener Resultate garantiert werden.

¹Das Beispiel zeigt eine Problem für Zahlen vom Typ `double`. Das Programm `assoziativ.cpp` im Verzeichnis `buch/chapters/experiments/assoziativ` findet analoge Beispiel für die anderen Floatingpoint Typen `float` und `long double`.

Dieses Kapitel befasst sich mit den Eigenschaften von Computer-Zahlensystemen. Im ersten Abschnitt werden die Zahlensysteme beschrieben und ihre Vor- und Nachteile gegeneinander abgewogen. Im zweiten Abschnitt wird gezeigt, wie Resultate bei unvorsichtiger Vorgehensweise verfälscht werden können. Rundungsfehler sind unvermeidlich, das Ziel muss daher sein, ihre Grösse unter Kontrolle zu halten. Numerische Instabilität liegt vor, wenn die Berechnung aufgrund von numerischen Effekten völlig aus dem Ruder läuft und sinnlose Resultate liefert, dies wird in Abschnitt 1.5 untersucht.

1.1 Zahlensysteme

Auf modernen Allzweck-Prozessoren steht eine ganze Reihe verschiedener numerischer Datentypen mit unterschiedlichen Eigenschaften bezüglich Geschwindigkeit und Fehlerverhalten zur Verfügung. In diesem Abschnitt sollen sie vorgestellt und miteinander verglichen werden. Es gilt, den für eine Berechnung zweckmässigsten Typen zu wählen, wobei Speicherbedarf, Laufzeit und Parallelisierbarkeit wesentliche Aspekte sind.

Microcontroller sind im Vergleich zu Allzweckprozessoren oft stark eingeschränkt. Meist sind nur Ganzahltypen mit oft sehr beschränkter Länge implementiert. Manchmal kann die Arithmetik-Einheit des Prozessors nicht einmal eine Multiplikation in Hardware ausführen, sie muss in Software zeitaufwendig nachgebildet werden. Für Floatingpoint-Operationen muss oft Bibliotheken zurückgegriffen werden, die den Speicherbedarf erhöhen und langsam sind. Die Implementation von numerischen Berechnungen in eingebetteten Anwendungen ist daher mit besonderen Herausforderungen konfrontiert.

Dieselbe Schwierigkeit haben auch Allzweck-Prozessoren, wenn die Genauigkeitsanforderungen die Möglichkeiten der von der Prozessor-Hardware implementierten Typen übersteigt. Dieser Fall tritt beispielsweise bei Berechnungen in der Kryptographie auf, wo oft mit Ganzzahlen mit Tausenden von Stellen gerechnet werden muss. Im Abschnitt 1.1.4 mit der GNU Multiprecision-Library ein Beispiel einer solchen Bibliothek vorgestellt.

1.1.1 Zahlendarstellung bezüglich verschiedener Basen

Allen Zahlensystemen gemeinsam ist die Positionsdarstellung. Eine Zahl wird als Zeichenkette $x = x_n x_{n-1} \dots x_3 x_2 x_1 x_0$ geschreiben, wobei die Zeichen x_i Ziffern mit $0 \leq x_i < b$ sind. b ist die Basis des Zahlensystems. Der Wert der Zahl x ist dann

$$x = \sum_{k=1}^n x_k b^k.$$

Um die Basis deutlich zu machen, hängen wir die Basis als Index an eine Zahlendarstellung an. Es ist also zum Beispiel

$$1291_{10} = 10100001011_2 = 1202211_3 = 2413_8 = 508_{16}.$$

Bruchzahlen können analog dargestellt werden. Die Zeichenkette

$$x = x_n x_{n-1} \dots x_2 x_1 x_0 \cdot x_{-1} x_{-2} x_{-3} \dots x_{-m} \dots$$

hat den Wert

$$x = \sum_{k=-m}^n x_k b^k.$$

Typ	0.5	0.2
float	0.50000000000000000000000000000000	0.20000000298023223877
double	0.50000000000000000000000000000000	0.20000000000000000000000001110
long double	0.50000000000000000000000000000000	0.20000000000000000000000001110

Tabelle 1.1: Nicht alle exakt darstellbaren Dezimalzahlen sind auch im Binärsystem exakt darstellbar. Die Zahl 0.2_{10} führt auf einee periodische Binärzahl, die beim Abspeichern gerundet werden muss. Je nach Datentyp weicht der gerundete Wert mehr oder weniger stark ab.

Die Zahl π hat daher die Darstellungen

$$\begin{aligned}\pi &= 11.001001000011111011011_2 \\ &= 10.01021101222201_3 \\ &= 3.11037552_8 \\ &= 3.1415926_{10} \\ &= 3.243F6A_{16}\end{aligned}$$

in den Basen 2, 3, 8, 10 und 16. Eine grösitere Basis erlaubt zwar eine kompaktere Darstellung, aber für die Rechnung ermöglicht die binäre Darstellung die einfachste und damit auch schnellste Implementation.

Man beachte, dass endliche Dezimalbrüche in anderen Basen durchaus nicht mehr endlich zu sein brauchen. So ist zum Beispiel

$$\begin{aligned}\frac{1}{2} &= 0.5_{10} = 0.1_2, \\ \frac{1}{5} &= 0.2_{10} = 0.001100110011\cdots = 0.\overline{0011}_2, \\ \frac{1}{3} &= 0.\overline{3}_{10} = 0.\overline{01}_2.\end{aligned}$$

Eine Konsequenz dieser Beobachtung ist, dass nur schon die Umwandlung einer Dezimalzahl ins Binärsystem und die Rückumwandlung in eine Dezimalzahl den Wert verändern kann. Zum Beispiel² bewirkt der C++-Code

```
double x = 0.2;
std::cout << x;
```

dass der Compiler zunächst die Dezimalzahl 0.2 in eine Binärzahl verwandelt, diese Form wird im ausführbaren Code gespeichert. Zur Laufzeit des Programms muss die I/O-Bibliothek dann die gespeicherte Zahl wieder in eine Dezimalzahl verwandeln. Für $x = 0.5$ wäre das unproblematisch, da diese Zahl sowohl dezimal wie auch binär ein endlicher Dezimalbruch ist. Für $x = 0.2$ tritt jedoch eine Abweichung auf, weil die im Code gespeichert Binärzahl nicht exakt in 0.2 zurückgewandelt werden kann. Stattdessen erhält man abhängig vom Datentyp die abweichenden Werte von Tabelle 1.1. Dass kein Unterschied zwischen `double` und `long double` besteht, ist nur scheinbar. Multipliziert man x mit 5 vor dem Output, wird plötzlich ein Unterschied sichtbar.

²Dieses Beispiel wurde mit dem Programm `format.cpp` aus dem Verzeichnis `buch/chapters/experiments/limits` von [4] gerechnet.

1.1.2 Festkommazahlen

Die bekanntesten Festkommazahlen sind die Ganzzahltypen, die jeder Prozessor zum Beispiel für Adressierung, Zähler und Indizierung benötigt. Damit ist auch bereits klar, dass man immer damit rechnen kann, dass mindestens die Addition und die Subtraktion von ganzen Zahlen mit Wortlängen implementiert sind, die der Prozessor zum Beispiel für relative Adressierung braucht. Ebenso kann man davon ausgehen, dass jeder Kern eines Prozessors eine Einheit für ganzzahlige Operationen hat, denn er könnte sonst nicht einmal die minimal notwendigen Adressberechnungen durchführen.

Addition

Das Verfahren der “schriftlichen Addition”, welches man in der Primarschule lernt, funktioniert auch für die Berechnung einer Summe in jeder beliebigen anderen Basis.

Vorzeichen

Ganzzahlen mit Vorzeichen können auf verschiedene Arten binär dargestellt werden, weitgehend durchgesetzt hat sich für Festkommazahlen jedoch die Zweierkomplement-Darstellung³. In ihr werden 8-bit Zeichenketten wie folgt als Zahlen interpretiert:

01111111	=	127
01111110	=	126
⋮		⋮
00000010	=	2
00000001	=	1
00000000	=	0
11111111	=	-1
11111110	=	-2
11111101	=	-3
⋮		⋮
10000010	=	-126
10000001	=	-127
10000000	=	-128

Diese Codierung ist in Hardware besonders leicht implementierbar. Ein Zähler für eine vorzeichenlose Ganzzahl von 8 bit Länge, initialisiert mit **10000000** wird beim Hochzählen genau die Zahlen von -128 bis 127 aufzählen.

Die Entgegengesetzte einer Zahl kann nach der folgenden Regel gefunden werden:

1. Man nehme das Komplement jedes einzelnen Bits einer Zahl
2. Addiere 1.

³Der Exponent einer Gleitkommanzahl ist zwar auch eine Ganzzahl, er wird aber gemäss Standard IEEE 754 nach einem anderen Verfahren codiert, siehe dazu auch Abschnitt 1.1.3

Beispiel. Die Zahl -1291 soll als 16-bit Ganzzahl in Zweierkomplement-Darstellung geschrieben werden. Zunächst wird die Binärdarstellung benötigt: $1291_{10} = 0000010100001011_2$.

1. Bits komplementieren: 1111101011110100

2. 1 addieren: 1111101011110101 ○

Die Addition vorzeichenbehafteter Ganzzahlen funktioniert für die Zweierkomplementdarstellung nach dem bekannten Algorithmus für die Addition. Die Differenz $111 - 88$ kann man als Summe $111 + (-88)$ schreiben. Als 8-bit Binärzahlen sind die beiden Operanden 01101111 und 10101000 . Ihre Summe ist

$$\begin{array}{r} 01101111 \\ 10101000 \\ \hline 00010111 \end{array}$$

Dabei ist zwar ein Überlauf aufgetreten, aber dieser kann ignoriert werden. Tatsächlich ist $23_{10} = 10111_2$. Der grosse Vorteil dieser Vorzeichenkonvention ist also, dass für die Addition vorzeichenbehafteter Ganzzahlen in Zweierkomplement-Darstellung die gleiche vorhandene Hardware verwendet werden kann wie für die Addition vorzeichenloser Ganzzahlen.

Multiplikation und Division

Man kann allerdings nicht davon ausgehen, dass ein Prozessor auch die Multiplikation von ganzen Zahlen und erst recht die Division von ganzen Zahlen implementiert. In den meisten Fällen benötigt der Prozessor nur die Multiplikation mit kleinen Zweierpotenzen, die sich viel effizienter als Verschiebeoperationen durchführen lassen.

Nachkommateil

Bisher wurden ausschliesslich Ganzzahlen betrachtet. Man kann diese Ganzzahlen aber auch als rationale Zahlen mit einem Nachkommateil fester Länge betrachten. Man könnte sich zum Beispiel nach den ersten 8 bit einer 16-bit Zahl ein Komma denken und die nachfolgenden Bits als Bruchteil betrachten. Die Bitfolge 0000000110010000 muss dann als

$$000000011.00100000_2 = 3.125_{10}$$

interpretiert werden. An den Algorithmen für Addition und Subtraktion ändert sich nichts, es ist daher keine neue Hardware für die Implementation dieser Operationen notwendig, die Ganzzahloperationen reichen aus.

Etwas komplizierter ist die Sache bei der Multiplikation. Nehmen wir an, dass wir mit 8-bit Festkomma-Zahlen arbeiten mit einem Nachkommateil von 4 bits. Wir möchten das Produkt $3.125 \cdot 2.0625$ berechnen. Die Binärdarstellungen dieser Zahlen sind $3.125_{10} = 11.001_2$ und $2.0625_{10} = 10.0001_2$. Das Produkt der 8-bit Ganzzahlen 00110010 und 00100001 wird mehr Platz beanspruchen, im schlimmsten Fall 16 bit:

$$00110010_2 \cdot 00100001_2 = 0000011001110010_2.$$

In den Faktoren sind die letzten 4 Stellen jeweils als Nachkommateil zu unterinterpretieren, also sind im Produkt die letzten 8 Stellen als Nachkommateil zu interpretieren. Das Produkt, wieder als 8-bit Festkommazahl geschrieben ist daher

$$0011.0010_2 \cdot 0010.0001_2 = 00000110.01110010_2 \approx 0110.0111_2$$

Auch für die Multiplikation ist keine neue Hardware erforderlich, doch muss das Resultat entsprechend mit Schiebeoperationen wieder so formattiert werden, dass das Komma an der “richtigen” Stelle landet.

Mit den Ganzzahl-Operationen einer CPU lassen sich also auch sehr schnelle Festkomma-Operationen realisieren.

Vor- und Nachteile von Festkommazahlen

- ⊕ Eine beliebige reelle Zahl muss bei der Darstellung als Festkommazahl um Maximal die Hälfte der Wertigkeit der letzten Stelle gerundet werden. Dieser absolute Fehler ist konstant.
- ⊕ Operationen sind typischerweise deutlich schneller als mit Gleitkommazahlen vergleichbarer Grösse. Dies gilt insbesondere dann, wenn die Operationen zum Teil in Software realisiert werden müssen.
- ⊕ Die Addition und Subtraktion sind von derart elementarer Bedeutung für einen CPU-Kern, dass jeder Kern mindestens eine Einheit für Ganzzahl-Operationen hat. In einer Multicore CPU kann man daher davon ausgehen, dass Festkomma-Operationen sich auf verschiedenen Cores nicht gegenseitig behindern.
- ⊖ Kleine Zahlen können nur mit wenigen signifikanten Stellen dargestellt werden.
- ⊖ Schon für mässig grosse Zahlen ist Überlauf möglich.

1.1.3 Gleitkommazahlen

Gleitkommazahlen erweitern den Bereich der darstellbaren Zahlen dadurch, dass sie zu einer Festkommazahl einen Exponentialfaktor hinzunehmen. Eine Gleitkommazahl x ist also von der Form

$$x = m \cdot b^k.$$

m heisst Mantisse, b ist die Basis und k ist der Exponent, üblicherweise eine kleine Ganzzahl. Die Mantisse wird typischerweise so strukturiert, dass genau eine Stelle vor dem Komma steht. Im Dezimalsystem sind also

$$1291 = 1.291 \cdot 10^3, \quad \gamma = 5.772156649 \cdot 10^{-1}, \quad N_A = 6.0221476 \cdot 10^{23}$$

korrekte Gleitkommazahlen.

Die meisten heutigen Prozessoren rechnen ausschliesslich binär. Sowohl für die Mantisse wie für den Exponenten wird daher eine Binärdarstellung verwendet, die Basis ist $b = 2$. Da die einzige Stelle vor dem Komma eine 1 sein muss, wird sie normalerweise nicht gespeichert. Das Vorzeichen der Zahl wird separat gespeichert.

Der 32 bit umfassende Gleitkommotyp `float` hat eine Mantisse von 24 bit, wovon aber nur 23 Bit gespeichert werden müssen. Von den verbleibenden 9 bit wird eines als Vorzeichen verwendet und 8 als Exponent. Mit einer 8 bit Ganzzahl lassen sich die Zahlen von 0 bis 255 darstellen. Um negative Exponenten zu ermöglichen, muss 127 subtrahiert werden. Die 8 Exponenten-bits codieren also die Exponenten -127 bis 128.

Die Zahl

$$\pi = 3.14159265_{10} = 11.0010100110001011000010_2 = 1.10010100110001011000010_2 \cdot 2^1$$

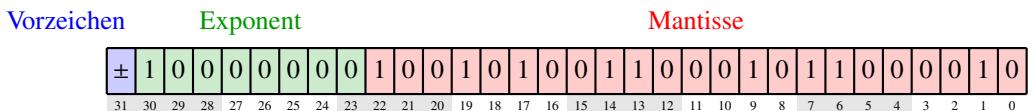
kann daher als `float`-Gleitkommazahl wie folgt gespeichert werden:

	float	double	long double
kleinste darstellbare Zahl	$1.17549 \cdot 10^{-38}$	$2.22507 \cdot 10^{-308}$	$3.3621 \cdot 10^{-4932}$
grösste darstellbare Zahl	$3.40282 \cdot 10^{38}$	$1.79769 \cdot 10^{308}$	$1.18973 \cdot 10^{4932}$
ϵ	$1.19209 \cdot 10^{-7}$	$2.22045 \cdot 10^{-16}$	$1.0842 \cdot 10^{-19}$
kleinster Exponent	-125	-1021	-16381
grösster Exponent	128	1024	16384
kleinste denormalisiert Zahl:	$1.4013 \cdot 10^{-45}$	$4.94066 \cdot 10^{-324}$	$3.6452 \cdot 10^{-4951}$

Tabelle 1.2: Eigenschaften der Gleitkommatypen float, double und long double. Die Zeile ϵ ist die Differenz zwischen 1 und der kleinsten darstellbaren Zahl, die grösser ist als 1. Einzelne Exponentenwerte haben eine spezielle Bedeutung (siehe Text), daher fallen die kleinstmöglichen Exponenten grösser aus als aufgrund ihrer Bitlänge zu erwarten ist.

Typ	Bytes	Mantisse	Exponent	IEEE-754
half, binary16	2	10	5	*
float, binary32	4	23	8	*
extended	5	29	10	
double, binary64	8	52	11	*
double extended, long double	10	63	15	*
quad binary128	16	112	15	*
binary256	32	236	19	*

Tabelle 1.3: Übliche Gleitkommatypen mit Länge der Mantisse und des Exponenten. Die meisten Compiler implementieren nur float und double, manchmal auch noch long double. Die im Standard IEEE 754-2008 definierten Typen sind in der letzten Spalte mit einem * versehen.



Die Ganzzahl $10000000_2 = 128_{10}$ im Exponentenfeld muss um 127 verringert werden um den Exponenten 1 zu ergeben.

Die grösste und kleinste mit einem float darstellbare positive Zahl ist somit

$$1.111111111111111111111111_2 \cdot 2^{-128} = 3.4028232_{10} \cdot 10^{-38}$$

$$1.00000000000000000000000000000000_2 \cdot 2^{-127} = 5.8774717_{10} \cdot 10^{-39}$$

Den 24 binären Mantissenbits entsprechen gut 7 Dezimalstellen.

Genau genommen können noch etwas kleinere Zahlen dargestellt werden, wenn man auf die Konvention verzichtet, dass vor dem Komma eine 1 stehen muss. Man nennt solche noch kleineren Zahlen *denormalisiert*. Da das Floatingpoint-Format immer auch eine implizite führende 1 beinhaltet, braucht es zusätzliche Konventionen für denormalisierte Zahlen, die die Abwesenheit der 1 signalisieren (Siehe auch den Abschnitt Denormalisierung auf Seite 15).

Gebräuchliche Formate

Der 1985 verabschiedete Standard IEEE 754 beschreibt die heute gebräuchlichen Implementation von Gleitkommatypen abschliessend. Damit ist gewährleistet, dass numerische Rechnungen auf verschiedenen Prozessoren reproduzierbare Resultate geben.

Die C++-Standardbibliothek bietet im `<limits>` Header die Möglichkeit, Informationen über die Datentypen zu erhalten. In Tabelle 1.2 sind die Eigenschaften der gebräuchlichsten Typen zusammengestellt. Allerdings offenbart sich hier auch ein Problem dieser Implementation von `<limits>`. Wie wir weiter unten sehen werden, definiert der IEEE 754 Standard Werte für $\pm\infty$, die kleiner sind als die angegebenen maximalen Werte.

Aktuelle Compiler unterstützen typischerweise die Gleitkomma-Typen `float`, `double` und `long double`. Bei Microkontrollern, wo Berechnungen mit der hohen Präzision eines `double` nur schon wegen des Platzbedarfs der Werte und des Zeitbedarfs für die Operationen kaum sinnvoll sind, ist oft nur der `float`-Typ implementiert. Der GNU-Compiler für die 8-bit AVR-Prozessorfamilie stellt behandelt zum Beispiel `double` genau gleich wie `float`. Graphikkarten unterstützen oft noch einen halben Gleitkommatyp `binary16`, dessen Genauigkeit für die Darstellung von 3D-Objekten ausreichend ist.

Rundung

Der IEEE 754 Standard schreibt auch vor, wie Resultate gerundet werden müssen. Bei vielen Operationen entstehen Resultate, die einen längeren Nachkommanteil haben als in das gegebene Gleitkommaformat passt, das Resultat muss gerundet werden. Der Standard kennt fünf verschiedene Rundungsverfahren und empfiehlt, dass die Funktionen wie Wurzeln, Exponentialfunktionen, trigonometrische Funktionen und viele weitere so implementiert werden müssen, dass das Resultat korrekt gerundet ist. Damit ist gemeint, dass das Resultat entsteht, in dem auf das mathematisch exakte Resultat die gewählte Rundungs-Regel angewendet wird.

Der Standard bezweckt mit diesen Regeln, dass man sich im Rahmen der Rundungsgenauigkeit auch auf die letzte Stelle eines Gleitkommawertes verlassen kann. Dies ist keinen Selbstverständlichkeit. Gleitkomma-Implementationen von GPUs beispielsweise erfüllen diese Bedingung oft nicht und garantieren in ihren Spezifikationen manchmal nur korrekte Resultate mit Ausnahme der letzten 1-2 bits.

Die Null

Die Gleitkomma-Darstellung definiert, dass die Mantisse ein implizites führendes 1-bit hat. In diesem Format lässt sich die Null aber nicht darstellen, es ist also eine separate Definition nötig:

- Alle Exponentenbits = 0.
- Alle Mantissenbits = 0.
- Vorzeichen 0 oder 1 erlaubt die Unterscheidung zwischen +0 und -0.

Dies ist ein Spezialfall einer denormalisierten Zahl (siehe auch Abschnitt Denormalisierung weiter unten auf Seite 15).

Unendlich grosse Werte

Im Laufe einer numerischen Berechnung kann es vorkommen, dass die Resultate so gross werden, dass sie nicht mehr im gegebenen Gleitkommatyp gespeichert werden können. Der Standard verlangt, dass diese Situation durch einen speziellen Wert für unendlich grosse Zahlen wiedergegeben werden kann, der wie folgt definiert ist:

- Alle Exponenten-Bits = 1
- Alle Mantissenbits = 0
- Vorzeichenbit 0 oder 1 um zwischen $+\infty$ und $-\infty$ unterscheiden zu können.

Dies bedeutet, dass der grösste nutzbare Exponent des `float`-Typs nur noch 127 ist.

Denormalisierung

Die Mantisse einer Gleitkommazahl beginnt immer mit einer 1, die aber nicht gespeichert wird. Wird eine Zahl kleiner als mit dem zur Verfügung stehenden Exponenten-Bereich darstellbar, kann sie nicht mehr in diesem Format gespeichert werden. Um solche Zahlen darzustellen, wurde vom Standard einem Exponenten aus lauter 0 eine besondere Bedeutung gegeben. Für den Typ `float` entspricht er nicht mehr dem Exponenten -127 sondern -126 und das implizite führend Bit der Mantisse ist jetzt 0. Mit diesen sogenannten denormalisierten Zahlen lassen sich noch kleinere Wert darstellen, die aber nicht mehr so präzis sind, weil sie weniger signifikante Stellen aufweisen.

Vor- und Nachteile

- ⊕ Wir eine beliebige reelle Zahl als Gleitkommazahl abgespeichert, muss sie um maximal um den halben Wert des letzten Mantissenbits gerundet werden. Der absolute Wert desselben hängt jedoch vom Exponenten ab. Bei l Mantissenbits ist er aber immer um den Faktor 2^l kleiner als die Zahl selbst. Es tritt ein konstanter *relativer Fehler* auf.
- ⊕ Dank des grossen Wertebereiches sind Über- und Unterlauf unwahrscheinlich.
- ⊖ Gleitkommazahlen brauchen mehr Speicherplatz. Der kleinste Gleitkommatyp `float` ist mit 32 bit bereits so gross wie der gebräuchlichste Ganzzahltyp `long`.
- ⊖ Geschwindigkeit: sofern keine Hardwarebeschleunigung zur Verfügung steht sind Gleitkommaoperationen deutlich langsamer als Operationen mit Festkommazahlen.
- ⊖ In einer Multicore CPU hat nicht unbedingt jeder Kern eine Gleitkomma-Einheit. Gleitkommaoperationen in verschiedenen Threads können sich also gegenseitig behindern.

1.1.4 Hochpräzisionsbibliotheken

Die Diskussion numerischer Effekte in Abschnitt 1.2 zeigen, dass es nötig sein kann, Berechnungen mit sehr viel höherer Präzision durchzuführen, um die Genauigkeit des Schlussresultats garantieren zu können. Weder die Gleitkommadatentypen noch die Arithmetikprozessoren der CPUs unterstützen aber beliebig grosse Gleitkommazahlen. Es müssen daher Bibliotheken verwendet werden, die solche Datentypen und die arithmetischen Operationen nachbilden.

```

1  || int      main(int argc, char *argv[]) {
2  || |    int      bits = 32;
3  || |    while (bits <= 512) {
4  || |        experiment(bits);
5  || |        bits <= 1;
6  || |
7  || }
8  || |
9  || return EXIT_SUCCESS;

```

Listing 1.1: Treiberprogramm zur Berechnung von e^x mit verschiedenen Hochpräzisionsbibliotheken.

Als Beispiel soll die Exponentialfunktion e^{-100} mit Hilfe der Taylor-Reihe

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

berechnet werden. Die Berechnung der einzelnen Terme kann dank der Formel

$$\frac{x^k}{k!} = \frac{x^k}{1 \cdot 2 \cdot 3 \cdot \dots \cdot (k-1) \cdot k} = \frac{x}{1} \cdot \frac{x}{2} \cdot \frac{x}{3} \cdot \dots \cdot \frac{x}{k-1} \cdot \frac{x}{k} \quad (1.1)$$

iterativ mit einer einfachen Multiplikation mit dem “nächsten” Faktor x/k erfolgen.

In Abschnitt 1.2.2 wird gezeigt, dass für negative Argument x die Berechnung wegen Verschmiereung mit sehr viel grösserer Genauigkeit durchgeführt werden muss, wenn das Resultat exakt sein soll. Das Problem kann behoben werden, wenn man für negatives x die Formel $e^x = 1/e^{-x}$ ausnutzt und die die Taylor-Reihe auf das Argument $-x$ anwendet, wo die Schwierigkeit nicht auftritt.

In diesem Beispiel wird dieses Problem illustriert, indem die gleiche Berechnung mit verschiedener Genauigkeit mit einer Hochpräzisionsbibliothek durchgeführt wird. Dazu wird das Treiberprogramm von Listing 1.1 verwendet. Die Implementation der Funktion `experiment(double x)` mit verschiedenen Hochpräzisionsbibliotheken wird weiter unten beschrieben. Man erwartet, dass bei ausreichend grosser Präzision ein ausreichend genaues Resultat erhalten werden kann.

GNU GMP

Die freie GNU GMP Bibliothek erschien 1991 und kann auf <http://gmplib.org> gefunden werden. Die Implementation ist nicht zu IEEE 754 konform und es ist nicht garantiert, dass verschiedene Maschinen identische Resultate mit dem gleichen Code erhalten. Daher wird empfohlen, für neue Projekte die MPFR Bibliothek (siehe nächsten Abschnitt) zu verwenden.

Der Code in Listing 1.2 zeigt die Berechnung der Taylor-Reihe mit Hilfe von GNU GMP. In Zeile 44 in der Funktion `void experiment(int bits)` wird die minimale Genauigkeit, mit der die Bibliothek rechnen soll. Die Aufrufe von `mpf_init(mpf_t op)` in der Funktion `double gmpexp(double x)` initialisieren die Variablen mit mindestens dieser Genauigkeit. Welche Genauigkeit genau gewählt wird hängt von der Wortlänge der verwendeten Maschine ab.

In den Zeilen 21–29 wird die Taylor-Reihe summiert. In der Variable `P` ist der Wert des aktuellen Summanden gespeichert. Der nächste Summand kann daraus gemäß (1.1) mittels $P \cdot x/i$ berechnet werden, wobei i der Wert der Zählervariablen ist, mit der die Summanden indiziert werden. In Zeile 31 wird die Summe wieder in einen `double`-Wert mit Maschinenpräzision umgewandelt.

```

1 #include <gmp.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <math.h>
5
6 double gmpexp(double x) {
7     mpf_t X, P, S;
8     /* Variablen initialisieren */
9     mpf_init(X);
10    mpf_init(P);
11    mpf_init(S);
12    mpf_set_d(X, x);
13    mpf_set_d(P, 1.);
14    mpf_set_d(S, 1.);

15    /* Hilfsvariablen fuer Zwischenresultate */
16    mpf_t R, Q;
17    mpf_init(R);
18    mpf_init(Q);

19    for (int i = 1; i < 1000; i++) {
20        /* P = P * X / i */
21        mpf_mul(Q, P, X);
22        mpf_div_ui(R, Q, i);
23        mpf_set(P, R);
24        /* S = S + P */
25        mpf_add(Q, S, P);
26        mpf_set(S, Q);
27    }

28    double result = mpf_get_d(S);

29    mpf_clear(X);
30    mpf_clear(P);
31    mpf_clear(S);
32    mpf_clear(R);
33    mpf_clear(Q);

34    return result;
35}
36
37 void experiment(int bits) {
38     /* Setze Mindestgenauigkeit der Variablen */
39     mpf_set_default_prec(bits);
40     double x = -100;
41     double y = gmpexp(x);
42     double z = exp(x);

43     printf("bits:      %d\n", bits);
44     printf("GMP:      %.20g\n", y);
45     printf("Prozessor:  %.20g\n", z);
46     printf("Fehler:    %.20g\n\n", y - z);
47 }
48
49
50
51
52
53 }
```

Listing 1.2: C-Programm zur Berechnung von e^x mit Hilfe der Taylor-Reihe, implementiert mit GMP

Bits	GNU GMP	double	Fehler
32	$-2.378447678 \cdot 10^{-19}$	$3.720075976 \cdot 10^{-44}$	$-2.378447678 \cdot 10^{-19}$
64	$-2.378447678 \cdot 10^{-19}$	$3.720075976 \cdot 10^{-44}$	$-2.378447678 \cdot 10^{-19}$
128	$1.384432347 \cdot 10^{-1}$	$3.720075976 \cdot 10^{-44}$	$1.384432347 \cdot 10^{-1}$
256	$-1.339761431 \cdot 10^{-39}$	$3.720075976 \cdot 10^{-44}$	$-1.339798632 \cdot 10^{-39}$
512	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$-4.978412222 \cdot 10^{-60}$

Tabelle 1.4: Tabelle der Resultate der Berechnung von e^{-100} mit Hilfe der Taylor-Reihe der Exponentialfunktion unter Verwendung des Programms von Listing 1.2 mit verschiedenen Bitlängen.

Bits	GNU MPFR	double	Fehler
32	$4.040280437 \cdot 10^{-32}$	$3.720075976 \cdot 10^{-44}$	$4.040280437 \cdot 10^{-32}$
64	$-1.797196708 \cdot 10^{-23}$	$3.720075976 \cdot 10^{-44}$	$-1.797196708 \cdot 10^{-23}$
128	$3.777954147 \cdot 10^{-3}$	$3.720075976 \cdot 10^{-44}$	$3.777954147 \cdot 10^{-3}$
256	$-5.943812579 \cdot 10^{-39}$	$3.720075976 \cdot 10^{-44}$	$-5.943849780 \cdot 10^{-39}$
328	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$3.196140646 \cdot 10^{-57}$
329	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$-1.941580766 \cdot 10^{-58}$
330	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$5.326901077 \cdot 10^{-58}$
331	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$4.132082144 \cdot 10^{-58}$
332	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$7.467618333 \cdot 10^{-59}$
333	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$-8.463300777 \cdot 10^{-59}$
334	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	$-9.956824444 \cdot 10^{-60}$
512	$3.720075976 \cdot 10^{-44}$	$3.720075976 \cdot 10^{-44}$	0

Tabelle 1.5: Tabelle der Resultate der Berechnung von e^{-100} mit Hilfe der Taylor-Reihe der Exponentialfunktion unter Verwendung des Programms von Listing 1.3 mit verschiedenen Bitlängen.

Die Resultate der Berechnung sind in Tabelle 1.4 zusammengestellt. Die Spalte double enthält die mit der Formel $e^x = 1/e^{-x}$ mit Maschinengenauigkeit erhaltenen Werte mit Maschinengenauigkeit. Zunächst fällt auf, dass der Fehler der Berechnung mit 32 bit und mit 64 bit gleich gross ist. Dies röhrt daher, dass die Bibliothek für die 32 bit Rechnung die gleiche Genauigkeit verwendet wie bei 64 bit. Selbst die Präzision von 256 bit reicht nicht aus, um e^{-100} zu berechnen. Erst im letzten Resultat stimmt das Resultat mit 16 signifikanten Stellen. Eine genauere Untersuchung zeigt, dass die volle Genauigkeit bei 384 bits erreicht wird.

GNU Multiple Precision Floatingpoint Reliable Library

Die GNU Multiple Precision Floatingpoint Reliable Library versucht, die Defizite der GMP zu kompensieren. Das nur wenig abweichende Listing 1.3 zeigt die Implementation.

Mit der Funktion `mpfr_set_default_prec(int bits)` in Zeile 44 wird nicht die minimale sondern die exakte Anzahl Bits festgelegt, mit der gerechnet werden soll. Bei jeder Operation, bei der gerundet werden muss, kann spezifiziert werden, in welche Richtung die Rundung vorgenommen werden soll, dies ist die Bedeutung der Konstanten `MPFR_RNDN`. Schliesslich ist die Implementation konform mit dem IEEE 754 Standard, so dass bei Verwendung dieser Bibliothek garantiert ist, dass die Resultate nicht von der Plattform abhängen.

```

1 #include <mpfr.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <math.h>
5
6 double mpfexp(double x) {
7     mpfr_t X, P, S;
8     /* Variablen initialisieren */
9     mpfr_init(X);
10    mpfr_init(P);
11    mpfr_init(S);
12    mpfr_set_d(X, x, MPFR_RNDN);
13    mpfr_set_d(P, 1., MPFR_RNDN);
14    mpfr_set_d(S, 1., MPFR_RNDN);
15
16    /* Hilfsvariablen fuer Zwischenresultate */
17    mpfr_t R, Q;
18    mpfr_init(R);
19    mpfr_init(Q);
20
21    for (int i = 1; i < 1000; i++) {
22        /* P = P * X / i */
23        mpfr_mul(Q, P, X, MPFR_RNDN);
24        mpfr_div_ui(R, Q, i, MPFR_RNDN);
25        mpfr_set(P, R, MPFR_RNDN);
26        /* S = S + P */
27        mpfr_add(Q, S, R, MPFR_RNDN);
28        mpfr_set(S, Q, MPFR_RNDN);
29    }
30
31    double result = mpfr_get_d(S, MPFR_RNDN);
32
33    mpfr_clear(X);
34    mpfr_clear(P);
35    mpfr_clear(S);
36    mpfr_clear(R);
37    mpfr_clear(Q);
38
39    return result;
40}
41
42 void experiment(int bits) {
43     /* Setze exakte Genauigkeit der Variablen */
44     mpfr_set_default_prec(bits);
45     double x = -100;
46     double y = mpfexp(x);
47     double z = exp(x);
48
49     printf("bits:      %d\n", bits);
50     printf("GMP:      %.20g\n", y);
51     printf("Prozessor: %.20g\n", z);
52     printf("Fehler:    %.20g\n\n", y - z);
53}

```

Listing 1.3: C-Programm zur Berechnung von e^x mit Hilfe der Taylor-Reihe, implementiert mit MPFR

In Tabelle 1.5 sind die Resultate zusammengestellt. Es fällt sofort auf, dass 32 bit und 64 bit verschiedene Resultate geben. Ab Bitlänge 328 ist in den gezeigten Stellen kein Unterschied mehr zwischen der Maschinenimplementation und der MPFR-Implementation erkennbar, trotzdem verbleibt eine Abweichung, der in der letzten Spalte dargestellt wird. Sie zeigt auch, dass in der MPFR-Implementation jedes einzelne Bit einen Unterschied macht. Die MPFR-Bibliothek kann daher auch als Labor für Experimente über die Abhängigkeit der Fehler von der Genauigkeit der Arithmetik dienen.

Die Programmierung mit diesen Bibliotheken ist wegen der vielen Funktionsaufrufe eher etwas mühsam. Moderne C++-Wrapper ermöglichen dank Operatorüberladung eine intuitivere Notation.

1.2 Numerische Effekte

Die Unzulänglichkeiten der in Computern verwendeten Zahlensysteme haben zwei Effekte zur Folge, denen bei der Konzeption eines numerischen Lösungsverfahrens Rechnung getragen werden muss.

1.2.1 Auslöschung

Auslöschung tritt auf, wenn die Differenz zweier ähnlich grosser Zahlen gebildet wird. Als Beispiel betrachten wir die beiden Zahlen $a = \pi$ und $b = \sqrt{10}$. Berechnen wir deren Differenz in Octave, erhalten wir:

```
octave:1> a=pi
a = 3.14159265358979
octave:2> b=sqrt(10)
b = 3.16227766016838
octave:3> a-b
ans = -0.0206850065785864
```

Die ersten zwei Stellen von a und b stimmen überein. Octave zeigt sowohl von a als auch von b 15 signifikante Stellen an. Ein Vergleich mit einer Berechnung mit noch mehr Stellen zeigt, dass diese 15 Stellen auch zuverlässig sind. Für die Differenz zeigt Octave ebenfalls 15 Stellen an, doch die letzte Stelle ist falsch, wie zum Beispiel die Berechnung mit 20 Stellen Genauigkeit zeigt⁴

```
scale=20
a=4*a(1)
a
3.14159265358979323844
b=sqrt(10)
b
3.16227766016837933199
a-b
-.02068500657858609355
```

⁴Diese Berechnung wurde mit dem Linux-Kommandozeilenprogramm bc durchgeführt, welches mit einstellbarer Festkommapräzision von tausenden von Stellen rechnen kann. Es ist Teil jeder Linux-Distribution.

Schreiben wir die Subtraktion in der tabellarischen Form

$$\begin{array}{r} 3.16227766016838 \\ -3.14159265358979 \\ \hline 0.02068500657859 \end{array}$$

wird erkennbar, dass nur 13 Stellen der Differenz tatsächlich bekannt sind.

Die Rechnung wird in Binärdarstellung etwas klarer. In der folgenden Tabelle sind die Werte in der mittleren Spalte in binärer Gleitkommadarstellung gezeigt, Vorzeichen, Exponent und Mantisse sind zur Verdeutlichung durch ein Leerzeichen getrennt. Zu Beginn der Mantisse muss man sich eine implizite 1 denken, die nicht gespeichert wird. In der dritten Spalte werden die gleichen Zahlen als binäre Festkommawerten geschrieben. Zur Berechnung der Differenz muss der Prozessor die Mantissen ja zunächst so schieben, dass sie den gleichen Exponenten bekommen, der Prozessor berechnet also die Differenz implizit in einer Festkommadarstellung.

	Gleikommawert	Festkommawert
$\sqrt{10}$	0 10000000 10010100110001011000010	11.0010100110001011000010
π	0 10000000 10010010000111111011011	11.0010010000111111011011
$\sqrt{10} - \pi$	0 01111001 01010010111001110000000	0.0000010101001011100111

Man kann gut erkennen, dass die Differenz nur 17 signifikante Stellen (rot hervorgehoben) hat. Bei der nachfolgenden Darstellung als Gleitkommazahl werden 7 Nullen hinzugefügt, die aber nichts mit der tatsächlichen Differenz $\sqrt{10} - \pi$ zu tun haben. Aus zwei Zahlenwerten mit einer Genauigkeit von 24 Binärstellen ist ein Wert mit einer Genauigkeit von nur 17 Binärstellen geworden. Es sind 7 Binärstellen Genauigkeit ausgelöscht worden.

Beispiel. Sei X ein standardnormalverteilte Zufallsvariable, es soll die Wahrscheinlichkeit dafür berechnet werden, dass $a \leq X \leq b$ ist. In der Wahrscheinlichkeitsrechnung lernt man, dass man dazu die Verteilungsfunktion

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-s^2/2} ds = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^x e^{-s^2/2} ds$$

der Standardnormalverteilung verwenden kann:

$$P(a \leq X \leq b) = \Phi(b) - \Phi(a).$$

Die Funktion $\Phi(x)$ wird in vielen Bibliotheken nicht direkt zur Verfügung gestellt, oft ist nur die sogenannte *Fehlerfunktion*

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

verfügbar. Die Variablentransformation $t = s/\sqrt{2}$ oder $s = \sqrt{2}t$ macht aus dem Integral für $\Phi(x)$ den Ausdruck

$$\begin{aligned} \Phi(x) &= \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^x e^{-s^2/2} ds = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^{\sqrt{2}x} e^{-t^2} \sqrt{2} dt = \frac{1}{2} + \frac{2}{\pi} \int_0^{\sqrt{2}x} e^{-t^2} dt \\ &= \frac{1}{2} \left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right) \right), \end{aligned}$$

Datentyp	Rechnung mit $\text{erf}(x)$	Rechnung mit $\text{erfc}(x)$
<code>float</code>	0.000000	$6.271826 \cdot 10^{-17}$
<code>double</code>	$1.110223 \cdot 10^{-16}$	$6.271826 \cdot 10^{-17}$
<code>long</code>	$6.272110 \cdot 10^{-17}$	$6.271826 \cdot 10^{-17}$

Tabelle 1.6: Berechnung der Wahrscheinlichkeit $P(4.18 \leq X \leq 5.18)$ einer standardnormalverteilten Zufallsvariable mit Hilfe der Bibliotheksfunktionen $\text{erf}(x)$ und $\text{erfc}(x)$. Starke Auslöschung macht die Berechnung mit $\text{erf}(x)$ unbrauchbar.

die Fehlerfunktion $\text{erf}(x)$ kann also zur Berechnung der gesuchten Wahrscheinlichkeit verwendet werden.

In der C-Bibliothek stehen Funktionen zur Berechnung von $\sqrt{2}$ und $\text{erf}(x)$ für alle zur Verfügung stehenden Datentypen bereit. Die Tabelle 1.6 zeigt die Resultate⁵ für $a = 4.18$ und $b = a + 1$.

Da die Werte von $\text{erf}(\sqrt{2}a)$ und $\text{erf}(\sqrt{2}b)$ fast gleich gross sind, findet starke Auslöschung statt. Beim Datentyp `float` ist überhaupt kein Unterschied mehr feststellbar.

Um dieses Problem in den Griff zu bekommen, stellt die C-Bibliothek zusätzlich die sogenannte *komplementäre Fehlerfunktion*

$$\text{erfc}(x) = 1 - \text{erf}(x) \quad \Rightarrow \quad \text{erf}(x) = 1 - \text{erfc}(x)$$

zur Verfügung. Damit kann die Wahrscheinlichkeit natürlich auch berechnet werden:

$$P(a \leq X \leq b) = \text{erf}(\sqrt{2}b) - \text{erf}(\sqrt{2}a) = (1 - \text{erfc}(\sqrt{2}b)) - (1 - \text{erfc}(\sqrt{2}a)) = \text{erfc}(\sqrt{2}a) - \text{erfc}(\sqrt{2}b).$$

Für grosse Werte von x streben die Werte dieser Funktion gegen 0, es kann also nicht mehr passieren, dass man einen kleinen Wert zu finden versucht, indem man zwei vergleichsweise grosse Zahlen voneinander subtrahiert. In der dritten Spalte von Tabelle 1.6 sind die Resultate der Berechnung mit Hilfe von $\text{erfc}(x)$ gezeigt. Der Auslöschungseffekt ist vollständig verschwunden. Man kann sogar ablesen, dass die Verwendung des Datentyps `long double` dem Problem der Auslöschung ebenfalls nicht begegnen konnte. Der mit $\text{erf}(x)$ berechnete Wert hat selbst bei Verwendung dieses längsten verfügbaren Typs nur drei korrekte Dezimalstellen. \circ

1.2.2 Verschmierung

Auslöschung kann nicht nur auftreten, wenn zwei fast gleich grosse Zahlen subtrahiert werden. Sie kann in etwas weniger offensichtlicher Form stattfinden, wenn bei der Summation einer Reihe im Vergleich zum Resultat grosse Zwischenresultate entstehen. Diesen Verlust an Genauigkeit infolge grosser Zwischenresultate wird *Verschmierung* genannt.

Beispiel: Exponentialreihe e^x

Die Taylor-Reihe

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

⁵Diese Resultate wurden mit dem Programm `normal1.cpp` Im Verzeichnis `buch/chapters/experimente/ausloesung` von [4] berechnet.

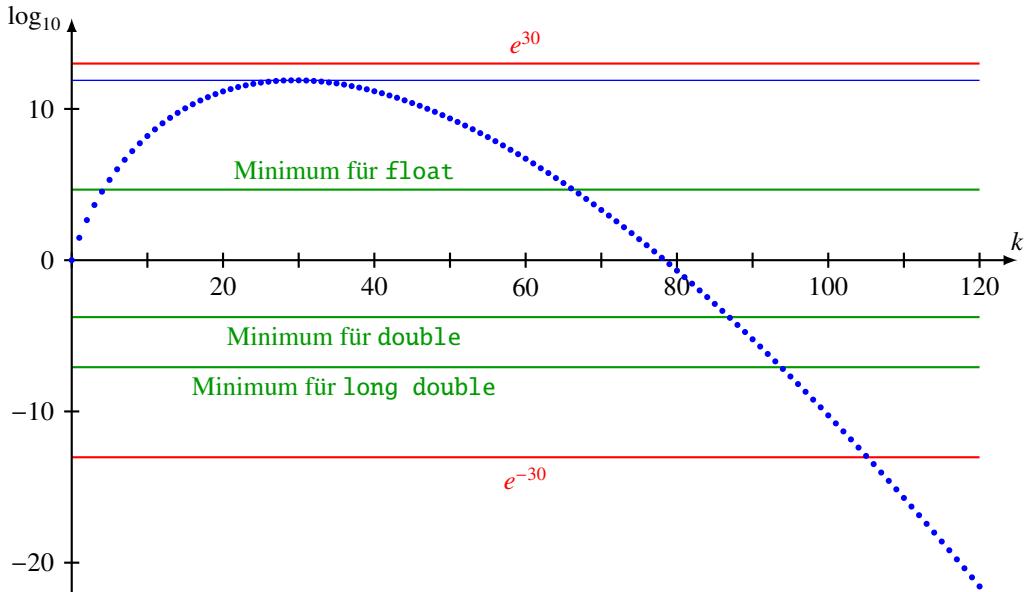


Abbildung 1.1: Verschmierung bei der Berechnung von e^{-30} und e^{30} mit der Exponentialreihe. Auf der vertikalen Achse ist der Zehnerlogarithmus der verschiedenen Größen abgetragen. Die beiden Werte e^{30} und e^{-30} sind als rote Linien am oberen und unteren Rand eingetragen. Blau ist der absolute Betrag des Terms $s_k = x^k/k!$ in der Exponentialreihe. Die grünen Linien zeigen den kleinsten Unterschied, der zwischen zwei Termen möglich ist, die die Größe des grössten Summanden in der Exponentialreihe haben.

der Exponentialfunktion ist sehr gut geeignet, Werte von e^x für positive x zu berechnen. Da der Nenner $k!$ exponentiell schnell anwächst, werden späte Terme in der Reihe sehr schnell vernachlässigbar klein.

Für negative Exponenten erwartet man ein sehr kleines Resultat. Die Terme der Taylor-Reihe haben alternierende Vorzeichen und werden zwischenzeitlich sehr gross. Im Laufe der Rechnung müssen sich also grosse Terme wieder wegheben. Dies ist in Abbildung 1.1 illustriert. Dort sind die Werte e^{30} und $e^{-30} = 1/e^{30}$ auf einer logarithmischen Skala vertikal als rote Linien eingezeichnet. Die einzelnen Summanden der Reihe sind in blau dargestellt.

Die grünen Linien zeigen, welche Genauigkeit mit verschiedenen Datentypen überhaupt noch möglich ist. Der float-Typ hat eine Mantisse von 24 bit, eine Zahl m ist daher nur unterschiedbar

Datentyp	e^{-30}
float	$-7.2959523438 \cdot 10^{-4}$
double	$6.1030424789 \cdot 10^{-6}$
long double	$-1.2489259417 \cdot 10^{-8}$
exakt	$9.3576229688 \cdot 10^{-14}$

Tabelle 1.7: Resultate der Summe der Taylor-Reihe ausgeführt mit verschiedenen Datentypen zeigt, dass die Genauigkeit der Resultate infolge Verschmierung vollständig ausgelöscht worden ist.

von $m(1 + \varepsilon)$, wenn $\varepsilon > 2^{-24}$. Dies entspricht $24 \cdot \log_{10}(2) = 7.22$ Dezimalstellen. Die grüne Linie für den `float`-Typ ist daher 7.22 unter dem Maximum der Terme Exponentialreihe eingetragen.

Beim `double`-Typen ist die Mantisse 52 bit lang, beim `long double` sind es 63 bit. Doch selbst beim `long double` ist die Verschmierung vollständig, das Resultat für e^{-30} hat nichts mit der Realität zu tun. Im Gegenteil, sie zeigen eher an, wie gross der verwendete Datentyp ist. Die grünen Linien in Abbildung 1.1 befinden sich ungefähr dort, wo die gefundenen Werte eingetragen werden müssten.

In Tabelle 1.7 sind die Resultate der Summierung der Reihe mit verschiedenen Datentypen zusammengestellt. Alle Summen weichen um mehrere Größenordnungen vom exakten Wert ab.

Summationsreihenfolge

Das Beispiel zur Exponentialfunktion hat gezeigt, wie Verschmierung die Präzision des Resultates vollständig ausgelöscht hat. Ein weniger dramatischer Genauigkeitsverlust kann schon bei wenigen Summanden stark unterschiedlicher Größenordnung auftreten, wie das folgende Beispiel aus [2] zeigt.

Es soll die Summe $10^4 + \pi + e$ berechnet werden. Die Rundung kann entweder ganz am Schluss oder nach jeder einzelnen Addition erfolgen, so wie es der IEEE 754-Standard verlangt. Es ergibt sich:

exakte Rechnung	nacheinander
$\begin{array}{r} 10000.0 \\ + \quad 3.14159 \\ \hline \end{array}$	$\begin{array}{r} 10000.0 \\ + \quad 3.14159 \\ \hline 10003.14159 \\ 10003.1 \\ + \quad 2.71828 \\ \hline 10005.81828 \\ 10005.8 \end{array}$
$\begin{array}{r} 10000.0 \\ + \quad 3.14159 \\ \hline 10003.14159 \\ 10003.1 \\ + \quad 2.71828 \\ \hline 10005.81828 \\ 10005.9 \end{array}$	

Links ist die exakte Addition mit Rundung auf sechs signifikante Stellen am Schluss durchgeführt, rechts steht die Addition mit Rundung nach jeder Operation. Bereits nach drei Termen zeigt sich ein Unterschied in der letzten Stelle, der Verschmierung zuzuschreiben ist.

Die unvermeidliche Rundung nach jeder Addition beeinträchtigt die Berechnung jeder unendlichen Summe

$$s_n = \sum_{k=0}^n a_k.$$

Die naive Berechnung in der Reihenfolge aufsteigender k wird in den meisten Fällen die Reihenfolge kleiner werdender Terme sein. Für genügend grosses k werden die zusätzlichen Terme von ähnlicher Größenordnung sein wie die Rundungsfehler, die sich bei den ersten Termen bereits gebildet haben.

Die Reihenfolge abnehmender k bietet die Chance, dass Rundungsfehler, die bei der Addition kleiner Terme entstanden sind, bei der Addition grösserer Terme mit kleinerem k verschwinden im neuen Rundungsfehler verschwinden. Dies geschieht allerdings nur, wenn die Beträge der Terme schneller wachsen als die Rundungsfehler in den kleinen Termen. Trotzdem ist es eine gute Idee, eine grosse Summe beginnend bei den kleinsten Termen zu summieren.

```

1 || double s = 0;
2 || double c = 0;
3 || for (int i = 1; i <= n; i++) {
4 ||     double y = a(i);
5 ||     y = y - c;
6 ||     double t = s + y;
7 ||     c = (t - s) - y;
8 ||     s = t;
9 ||
}

```

Listing 1.4: Kahan-Summations-Algorithmus zur Vermeidung sich akkumulierender Rundungsfehler.

Methode	float	long double
vorwärts	15.40368270	18.99789641385390515
rückwärts	18.80791855	18.99789641385389798
Kahan-Summation	18.99789619	18.99789641385389827

Tabelle 1.8: Berechnung der Summe h_n der harmonischen Reihe für $h = 10^8$ mit verschiedenen Methoden.

Kahan-Summationsalgorithmus

Der Kahan-Summationsalgorithmus versucht, über die sich ansammelnden Rundungsfehler in den niedrigerwertigen Stellen in einer separaten Variable Buch zu führen. Um das Prinzip zu verstehen, sei also s_{n-1} eine Teilsumme und a_n der nächste Term der zur Summe hinzugefügt werden soll. Die Summe $s_n = s_{n-1} + a_n$ wird natürlich gerundet. Die Differenz $c_n = (s_n - s_{n-1}) - a_n$, genau in der Reihenfolge der Klammern ausgewertet, enthält die durch Rundung verlorenen Stellen.

Der nächste Summand a_{n+1} dürfte kleiner als die Summe s_n , man kann ihn daher um den Betrag c_n korrigieren und damit verlorene Genauigkeit wiederherstellen. Wir addieren daher $\bar{a}_{n+1} = a_{n+1} - c_n$ anstelle von a_{n+1} und erhalten die Summe $s_{n+1} = s_n + \bar{a}_{n+1}$. Natürlich können wir auch hier wieder den Fehler als $c_{n+1} = (s_{n+1} - s_n) - \bar{a}_{n+1}$ berechnen. Damit erhalten wir den Algorithmus in Listing 1.4. Er verwendet die Funktion $a(i)$, welche den Term mit Index i der Summe berechnet. Die Kahan-Summation vervierfacht die Anzahl der Additionen hat aber das Potential die Akkumulation von Rundungsfehlern fast vollständig zu eliminieren.

Beispiel. Die harmonische Reihe

$$h_n = \sum_{k=1}^n \frac{1}{k}$$

kann gegen unten abgeschätzt werden mit Hilfe des Integrals

$$\sum_{k=1}^n \frac{1}{k} > \int_1^{n+1} \frac{1}{x} dx = \log(n+1),$$

welches divergiert. Die Folge h_n divergiert also, aber sehr langsam. Beim numerischen Aufsummieren gibt es ausgiebig Gelegenheit für Genauigkeitsverlust. Am akutesten ist der Verlust, wenn man die Summe mit dem grössten Term $k = 1$ beginnt, eine kleine Verbesserung ist von der umgekehrten Reihenfolge zu erwarten.

Die Resultate der Berechnung von h_{10^8} mit dem `float` Datentyp sind in Tabelle 1.8 zusammengestellt. Die naheliegende Summierung beginnend beim ersten Term führt auf ein unbrauchbares Resultat. Beginnt man mit dem kleinsten Term, könnten die Rundungsfehler durch die späteren größeren Terme übertönt werden, leider divergiert die Summe so langsam, dass damit nicht alle Fehler zum Verschwinden gebracht werden können. Es sind nur gerade die Stellen vor dem Komma korrekt. Die Kahan-Summation vermeidet dieses Problem vollständig.

Zum Vergleich ist in der Spalte rechts in Tabelle 1.8 die Summe berechnet mit dem Typ `long double` dargestellt. Dies zeigt, dass alle signifikanten Stellen der Kahan-Summation korrekt sind.



1.3 Iteration

Die meisten numerischen Problemlösungen mit einem Computer nutzen deren Fähigkeit aus, dieselbe Rechnung immer wieder zu wiederholen, bis zum Beispiel die gewünschte Genauigkeit erzielt ist. Es lohnt sich daher ganz unabhängig irgendwelchen Einschränkungen der Computer-Hardware zu überlegen, was passiert, wenn man eine Funktion $f: \mathbb{R} \rightarrow \mathbb{R}$ immer wieder auf ihren eigenen Output anwendet, wie das zum Beispiel der Code in einer Schleife bei jedem Durchlauf macht.

In diesem Abschnitt ist also

$$f: \mathbb{R} \rightarrow \mathbb{R} : x \mapsto f(x)$$

eine differenzierbare Funktion. Mit einem gegebenen Startwert $x_0 \in \mathbb{R}$ lässt sich durch wiederholte Anwendung von f die Folge

$$x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), x_4 = f(x_3), \dots$$

konstruieren.

Ist der Punkt x^* ein *Fixpunkt* der Funktion $f(x)$, ist also $f(x^*) = x^*$, dann ist die mit $x_0 = x^*$ gebildete Iterationsfolge konstant. Es stellt sich damit automatisch die Frage, was mit einem von x^* abweichenden Startwert passiert. Entfernen sich die Werte x_k von x^* oder konvergiert die Folge am Ende gegen x^* ?

1.3.1 Beispiele

Wir illustrieren die verschiedenen Situationen, die beim Iterieren der Funktion f auftreten können an einigen Beispielen.

Beispiel. Wir betrachten die Funktion

$$f: \mathbb{R} \rightarrow \mathbb{R} : x \mapsto \sqrt{x+2}$$

Die Iterationsfolge ausgehend vom Startwert $x_0 = 0$ ist in Tabelle 1.9 dargestellt.

Die Werte konvergieren offenbar gegen den Wert 2. In der dritten Spalte steht die Abweichung δ_k des k -ten Folgengliedes vom Grenzwert 2. Mit jeder Iteration wird der Fehler um den Faktor 4 kleiner, wie die vierte Spalte zeigt, in der Quotient aufeinanderfolgender Fehler berechnet ist.

Dieses Verhalten des Fehlers kann man auch analytisch verstehen. Nehmen wir an, dass $x_n = 2 + \delta_n$ und versuchen wir x_{n+1} zu berechnen. Indem wir die Funktion $f(x)$ im Punkt $x = 2$ mit Hilfe der Ableitung linear approximieren, erhalten wir wegen

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

k	x_k	$\delta_k = 2 - x_k$	δ_{k-1}/δ_k
0	0.0000000000000000	2.0000000000000000	
2	1.4142135623730951	0.5857864376269049	3.4142
3	<u>1.8477590650225735</u>	0.1522409349774265	3.8478
4	<u>1.9615705608064609</u>	0.0384294391935391	3.9616
5	<u>1.9903694533443939</u>	0.0096305466556061	3.9904
6	<u>1.9975909124103448</u>	0.0024090875896552	3.9976
7	<u>1.9993976373924085</u>	0.0006023626075915	3.9994
8	<u>1.9998494036782890</u>	0.0001505963217110	3.9998
9	<u>1.9999623505652022</u>	0.0000376494347978	4.0000
10	<u>1.9999905876191524</u>	0.0000094123808476	4.0000
11	<u>1.9999976469034038</u>	0.0000023530965962	4.0000
12	<u>1.9999994117257645</u>	0.0000005882742355	4.0000
13	<u>1.9999998529314358</u>	0.0000001470685642	4.0000
14	<u>1.9999999632328584</u>	0.0000000367671416	4.0000
15	<u>1.9999999908082147</u>	0.0000000091917853	4.0000
16	<u>1.9999999977020537</u>	0.0000000022979463	4.0000
17	<u>1.9999999994255133</u>	0.0000000005744867	4.0000
18	<u>1.9999999998563782</u>	0.0000000001436218	4.0000
19	<u>1.999999999640945</u>	0.000000000359055	4.0000
20	<u>1.9999999999910236</u>	0.000000000089764	4.0000
21	<u>1.9999999999977558</u>	0.00000000022442	3.9998
22	<u>1.999999999994389</u>	0.00000000005611	3.9996
23	<u>1.999999999998597</u>	0.0000000001403	3.9984
24	<u>1.999999999999649</u>	0.0000000000351	4.0000
25	<u>1.99999999999911</u>	0.0000000000089	3.9500
26	<u>1.999999999999978</u>	0.0000000000022	4.0000
27	<u>1.999999999999993</u>	0.0000000000007	3.3333
28	<u>1.999999999999998</u>	0.0000000000002	3.0000
29	<u>2.0000000000000000</u>	0.0000000000000000	
30	<u>2.0000000000000000</u>	0.0000000000000000	

Tabelle 1.9: Iterationsfolge für die Funktion $f(x) = \sqrt{x+2}$ ausgehend vom Startwert $x_0 = 0$.

den Wert

$$x_{n+1} = f(x_n) = f(2 + \delta_n) \approx f(2) + f'(2) \cdot \delta_n = 2 + \underbrace{\frac{1}{4}}_{\approx \delta_{n+1}} \cdot \delta_n$$

für x_{n+1} . Der Fehler von x_{n+1} ist also $\delta_{n+1} \approx \frac{1}{4}\delta_n$. In jedem Iterationsschritt gewinnen wir daher etwa 2 bit Genauigkeit. Für die 52 bit Mantisse des double Typs brauchen wir also etwa 26 Iterationen. \circlearrowright

Beispiel. Wir betrachten die Funktion

$$f(x) = 3x(1 - x).$$

Sie hat zwei Fixpunkte, die man durch Lösen der quadratischen Gleichung

$$f(x^*) = x^* \Rightarrow x^* = -3x^{*2} + 3x^* \Rightarrow 3x^{*2} - 2x^* = 3x^*(x^* - \frac{2}{3}) = 0 \Rightarrow x^* = \begin{cases} 0 \\ \frac{2}{3} \end{cases}$$

findet.

Für einen Startwert x_0 nahe des Fixpunktes $x^* = 0$ gilt

$$f(\delta) = 3\delta(1 - \delta) = 3\delta - 3\delta^2.$$

Für kleine Werte von δ kann man den quadratischen Term vernachlässigen und sieht, dass der Fehler durch die Iteration verdreifacht wird. Konvergenz zu diesem Fixpunkt ist also nicht möglich.

Für den Fixpunkt $x^* = \frac{2}{3}$ finden wir

$$f(\frac{2}{3} + \delta) = 3(\frac{2}{3} + \delta)(\frac{1}{3} - \delta) = \frac{(2 + 3\delta)(1 - 3\delta)}{3} = \frac{2 - 3\delta + 9\delta^2}{3} = \frac{2}{3} - \delta - 3\delta^2 \quad (1.2)$$

Der Fehler δ wird zu -9δ , er ändert also sein Vorzeichen. Ist $\delta > 0$ wird der Fehlerbetrag wird nur um den Faktor $1 - 9\delta < 1$ reduziert. Ist aber $\delta < 0$, dann ist $1 - 9\delta > 0$, der Fehlerbetrag wird wieder vergrößert.

Sei jetzt $\delta > 0$, wir wollen den Fehler nach zwei Iterationsschritten berechnen. Nach dem ersten ist der Fehler $\delta(1 - 9\delta)$, nach dem zweiten

$$\delta(1 - 9\delta)(1 - \delta(1 - 9\delta)) = \delta(1 - 18\delta + 162\delta^2 - 729\delta^3).$$

Für kleines δ können die Terme höherer als erster Ordnung vernachlässigt werden und man kann schliessen, dass der Fehler nach zwei Iterationen tatsächlich um den Faktor $(1 - 18\delta)$ kleiner geworden ist.

Wir möchten wissen, wieviele Iterationsschritte nötig sind, um eine bestimmte Genauigkeit zu erreichen. Wir möchten also den Fehler δ_{2n} vorgeben und das zugehörige n bestimmen. Wie wir soeben berechnet haben, nimmt der Betrag des Fehlers zwischen den Schritten k und $k + 2$ gemäss

$$\delta_{k+2} = \delta_k(1 - 18\delta_k)$$

um den Faktor $(1 - 18\delta_k)$ ab. Es ist übersichtlicher, mit dem Logarithmus des Fehlers zu rechnen. Dieser verändert sich gemäss

$$\log \delta_{k+2} = \log \delta_k + \log(1 - 18\delta_k) \approx \log \delta_k - 18\delta_k,$$

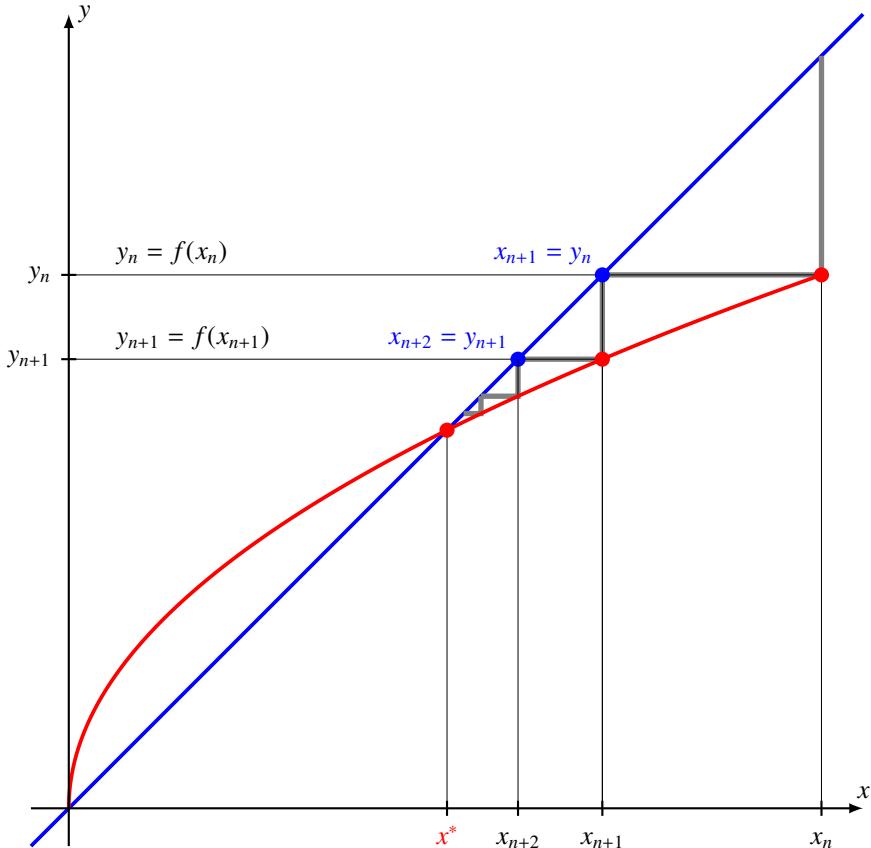


Abbildung 1.2: Ein Fixpunkt x^* der Funktion $f(x)$ manifestiert sich als Schnittpunkt des Graphen $y = f(x)$ mit der 45° -Geraden. Die Iterationsfolge ausgehend von einem Startwert x_0 wird als Treppe zwischen dem Graphen von f und der 45° -Geraden sichtbar.

wobei wir für den zweiten Term die lineare Approximation aus der Taylor-Reihe

$$\log(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots$$

von $\log x$ an der Stelle $x = 1$ verwendet haben. Zwischen δ_0 und δ_{2n} bedeutet dies

$$\log \delta_{2n} = \log \delta_0 - 18 \sum_{k=0}^{n-1} \delta_{2k} \quad \Rightarrow \quad \frac{1}{18} \log \frac{\delta_0}{\delta_{2n}} = \sum_{k=0}^{n-1} \delta_{2k}.$$

Da der Fehler immer kleiner wird, kann man für eine erste grobe Abschätzung der Summe auf der rechten Seite die größten und kleinsten Terme verwenden, um die Summe nach unten und oben durch

$$n\delta_{2n-2} \leq \sum_{k=0}^{n-1} \delta_{2k} \leq n\delta_0$$

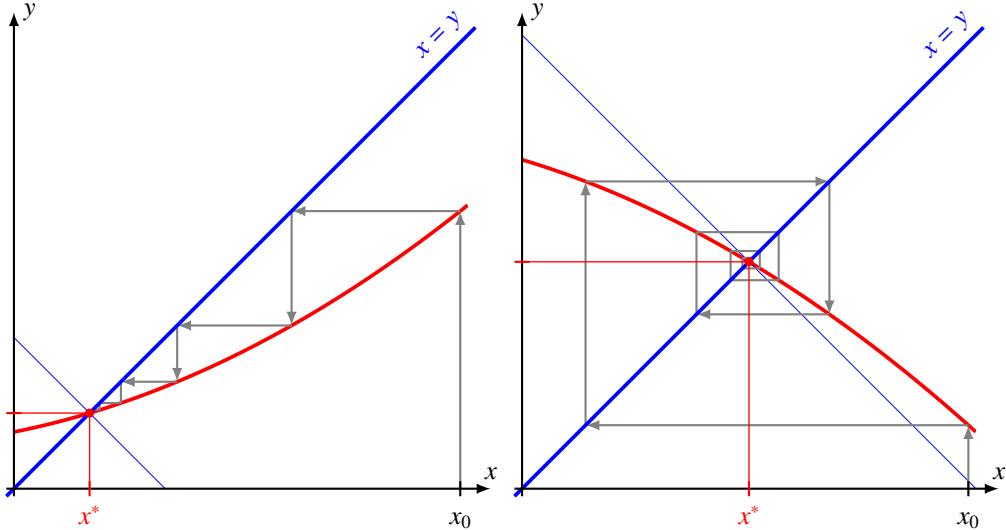


Abbildung 1.3: Die Fixpunktiteration $x_{n+1} = f(x_n)$ konvergiert gegen den Fixpunkt x^* falls $|f'(x^*)| < 1$ mit mindestens linearer Konvergenzgeschwindigkeit.

abzuschätzen. Einsetzen ergibt

$$n\delta_{2n-2} \leq \frac{1}{18} \log \frac{\delta_0}{\delta_{2n}} \leq n\delta_0 \quad \Rightarrow \quad \frac{1}{18\delta_0} \log \frac{\delta_0}{\delta_{2n}} \leq n \leq \frac{1}{18\delta_{2n-2}} \log \frac{\delta_0}{\delta_{2n}} < \frac{1}{18\delta_{2n}} \log \frac{\delta_0}{\delta_{2n}}.$$

Eine Verbesserung um zwei Stellen von $\delta_0 = 0.01$ auf $\delta_{2n} = 0.0001$ braucht also zwischen 25 und 2558 Iterationen. \circ

1.3.2 Graphische Analyse

Ein Fixpunkt der Funktion $f(x)$ manifestiert sich in einem Schnittpunkte des Graphen von f mit der 45° -Geraden $y = x$ wie in Abbildung 1.2. Die Iterationsfolge x_n kann graphisch wie folgt dargestellt werden. Ausgehend vom Wert x_n auf der x -Achse folgt man der Vertikalen, bis man auf den Graphen der Funktion f trifft, dies liefert den Wert $y_n = f(x_n)$. Dieser soll jetzt als neuer x -Wert verwendet werden. Dazu folgt man der Horizontalen bis zur 45° -Geraden, die x -Koordinate des Schnittpunktes ist x_{n+1} . So entsteht Treppenlinie in Abbildung 1.2.

1.3.3 Konvergenzbedingung

Aus der graphischen Analyse von Abschnitt 1.3.2 kann man jetzt leicht Kriterien ableiten, wann die Iterationsfolge konvergent ist. Die Situation in Abbildung 1.2 tritt zum Beispiel immer ein, wenn in einer Umgebung von x^* der Graph der Funktion f zwischen der horizontalen und der 45° -Geraden verläuft, oder anders ausgedrückt, wenn

$$\begin{aligned} x^* < f(x) < x & \quad \text{für } x > x^* \text{ und} \\ x^* > f(x) < x & \quad \text{für } x < x^* \end{aligned}$$

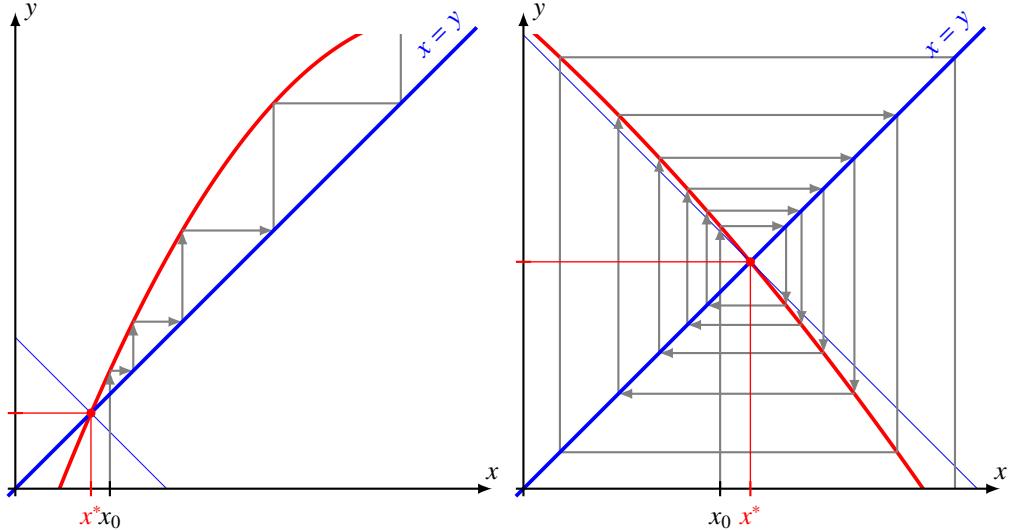


Abbildung 1.4: Die Fixpunktiteration $x_{n+1} = f(x_n)$ mit Fixpunkt x^* divergiert für $|f'(x^*)| > 1$.

gilt.

Ist dagegen der Graph von f in einer Umgebung von x^* steiler als die 45° -Gerade, dann gilt

$$f(x) \begin{cases} > x & \text{für } x > x^* \\ < x & \text{für } x < x^*. \end{cases} \quad (1.3)$$

Es folgt dann

$$x_{n+1} = f(x_n) \begin{cases} > x_n & \text{für } x_n > x^* \\ < x_n & \text{für } x_n < x^*. \end{cases}$$

In beiden Fällen ist x_{n+1} weiter vom Fixpunkt x^* entfernt als x_n , die Folge kann also nicht konvergieren. Die beiden Situationen sind in den Abbildungen 1.3 und 1.4 dargestellt.

Allerdings ist die Bedingung 1.3 noch etwas zu speziell. Der Funktionswert $f(x_n)$ darf durchaus auch kleiner als x^* werden, wenn nur der Betrag der Differenz zu x^* abnimmt, wenn also gilt

$$|x^* - f(x_n)| < |x^* - x_n| \quad (1.4)$$

Diese Bedingung ist genügend nahe bei x^* erfüllt, wenn $|f'(x^*)| < 1$ ist. Andererseits liegt Divergenz vor, wenn

$$|x^* - f(x_n)| > |x^* - x_n|,$$

was nahe bei x^* erfüllt ist, wenn $|f'(x^*)| > 1$ gilt. Wir fassen diese Resultate im folgenden Satz zusammen.

Satz 1.1. Die Iterationsfolge $x_{n+1} = f(x_n)$ der Funktion $f(x)$ mit Fixpunkt $x^* = f(x^*)$ konvergiert für einen Startwert x_0 nahe genug bei x^* , wenn $|f'(x^*)| < 1$, sie divergiert, wenn $|f'(x^*)| > 1$.

1.3.4 Die logistische Gleichung

Die logistische Gleichung ist

$$f_\lambda(x) = \lambda x(1 - x)$$

mit dem positiven Parameter λ . Im Beispiel auf Seite 28 haben wir diese Funktion bereits einmal angetroffen mit dem Parameterwert $\lambda = 3$. Eine detailliertere Untersuchung des Konvergenzverhaltens der Iterationen dieser Funktionen ist im Kapitel 8 zu finden.

Der Graph von $f_\lambda(x)$ ist eine Parabel, die die x -Achse in den Punkten $(0, 0)$ und $(1, 0)$ schneidet. Das Maximum wird bei $x = \frac{1}{2}$ angenommen und ist $f_\lambda(\frac{1}{2}) = \lambda/4$. Die Funktion bildet also das Intervall $[0, 1]$ wieder in das selbe Intervall ab, wenn $0 \leq \lambda \leq 4$.

Wir berechnen die Fixpunkte von $f_\lambda(x)$ im Intervall $[0, 1]$, also die Lösungen der quadratischen Gleichung

$$x^* = \lambda x^*(1 - x^*) \Rightarrow \lambda x^{*2} + (1 - \lambda)x^* = x^*(\lambda x^* + 1 - \lambda) = 0 \Rightarrow x^* = \begin{cases} 0 \\ \frac{\lambda - 1}{\lambda}. \end{cases}$$

Für $\lambda < 1$ gibt es keinen weiteren Fixpunkt im Intervall $[0, 1]$, für $\lambda > 1$ ist $(\lambda - 1)/\lambda < 1$ immer im Intervall.

Zur Beurteilung der Konvergenz der Iterationsfolge müssen wir die Ableitung von f_λ im Fixpunkt berechnen. Die Ableitung ist $f'_\lambda(x) = \lambda(1 - 2x)$, also gilt in den Fixpunkten

$$\begin{aligned} f'_\lambda(0) &= \lambda \\ f'_\lambda\left(\frac{\lambda - 1}{\lambda}\right) &= \lambda\left(1 - 2\left(\frac{\lambda - 1}{\lambda}\right)\right) = \lambda - 2(\lambda - 1) = 2 - \lambda. \end{aligned}$$

Daraus liest man mit dem Konvergenzkriterium von Satz 1.1 ab, dass Konvergenz zum Punkte $(\lambda - 1)/\lambda$ vorliegt für $\lambda \in (1, 3)$. Für $\lambda > 3$ divergiert die Iterationsfolge.

Im Beispiel auf Seite 28 haben wir den Grenzfall $\lambda = 3$ untersucht und festgestellt, dass die Iterationsfolge gerade noch konvergiert, allerdings sehr langsam. Man beachte, dass das Kriterium (1.4) in diesem Fall nicht erfüllt ist. Für $\lambda < 1$ konvergiert die Iterationsfolge zum Nullpunkt.

Wie verhält sich die Folge im Grenzfall $\lambda = 1$? Auch in diesem Grenzfall ist das Kriterium nicht erfüllt, wir müssen das Problem als wieder gesondert analysieren. Es gilt

$$x_{n+1} = x_n(1 - x_n) = x_n - x_n^2.$$

Für $x_n > 0$ folgt also, dass $0 < x_{n+1} < x_n$ ist, die Folge wird also konvergieren. Allerdings ist die Konvergenz wieder ähnlich langsam wie im Falle $\lambda = 3$. Für einen Wert $x_n < 0$ ist allerdings $x_{n+1} = x_n - x_n^2 < x_n$, d. h. die Folge divergiert. Dieses Verhalten lässt sich auch allgemein formulieren.

Satz 1.2. Falls $f'(x^*) = 1$, dann konvergiert die Iterationsfolge für einen Startwert x_0 genügend nahe und grösser als x^* wenn $f''(x^*) < 0$ und sie divergiert für $f''(x^*) > 0$. Für einen Startwert genügend nahe und kleiner als x^* dagegen konvergiert die Iterationsfolge für $f''(x^*) < 0$ und divergiert für $f''(x^*) > 0$.

Beweis. Wir entwickeln die Funktion f im Punkt x^* die Potenzreihe

$$\begin{aligned} f(x^* + \delta) &= f(x^*) + f'(x^*) \cdot \delta + \frac{1}{2} f''(x^*) \cdot \delta^2 + O(\delta^3) \\ &= x^* + \delta + \frac{1}{2} f''(x^*) \delta^2 \end{aligned}$$

Die Entfernung zu Fixpunkt ist

$$|f(x^* + \delta) - x^*| \approx |\delta + \frac{1}{2}f''(x^*)\delta^2| = |\delta| \cdot |1 + \frac{1}{2}f''(x^*)\delta|.$$

Ob die Entfernung wird genau dann grösser, wenn $f''(x^*)\delta$ positiv ist. sie wird kleiner, wenn $f''(x^*)\delta$ negativ sind. Der erste Fall tritt ein, wenn δ und $f''(x^*)$ gleiches Vorzeichen haben. Für Werte grösser als x^* heisst das, dass $f''(x^*) > 0$ sein muss. Analog folgen alle anderen Fälle. \square

Für den Fall $\lambda = 3$ der logistischen Gleichung können wir den folgenden Satz formulieren:

Satz 1.3. Falls $f'(x^*) = -1$, dann konvergiert die Iterationsfolge für einen Startwert genügend nahe bei x^* , wenn $f''(x^*) < 0$, sie divergiert, wenn $f''(x^*) > 0$.

Beweis. Wir verwenden wieder die Entwicklung

$$\begin{aligned} f(x_n) &= f(x^* + \delta_n) = f(x^*) + f'(x^*) \cdot \delta_n + \frac{1}{2}f''(x^*) \cdot \delta_n^2 + \dots = x^* - \delta_n + \frac{1}{2}f''(x^*)\delta_n^2 + \dots \\ \delta_{n+1} &= -\delta_n + \frac{1}{2}f''(x^*)\delta_n^2 \dots \end{aligned}$$

Daraus kann man zunächst ableiten, dass der Fehler bei jedem Iterationsschritt das Vorzeichen wechselt. Wir berechnen den Fehler nach zwei solchen Schritten.

$$\begin{aligned} \delta_{n+2} &= -\delta_{n+1} + \frac{1}{2}f''(x^*)\delta_{n+1}^2 = \delta_n - \frac{1}{2}f''(x^*)\delta_n^2 + \frac{1}{2}f''(x^*)(-\delta_n + f''(x^*)\delta_n^2)^2 \\ &= \delta_n - \frac{1}{2}f''(x^*)\delta_n^2 + \frac{1}{2}f''(x^*)(\delta_n^2 + 2f''(x^*)\delta_n^3 + f''(x^*)^2\delta_n^4) \\ &= \delta_n + f''(x^*)^2\delta_n^3 + \frac{1}{2}f''(x^*)^3\delta_n^4 + \dots \\ &= \delta_n(1 + f''(x^*)\delta_n^2) \end{aligned}$$

Die Terme höherer Ordnung als 3 können für kleines δ_n vernachlässigt werden. Nach zwei Iterationsschritten hat also der Fehler wieder das gleiche Vorzeichen, aber der Betrag hat sich im den Faktor

$$1 + f''(x^*)\delta_n^2$$

verändert. Dieser Faktor ist genau dann < 1 , wenn $f''(x^*) < 0$ ist. \square

Ähnlich wie im Beispiel auf Seite 28 kann man auch in diesem Fall zeigen, dass die Konvergenz der Folge sehr langsam ist.

1.3.5 Dritte Ableitung im Fixpunkt

Die bisher zusammengetragenen Sätze decken die Fälle $|f'(x^*)| = 1$ mit nicht verschwindender zweiter Ableitung $f''(x^*) \neq 0$ ab. In einigen Fällen ist Konvergenz nicht gesichert, doch wenn Konvergenz vorliegt, dann ist sie sehr langsam. Speziell an dieser Situation ist, dass $f(x) - (x^* + f'(x^*)(x - x^*))$ das Vorzeichen in einer Umgebung von x^* nicht wechselt.

Noch nicht untersucht wurde der Fall $f''(x^*) = 0$. Es ist klar, dass die Konvergenzgeschwindigkeit nicht schneller werden kann. Für $f'''(x^*) \neq 0$ folgt zudem, dass in diesem Fall $f(x) - (x^* + f'(x^*)(x - x^*))$ das Vorzeichen in einer Umgebung von x^* wechselt. Da aber das Iterationsverfahren

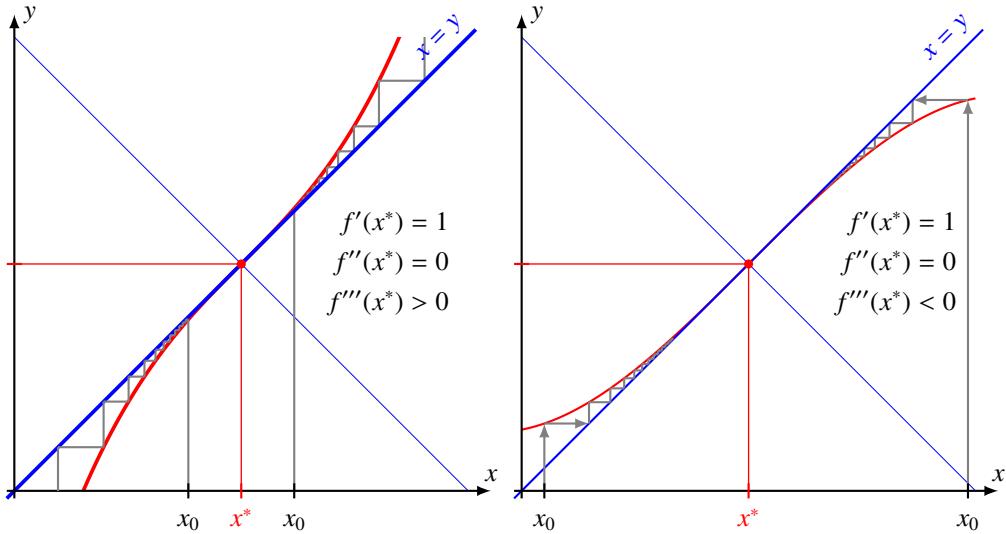


Abbildung 1.5: Die Fixpunktiteration $x_{n+1} = f(x_n)$ mit Fixpunkt x^* mit $f'(x^*) = 1$ und $f''(x^*) = 0$ divergiert für $f'''(x^*) > 0$ und konvergiert sehr langsam für $f'''(x^*) < 0$.

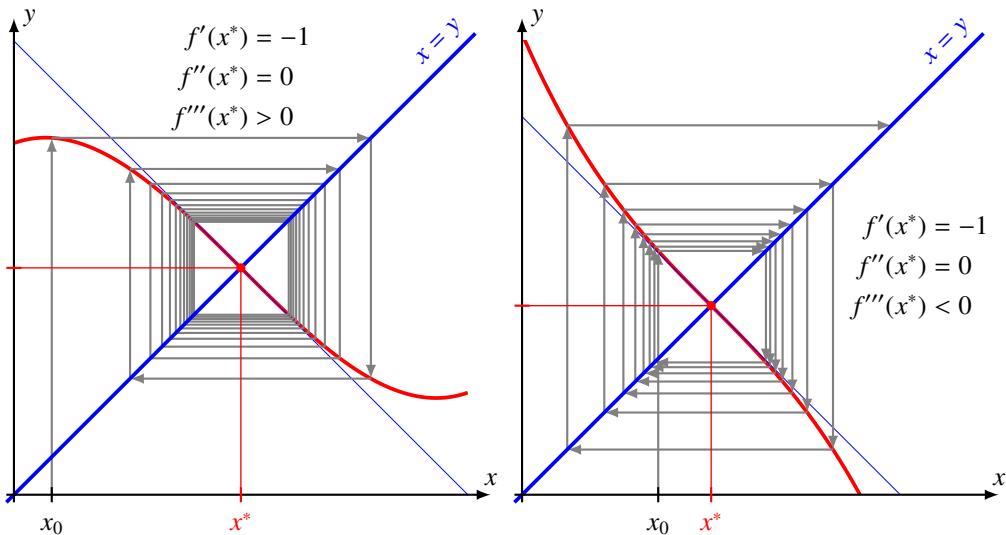


Abbildung 1.6: Die Fixpunktiteration $x_{n+1} = f(x_n)$ mit Fixpunkt x^* mit $f'(x^*) = -1$ und $f''(x^*) = 0$ konvergiert sehr langsam für $f'''(x^*) > 0$ und divergiert für $f'''(x^*) < 0$.

sehr empfindlich darauf regiert, ob der Graph von f oberhalb oder unterhalb der Geraden $y = \pm x$ liegt, erwarten wir ein anderes Verhalten als im Fall $f''(x^*) \neq 0$. Alle möglichen Situationen sind in den Abbildungen 1.5 und 1.6 dargestellt.

Wir gehen also davon aus, dass sich f in einer Umgebung von x^* in der Form

$$f(x^* + \delta) \approx \tilde{f}(x^* + \delta) = x^* + f'(x^*)\delta + \frac{1}{6}f'''(x^*)\delta^3 = x^* + s\delta + a\delta^3$$

entwickeln lässt, wobei $s = \pm 1$ und $a \neq 0$ ist.

Der Iterationsfehler δ_{n+1} verhält sich wie

$$\begin{aligned} x^* + \delta_{n+1} &= f(x_n) = f(x^* + \delta_n) = x^* + s\delta_n + a\delta_n^3 + O(\delta_n^4). \\ \delta_{n+1} &= s\delta_n + a\delta_n^3 + O(\delta_n^4) = \delta_n(s + a\delta_n^2) + O(\delta_n^4) = s\delta_n\left(1 + \frac{a}{s}\delta_n^2\right) + O(\delta_n^4) \end{aligned}$$

Da $\delta_n^2 > 0$ ist, ist der Klammerausdruck genügend nahe bei x^* immer positiv. Der Betrag des Fehlers verhält sich dann wie

$$|\delta_{n+1}| = |\delta_n| \cdot \left(1 + \frac{a}{s}\delta_n^2\right) + O(\delta_n^4).$$

Ob der Fehler bei der Iteration grösser oder kleiner wird, hängt also einzig vom Vorzeichen von a/s ab. Ist $a/s > 0$, wird der Fehler grösser, die Iterationsfolge ist divergent, ist $a/s < 0$, wird der Fehler kleiner, die Iterationsfolge konvergiert. Wir fassen diese Resultate zusammen im folgenden Satz.

Satz 1.4. Falls $f: \mathbb{R} \rightarrow \mathbb{R}$ den Fixpunkt $x^* \in \mathbb{R}$ hat, und $f'(x^*) = \pm 1$, $f''(x^*) = 0$ und $f'''(x^*) \neq 0$ ist, dann ist die Iterationsfolge $x_{n+1} = f(x_n)$ konvergent, falls $f'(x^*) \cdot f'''(x^*) < 0$ ist, gleichbedeutend damit, wenn $f'(x^*)$ und $f'''(x^*)$ verschiedene Vorzeichen haben. Wenn $f'(x^*)$ und $f'''(x^*)$ das gleiche Vorzeichen haben, dann ist die Fixpunktiterationsfolge divergent.

Abbildung 1.5 zeigt die Situation $f'(x^*) = 1$ während Abbildung 1.6 die Situation $f'(x^*) = -1$ abdeckt.

1.4 Konvergenzgeschwindigkeit

Sind die Fehler einer iterativen numerischen Berechnungsmethoden bekannt, kann die Gesetzmässigkeit der Fehler ausgenutzt werden, sie weitgehend zu eliminieren, und so die Konvergenz des Verfahrens zu beschleunigen. In diesem Abschnitt wollen wir das Prinzip veranschaulichen, es soll später im Abschnitt 4.2 bei der Berechnung von Integralen systematisch angewendet werden.

1.4.1 Lineare und quadratische Konvergenz

Wir betrachten zwei Beispiele von Iterationsfolgen genauer, die deutlich unterschiedlich schnell konvergieren. Ziel ist, diesen Unterschied zu quantifizieren und ein Kriterium zu finden, welches schnelle Konvergenz von Iterationsfolgen garantiert.

Lineare Konvergenz

Am Beispiel der Iterationsfolge der Funktion $f(x) = \sqrt{x+2}$ von Seite 26 wurde berechnet, wie sich der Fehler von einer Iteration zur nächsten entwickelt. Dabei wurde gefunden, dass der Fehler δ_n in der n -ten Iteration zu $\delta_{n+1} \approx f'(x^*) \cdot \delta_n$ in der $(n+1)$ -ten Iteration wird. Sind in der n -ten

Iteration k Nachkommabits korrekt, ist der Fehler also von der Größenordnung 2^{-k} , wird der Fehler in der $(n+1)$ -ten Iteration von der Größenordnung $f'(x^*) \cdot 2^{-k} = 2^{-k+\log_2 f'(x^*)}$ sein. In jeder Iteration werden also $\log_2 f'(x^*)$ Binärstellen Genauigkeit gewonnen.

Man sagt, eine Folge ist *linear konvergent*, wenn die Anzahl korrekter Stellen in jeder Iteration um die gleiche Anzahl zunimmt. Die Anzahl korrekter Stellen wächst in diesem Fall linear mit der Anzahl Iterationen.

Für eine konvergente Iterationsfolge einer Funktion f liegt also normalerweise lineare Konvergenz vor, es werden $\log_2 f'(x^*)$ Binärstellen oder $\log_{10} f'(x^3)$ Dezimalstellen Genauigkeit in jeder Iteration gewonnen. Falls die Ableitung $f'(x^*)$ verschwindet liegt offenbar ein Spezialfall vor, er wird im übernächsten Abschnitt genauer untersucht.

Quadratische Konvergenz

Ist $x = \sqrt{a}$, dann muss $x = a/x$ gelten, also auch

$$x = \frac{1}{2}\left(x + \frac{a}{x}\right) = f_a(x).$$

Mit der Funktion $f_a(x)$ kann man jetzt ein Iterationsfolge konstruieren, die gegen den Fixpunkt

$$x^* = f_a(x^*) \quad \Rightarrow \quad x^* = \frac{1}{2}\left(x^* + \frac{a^*}{x^*}\right) \quad \Rightarrow \quad x^{*2} - a = 0 \quad \Rightarrow \quad x^* = \sqrt{a}$$

von $f_a(x)$ konvergiert. Wir untersuchen die Konvergenzgeschwindigkeit, indem wir $f_a(x)$ mit Hilfe der Taylor-Reihe approximieren:

$$f_a(x^* + \delta) \approx f_a(x^*) + f'_a(x^*) \cdot \delta + \frac{1}{2} f''_a(x^*) \cdot \delta^2$$

Die Ableitungen von f_a sind

$$\begin{aligned} f'_a(x) &= \frac{1}{2} - \frac{a}{2x^2} & \Rightarrow & & f'_a(x^*) &= 0 \\ f''_a(x^*) &= \frac{a}{x^3} & \Rightarrow & & f''_a(x^*) &= \frac{1}{\sqrt{a}} \end{aligned}$$

Mit $x = x^* = \sqrt{a}$ folgt für den Fehler

$$f_a(x^* + \delta) \approx x^* + \underbrace{\left(\frac{1}{2} - \frac{a}{2x^{*2}}\right)}_{=0} \delta + \frac{a}{2x^{*3}} \delta^2 = x^* + \frac{1}{2\sqrt{a}} \delta^2.$$

Der Fehler wird also im Wesentlichen quadriert. Sind k Nachkommabits korrekt, liegt ein Fehler von der Größenordnung 2^{-k} vor. Daraus wird in der nächsten Iteration ein Fehler von der Größenordnung 2^{-2k} . Die Anzahl korrekter Nachkommabits ist $2k$, sie hat sich also verdoppelt.

Man sagt, eine Folge konvergiert *quadratisch* gegen x^* , wenn der Fehler $x_n - x^*$ für $n \rightarrow n+1$ quadriert wird, die Anzahl korrekter Stellen also verdoppelt wird. Konvergiert eine Folge quadratisch, werden N korrekte Stellen innert $\log_2 N$ Iterationen erreicht. Quadratische Konvergenz ist also im Vergleich zu linearer Konvergenz exponentiell schneller und wird daher in Anwendungen angestrebt.

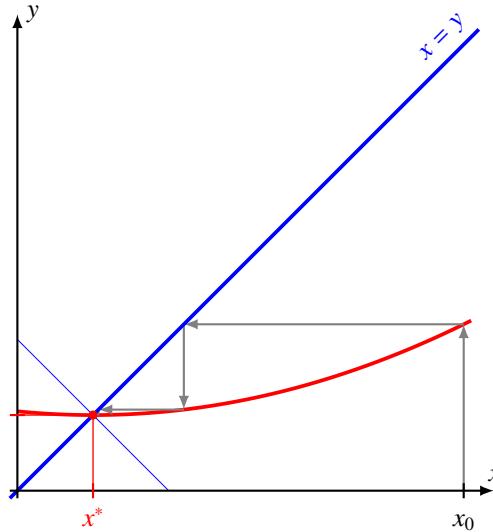


Abbildung 1.7: Die Fixpunktiteration $x_{n+1} = f(x_n)$ konvergiert quadratisch gegen den Fixpunkt x^* für $f'(x^*) = 0$.

Konvergenzgeschwindigkeit von Iterationsfolgen

Früher wurde gezeigt, dass die Iterationsfolge $x_{n+1} = f(x_n)$ einer Funktion f für genügend Nahe bei einem Fixpunkt x^* liegende Startwerte x_0 gegen x^* konvergiert, wenn $|f'(x^*)| < 1$. Wir sind jetzt auch in der Lage, die Konvergenzgeschwindigkeit zu quantifizieren. Dazu entwickeln wird $f(x)$ um x^* bis zur zweiten Ordnung und erhalten für den Fehler

$$f(x^* + \delta) \approx f(x^*) + f'(x^*) \cdot \delta + \frac{1}{2} f''(x^*) \cdot \delta^2 \quad (1.5)$$

Sofern $f'(x^*) \neq 0$ ist, ist der zweite Term dominant, der Fehler wird in jeder Iteration mit $f'(x^*)$ multipliziert, es werden in jeder Iteration $\log_2 f'(x^*)$ Genauigkeit gewonnen. Es liegt also *lineare Konvergenz* vor.

Verschwindet die Ableitung $f'(x^*) = 0$, dann fällt der zweite Term auf der rechten Seite von (1.5) weg, der Fehler ist im wesentlich quadratisch kleiner, es liegt *quadratische Konvergenz* vor. Diese Situation ist in Abbildung 1.7 Wenden wir das Kriterium auf das Beispiel

$$f(x) = \frac{1}{2} \left(x + \frac{a}{x} \right) \quad \text{mit der Ableitung} \quad f'(x) = \frac{1}{2} \left(1 - \frac{a}{x^2} \right)$$

an, finden wir für $f'(x^*) = f'(\sqrt{a}) = 0$. Die Iterationsfolge muss also quadratisch konvergieren, wie wir im vorangegangenen Abschnitt auch direkt nachgerechnet haben.

1.4.2 Konvergenzbeschleunigung

Die genaue Kenntnis des Fehlers kann auch ermöglichen, einen Teil des Fehlers zu eliminieren. Wir illustrieren dieses Prinzip wieder am Beispiel der Iterationsfolge

$$x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), \dots$$

mit der Funktion $f(x) = \sqrt{x+a}$. Die Folge konvergiert gegen den Fixpunkt $x^* = f(x^*)$, der sich durch Quadrieren und Lösen der quadratische Gleichung

$$\begin{aligned} x^{*2} &= x^* + a \\ x^{*2} - x^* - a &= 0 \\ \Rightarrow \quad x^* &= \frac{1}{2} + \sqrt{\frac{1}{4} + a} \end{aligned}$$

finden lässt. Die negative Wurzel kommt nicht in Frage, weil dies auf eine negative Zahl führen würde, die nicht Fixpunkt der Wurzelfunktion sein kann, die nur positive Werte hat.

Wie im früheren Beispiel mit $a = 2$ können wir das Verhalten des Fehlers $\delta_n = x_n - x^*$ mit Hilfe der linearen Approximation

$$x_n = x^* + \delta_n \quad \Rightarrow \quad x_{n+1} = f(x_n) = f(x^* + \delta_n) = f(x^*) + f'(x^*) \cdot \delta_n = x^* + \underbrace{f'(x^*) \cdot \delta_n}_{\approx \delta_{n+1}}$$

ermitteln. Der Fehler wird also in jeder Iteration um den Faktor

$$f'(x^*) = \frac{1}{2\sqrt{x^* + a}} = \frac{1}{2x^*}$$

kleiner. Wir wissen somit, dass der Fehler in jeder Iteration um den *gleichen* Faktor kleiner wird, aber da wir x^* noch nicht kennen, kennen wir den Wert q des Faktors noch nicht.

Drei aufeinanderfolgende Folgenglieder sind

$$x_{n-1} = x^* + \delta \quad (1.6)$$

$$x_n = x^* + q \delta \quad (1.6)$$

$$x_{n+1} = x^* + q^2 \delta \quad (1.7)$$

Darin sind alle drei Größen auf der rechten Seite unbekannt. Die Differenzen aufeinanderfolgender Folgenglieder sind

$$\begin{aligned} x_n - x_{n-1} &= \delta(q-1) \\ x_{n+1} - x_n &= \delta(q^2 - q) = \delta q(q-1). \end{aligned}$$

Der Differenzenquotient

$$\frac{x_{n+1} - x_n}{x_n - x_{n-1}} = q$$

kann verwendet werden, eine bessere Approximation zu bestimmen. Dazu multipliziert man (1.6) mit q und subtrahiert das Resultat von (1.7). Man erhält

$$x_{n+1} - qx_n = (1-q)x^* \quad \Rightarrow \quad x^* = \frac{x_{n+1} - qx_n}{1-q}.$$

Damit können wir eine neue Iteration definieren:

1. Berechne aus x_0 die Werte $x_1 = f(x_0)$ und $x_2 = f(x_1)$.

2. Berechne

$$q = \frac{x_2 - x_1}{x_1 - x_0} \quad \text{und} \quad x^* = \frac{x_2 - qx_1}{1-q}.$$

k	Fixpunkt	Fehler	q
1	2.039426062845019	0.039426062845019	0.306
2	2.000008018463113	0.000008018463113	0.249
3	2.000000000000335	0.000000000000335	0.249
4	2.000000000000000	0.000000000000000	0.249

Tabelle 1.10: Resultate des beschleunigten Verfahrens zur Bestimmung eines Fixpunktes der Iterationsfolge der Funktion $f(x) = \sqrt{x+a}$ mit $a = 2$.

3. Setze $x_0 = x^*$ und beginne bei 1.

Das neue Verfahren zur Berechnung des Fixpunktes konvergiert jetzt viel schneller, wie die Resultate in Tabelle 1.10 zeigen. Die Konvergenz ist sehr rasch, es scheint als ob sich in jeder Iteration die Anzahl der korrekten Stellen verdoppelt, dass also quadratische Konvergenz vorliegt. Dies lässt sich auch mit Hilfe einer analytischen Rechnung bestätigen. Dazu approximiert man $f(x^* + \delta)$ bis zu Termen zweiter Ordnung, also als

$$f(x^* + \delta) = f(x^*) + f'(x^*) \cdot \delta + \frac{1}{2} f''(x^*) \cdot \delta^2 + \dots = x^* + q\delta + b\delta^2$$

verwendet dies für die Analyse des neuen Iterationsverfahrens. Wir schreiben also

$$\begin{aligned} x_0 &= x^* + \delta && (1.8) \\ x_1 &= x^* + q\delta + b\delta^2 \\ x_2 &= x^* + q(q\delta + b\delta^2) + b(q\delta + b\delta^2)^2 = x^* + q^2\delta + qb\delta^2 + bq^2\delta^2 + 2qb^2\delta^3 + b^3\delta^4 \\ &= x^* + q^2\delta + qb(1+q)\delta^2 \end{aligned}$$

und wenden den neuen Algorithmus darauf an. Der Quotient ist

$$\begin{aligned} \frac{x_2 - x_1}{x_1 - x_0} &= \frac{x^* + q^2\delta + qb(1+q)\delta^2 - (x^* + q\delta + b\delta^2)}{x^* + q\delta + b\delta^2 - (x^* + \delta)} = \frac{q(q-1)\delta + b(q^2 + q - 1)\delta^2}{(q-1)\delta + b\delta^2} \\ &\approx \frac{q(q-1)\delta}{(q-1)\delta} = q. \end{aligned}$$

In der zweiten Zeile gehen wir davon aus, dass $\delta^2 \ll \delta$ und dass daher die Terme mit δ^2 im Bruch vernachlässigt werden dürfen. Nach dem zweiten Schritt des Algorithmus ist der neue Näherungswert von x^* gegeben durch

$$\begin{aligned} \frac{x_2 - qx_1}{1-q} &= \frac{x^* + q^2\delta + qb(1+q)\delta^2 - q(x^* + q\delta + b\delta^2)}{1-q} = \frac{(1-q)x^* + qb(1+q)\delta^2 - qb\delta^2}{1-q} && (1.9) \\ &= x^* + \frac{q^2b}{1-q}\delta^2. && (1.10) \end{aligned}$$

Der ursprüngliche Fehler δ in (1.8) wurde durch die Iteration nach (1.10) im wesentlichen quadriert worden. Damit ist die quadratische Konvergenz bestätigt.

Ein fairer Vergleich der Konvergenzgeschwindigkeit der ursprünglichen Iterationsfolge mit dem neuen Algorithmus sollte die Anzahl der Auswertungen der Funktion f mit zählen. In der ursprünglichen Iterationsfolge wird in jeder Iteration die Funktion einmal ausgewertet, der neue Algorithmus

braucht zwei Funktionsauswertungen pro Iteration. In der ursprünglichen Iterationsfolge wurden 52 bit Genauigkeit nach 26 Schritten und damit nach 26 Funktionsauswertungen erreicht. Die neue Folge erreicht Konvergenz in 4 Iterationen und damit 8 Funktionsauswertungen. Auch unter Berücksichtigung der Anzahl Funktionsauswertungen ist das neue Verfahren bedeutend weniger aufwendig.

Trotz dieses spektakulären Erfolgs der Konvergenzbeschleunigung weist das Verfahren für praktische Rechnungen einige entscheidende Mängel auf. Da die Folge gegen x^* konvergiert, werden die Werte x_0 , x_1 und x_2 fast gleich gross sein, so dass es zu Auslöschung und damit zu Werten sehr geringer Genauigkeit für q kommen kann. Ein solcher Fehler in q schlägt sich wegen (1.9) sofort auch im nächsten Approximationswert für x^* nieder.

Im Abschnitt 4.2 wird am Beispiel des Romberg-Integrationsverfahrens gezeigt, wie die Konvergenz der Integralberechnung mit der Trapezregel beschleunigt werden kann.

1.5 Numerische Instabilität

Die Untersuchungen zum Verhalten von Iterationsfolgen sind davon ausgegangen, dass alle Rechnung ohne Fehler ausgeführt werden können. Dies ist aber nicht realistisch, in den meisten numerischen Berechnungen müssen Zwischenresultate mit der beschränkten verfügbaren Genauigkeit der Gleitkommaformate gespeichert werden, die die Maschine zur Verfügung stellt. Es ist daher zu untersuchen, wie sich die Konvergenz eines Lösungsverfahrens ändert, wenn es durch Rundungsfehler im Laufe der Rechnung gestört wird.

1.5.1 Eine instabile Quadratwurzel

Die Quadratwurzel von a ist ein Fixpunkt der Funktion $f(x) = x^3/a$ denn $f(\sqrt{a}) = a^{\frac{3}{2}}/a = \sqrt{a}$. Das Konvergenzkriterium Satz 1.1 besagt, dass Konvergenz garantiert ist, wenn der Betrag der Ableitung von $f'(x)$ am Fixpunkt kleiner als 1 ist. Aber

$$f'(x) = \frac{3x^2}{a} \quad \Rightarrow \quad f'(\sqrt{a}) = \frac{3\sqrt{a}^2}{a} = 3,$$

die Iterationsfolge wird also niemals konvergieren. Im Gegenteil, der Fehler wird in jeder Iteration um den Faktor 3 anwachsen.

Der `float`-Gleitkommatyp verwendet eine Mantisse von 23 bit, das niederwertigste Bit hat also die Wertigkeit 2^{-23} . Eine Approximation von \sqrt{a} wird daher, sofern sie nicht exakt ist, beim Speichern als `float`-Zahl einen Fehler von der Größenordnung 2^{-24} aufnehmen. Nach k Iterationen wird dieser Fehler auf $3^k 2^{-24} = 2^{k \log_2 3 - 24}$ angewachsen sein. Nach $k \geq 24 / \log_2 3 = 15.142$ Iterationen ist also der Fehler von der gleichen Größenordnung wie das gesuchte Resultat.

Wir illustrieren dies mit einer Rechnung⁶, deren Resultate in Tabelle 1.11 zusammengestellt sind. Zunächst berechnen wir die Quadratwurzel $\sqrt{2}$ als `float`-Zahl. Dann bilden wir die nächsten Nachbarn, die sich nur in den letzten zwei Bits unterscheiden, die letzten fünf Bits der Mantisse dieser Zahlen sind in der obersten Kopfzeile der Tabelle gezeigt. In der zweiten Kopfzeile sieht man, dass diese Unterschiede so klein sind, dass sie nur gerade die Rundung in der sechsten dezimalen Nachkommastelle beeinflussen können. Der Unterschied dieser Startwerte zum Maschinen-Wert für $\sqrt{2}$ ist in der dritten Kopfzeile dargestellt.

Im unteren Teil der Tabelle wird dann ausgehend von jedem dieser Startwerte die Iterationsfolge mit der Funktion $f(x) = x^3/2$ gebildet. Spätestens nach der zweiten Iteration beginnen sich die Werte

⁶Das C++-Programm hierzu ist `sqrt.cpp` im Verzeichnis `buch/chapters/experimente/sqrt` von [4].

	10001 1.414213 $-2.384 \cdot 10^{-7}$	10010 1.414213 $-1.192 \cdot 10^{-7}$	10011 1.414214 0	10100 1.414214 $1.192 \cdot 10^{-7}$	10101 1.414214 $2.384 \cdot 10^{-7}$
1	1.414213	1.414213	1.414213	1.414214	1.414214
2	1.414211	1.414212	1.414213	1.414214	1.414216
3	1.414207	1.414210	1.414212	1.414216	1.414220
4	1.414193	1.414204	1.414210	1.414220	1.414232
5	1.414152	1.414184	1.414204	1.414232	1.414268
6	1.414029	1.414126	1.414184	1.414268	1.414377
7	1.413660	1.413951	1.414126	1.414377	1.414705
8	1.412553	1.413427	1.413951	1.414705	1.415687
9	1.409239	1.411856	1.413427	1.415687	1.418640
10	1.399341	1.407152	1.411856	1.418640	1.427533
11	1.370064	1.393135	1.407152	1.427533	1.454550
12	1.285858	1.351915	1.393135	1.454550	1.538706
13	1.063039	1.235429	1.351915	1.538706	1.821534
14	0.600645	0.942808	1.235429	1.821534	3.021912
15	0.108348	0.419024	0.942808	3.021912	13.797982
16	$6.359 \cdot 10^{-4}$	$3.678 \cdot 10^{-2}$	0.419024	13.797982	$1.313 \cdot 10^3$
17	$1.286 \cdot 10^{-10}$	$2.489 \cdot 10^{-5}$	$3.678 \cdot 10^{-2}$	$1.313 \cdot 10^{-3}$	$1.132 \cdot 10^9$
18	$1.063 \cdot 10^{-30}$	$7.710 \cdot 10^{-15}$	$2.489 \cdot 10^{-5}$	$1.132 \cdot 10^{-9}$	$7.271 \cdot 10^{26}$
19	0.000000	$2.298 \cdot 10^{-43}$	$7.710 \cdot 10^{-15}$	$7.271 \cdot 10^{26}$	∞
20	0.000000	0.000000	$2.298 \cdot 10^{-43}$	∞	∞

Tabelle 1.11: Iterationsfolge der Funktion $f(x) = x^3/2$ für verschiedene Näherungen des Fixpunktes $x^* = \sqrt{2}$. Die Startwerte unterscheiden sich nur in den letzten zwei Bits ihrer Darstellung als float-Gleitkommazahl, die letzten fünf Bits der Mantisse sind in der ersten Kopfzeile dargestellt. In der dritten Zeile stehen die Differenzen zur besten Approximation von $\sqrt{2}$ durch eine float-Zahl.

vom Fixpunkt zu entfernen, nach 16 Iterationen sind die Abweichungen von der Größenordnung 1 wie in der Überschlagsrechnung weiter oben vorhergesagt.

1.5.2 Numerische Instabilität

Von *numerischer Instabilität* spricht man, wenn ein Berechnungsverfahren allein wegen der unvermeidlichen Rundungsfehler keine sinnvollen Resultate liefern kann.

Beispiel. Es soll das Integral

$$I_n = \int_0^1 \frac{x^n}{x+a} dx$$

für festes $a > 1$ und für ganze Zahlen n mit $1 \leq n \leq 15$ berechnet werden. Da im Intervall $[0, 1]$ gilt $x^{n+1} > x^n$ ist I_n eine monoton abnehmende Folge. Es ist auch klar, dass $\lim_{n \rightarrow \infty} I_n = 0$.

n	I_n	Rückwärtsiteration
0	0.0953101798043249	0.09531017980432486
1	0.0468982019567507	0.04689820195675140
2	0.0310179804324935	0.03101798043248600
3	0.0231535290083985	0.02315352900847329
4	0.0184647099160155	0.01846470991526711
5	0.0153529008398455	0.01535290084732894
6	0.0131376582682119	0.01313765819337729
7	0.0114805601750236	0.01148056092337000
8	0.0101943982497636	0.01019439076629997
9	0.0091671286134746	0.00916720344811137
10	0.0083287138652537	0.00832796551888631
11	0.0076219522565537	0.00762943572022779
12	0.0071138107677959	0.00703897613105546
13	0.0057849692451174	0.00653331561252233
14	0.0135788789773972	0.00609541530334817
15	-0.0691221231073049	0.00571251363318499
16	0.753721231073049	0.00537486366815009
17	-7.47838878131872	0.00507489273026384
18	74.8394433687428	0.00480662825291712
19	-748.341802108480	0.00456529641819719
20	7483.46802108480	0.00434703581802811

Tabelle 1.12: Instabile Iteration zur Berechnung der Integrale I_n mit $a = 10$. In jedem Schritt wird der Fehler mit a multipliziert. In der dritten Spalte die Resultate der stabilen Rückwärtsiteration.

Das Integral ist im Prinzip nicht schwierig zu berechnen, wenn man im Integranden die Polynomdivision ausführt:

$$\begin{aligned} \frac{x^n}{x+a} &= x^{n-1} - ax^{n-2} + a^2 x^{n-3} - \cdots + (-1)^n x a^{n-2} + (-1)^{n-1} a^{n-1} + (-1)^n \frac{a^n}{a+x} \\ \int_0^1 \frac{x^n}{x+a} dx &= \frac{1}{n} - \frac{a}{n-1} + \frac{a^2}{n-2} - \cdots + (-1)^n \frac{a^{n-2}}{2} + (-1)^{n-1} a^{n-1} + \log(1+a) - \log a. \quad (1.11) \end{aligned}$$

Jeder Term ist ein elementares Integral. ○

Wenn n gross ist, ist (1.11) eine ziemlich aufwendig Rechnung. Da man das Integral für alle n haben will, bietet sich eine Rekursionsformel an. Es ist nämlich

$$\begin{aligned} I_n &= \int_0^1 \frac{x^n}{x+a} dx = \int_0^1 \frac{x^{n-1}(x+a-a)}{x+a} dx = \int_0^1 x^n - \frac{ax^{n-1}}{x+a} dx \\ &= \int_0^1 x^n dx - aI_{n-1} = \frac{1}{n+1} - aI_{n-1}. \end{aligned}$$

Wenn man also I_0 berechnet hat, dann kann man mit dieser Rekursionsformel alle anderen I_n berechnen. I_0 ist nicht schwierig zu berechnen, es ist

$$I_0 = \int_0^1 \frac{1}{x+a} dx = \log(1+a) - \log a = \log \frac{1+a}{a}.$$

Um die Empfindlichkeit der Rekursion auf Rundungsfehler zu untersuchen, nehmen wir vereinfachend an, dass ausschliesslich bei der Berechnung von I_0 ein unvermeidlicher Rundungsfehler der Grösse ε passiert und dass alle folgenden Operationen exakt ausgeführt werden können. Da der Logarithmus eine transzendente Funktion ist, werden fast alle Werte des Logarithmus bei Speichern als Gleitkommazahl gerundet werden müssen. Wir müssen jetzt also die Rekursionsfolge I_n^* ausgehend von $I_0^* = I_0 + \varepsilon$ bilden:

$$\begin{aligned} I_0^* &= I_0 + \varepsilon \\ I_1^* &= \frac{1}{2} - a(I_0^*) = \frac{1}{2} - aI_0 - a\varepsilon &= I_1 - a\varepsilon \\ I_2^* &= \frac{1}{3} - a(I_1^*) = \frac{1}{3} - aI_1 + a^2\varepsilon &= I_2 + a^2\varepsilon \\ \vdots &\quad \vdots \\ I_n^* &= \frac{1}{n+1} - a(I_{n-1}^*) = \frac{1}{n+1} - aI_{n-1} + (-a)^n\varepsilon &= I_{n-1} + (-a)^n\varepsilon. \end{aligned}$$

In jedem Iterationsschritt wird der Fehler mit a multipliziert. In jedem Schritt gehen als $\log_2 a$ Binärstellen Genauigkeit verloren. Für $a = 10$ bedeutet dies, dass in jedem Schritt eine Dezimalstelle verloren geht.

Die Instabilität röhrt daher, dass der Fehler in jedem Schritt mit $a > 1$ multipliziert wird. Wir könnten versuchen, die Rekursion umzukehren, so dass durch a dividiert wird, so könnte das Problem möglicherweise behoben werden. Indem wir nach I_{n-1} auflösen, erhalten wir die Rekursionsformel

$$I_{n-1} = \frac{1}{a} \left(\frac{1}{n+1} - I_n \right).$$

In dieser Rekursion wird tatsächlich in jedem Schritt durch a dividiert, man darf daher davon ausgehen, dass in jeder Iteration $\log_2 a$ Binärstellen Genauigkeit gewonnen werden.

Allerdings muss man für diese Iteration einen der Werte I_n bereits kennen. Man weiss aber, dass $\lim_{n \rightarrow \infty} I_n = 0$ ist, d. h. wenn man I_n durch 0 ersetzt macht man einen Fehler exakt von der Grössenordnung I_n . Die Rekursion reduziert diesen Fehler in jedem Schritt um $\log_2 a$ Binärstellen. Da die Werte I_n alle kleiner als 1 sind, macht man also niemals einen Fehler grösser als 1 und nach k Rückwärtsiterationsschritten ist der Fehler kleiner als a^k . Man kann also selbst ohne die Kenntnis eines Startwertes ausreichend genaue Werte von I_n bestimmen. Dies ist in der dritten Spalte von Tabelle 1.12 durchgeführt.

1.6 Kondition

Die Beispiele zur numerischen Instabilität haben deutlich gemacht, dass Instabilität dadurch entstehen kann, dass der Fehler im Laufe der Rechnung grösser wird und dass diese Rechnung vielfach wiederholt wird.

Man sagt, ein Problem sei *schlecht konditioniert*, wenn eine kleine Änderung der Eingangsdaten eine grosse Änderung der Resultate zur Folge hat. Solche Probleme verlangen, dass von Anfang an mit hoher Genauigkeit gerechnet wird und sie lassen sich schlecht iterieren, da sich die Rundungsfehler mit der Zeit derart aufschaukeln werden, dass man kein Vertrauen mehr in die gefundenen Resultate haben kann.

Gut konditionierte Probleme sind dagegen solche, bei denen kleine Störungen der Eingangsdaten nur geringe Fehler in den Resultaten zur Folge haben. In einem gut konditionierten Problem werden sich während der Rechnung auftretende Rundungsfehler kaum gravierend auswirken.

Übungsaufgaben

1.1. Betrachten Sie die Funktion

$$f(x) = \frac{1 - \cos x}{x}.$$

- a) Berechnen Sie $f(10^{-10})$.
- b) Berechnen Sie $\lim_{x \rightarrow 0} f(x)$.
- c) Was für ein Problem tritt bei der numerischen Berechnung von $f(x)$ für kleine Werte von x auf?
- d) Schätzen Sie ab, wie klein x maximal werden darf, damit die naive Berechnung von $f(x)$ gemäss obiger Formel mit float- oder double-Zahlen einen vom in a) berechneten Grenzwert verschiedenen Wert liefert.
- e) Finden Sie eine Berechnungsformel für $f(x)$, die auch für kleine Werte von x funktioniert.

Lösung. a) Mit jedem Datentyp findet man $f(10^{-10}) = 0$.

- b) Der Grenzwert kann mit Hilfe der Regel von de l'Hospital berechnet werden:

$$\lim_{x \rightarrow 0} f(x) = \lim_{x \rightarrow 0} \frac{1 - \cos x}{x} = \lim_{x \rightarrow 0} \frac{\frac{d}{dx}(1 - \cos x)}{\frac{d}{dx}x} = \lim_{x \rightarrow 0} \frac{\sin x}{1} = 0.$$

- c) Der Wert von $\cos x$ ist sehr nahe bei 1, daher tritt Auslöschung auf.
- d) Die Taylor-Reihe für $\cos x$ ist

$$\cos x = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \dots$$

Der Funktionswert von $f(x)$ lässt sich nicht mehr von 0 unterscheiden, wenn der zweite Term der Reihe sich nicht mehr von 1 unterscheiden lässt. Dies ist genau der Wert für ε für den verwendeten Datentyp, den man der Tabelle 1.2 entnehmen kann. Daraus leitet man ab:

$$\frac{1}{2}x^2 = \varepsilon \quad \Rightarrow \quad x = \sqrt{2\varepsilon}.$$

Für float findet man $x \approx 0.000488281$, für double dagegen $x \approx 2.10734 \cdot 10^{-8}$.

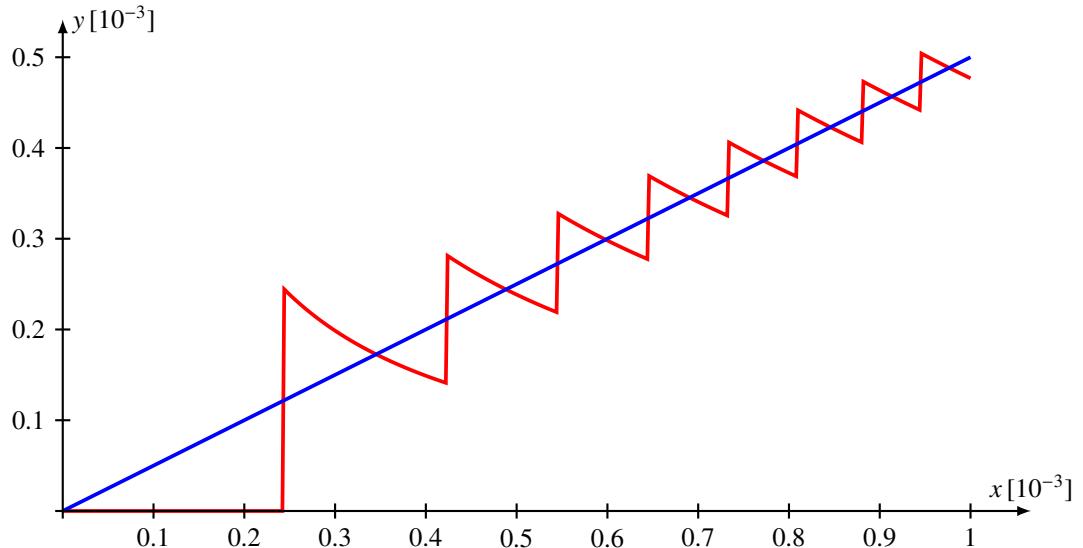


Abbildung 1.8: Numerische Berechnung der Funktion $f(x)$ von Aufgabe 1.1 für kleine Werte von x . Die rote Kurve zeigt die Unzuverlässigkeit der Resultate infolge Auslöschung. Die blaue Kurve zeigt die Berechnung mit der verbesserten Funktion $g(x)$, in der Auslöschung vermieden wird.

- e) Die Halbwinkelformel für die sin-Funktion liefert

$$\sin^2 \frac{x}{2} = \frac{1 - \cos x}{2} \quad \Rightarrow \quad f(x) = \frac{1 - \cos x}{x} = \frac{2 \sin^2(x/2)}{x} = \frac{2}{x} \sin^2 \frac{x}{2} =: g(x).$$

Der Ausdruck $g(x)$ leidet nicht unter Auslöschung. ○

Abbildung 1.8 vergleicht die beiden Ausdrücke $f(x)$ und $g(x)$ zur Berechnung der gegebenen Funktion.

1.2. Man kann zeigen, dass die durch

$$x_{n+1} = 2^{n+1} (\sqrt{1 + 2^{-n} x_n} - 1) \quad (1.12)$$

definierte Folge für Startwerte $x_0 > -1$ gegen $\log(x_0 + 1)$ konvergiert.

- a) Warum tritt in der Rekursionsformel (1.12) Auslöschung auf?
 b) Formulieren Sie (1.12) derart, dass keine Auslöschung auftritt.

Hinweis. Verwenden Sie die Halbwinkelformel

$$\tan \frac{\alpha}{2} = \frac{\sqrt{1 + \tan^2 \alpha} - 1}{\tan \alpha}. \quad (1.13)$$

Lösung. a) Da die Folge x_n konvergiert, geht $2^{-n} x_n$ gegen 0 und damit geht $\sqrt{1 + 2^{-n} x_n}$ gegen 1. Die Differenz mit 1 führt dann zu Auslöschung.

n	x_n nach (1.12)	x_n nach (1.14)
0	0.50000000000	0.50000000000
1	0.4494898319	0.4494897127
2	0.4267277718	0.4267276525
3	0.4159164429	0.4159160256
4	0.4106464386	0.4106463492
5	0.4080467224	0.4080447853
6	0.4067535400	0.4067522287
7	0.4061126709	0.4061080217
8	0.4057922363	0.4057863951
9	0.4056396484	0.4056257606
10	0.4055175781	0.4055454433
11	0.4055175781	0.4055052698
12	0.4052734375	0.4054851830
13	0.4052734375	0.4054751098
14	0.4042968750	0.4054700732
15	0.4023437500	0.4054675400
16	0.3984375000	0.4054662883
17	0.3906250000	0.4054656327
18	0.3750000000	0.4054653049
19	0.3750000000	0.4054651558
20	0.3750000000	0.4054650664
21	0.2500000000	0.4054650068
∞	0.4054650962	0.4054650962

Tabelle 1.13: Auslöschung in der Folge x_n berechnet mit den zwei verschiedenen Rekursionsformeln (1.12) und (1.14) mit Hilfe von float-Zahlen.

- b) Im Zähler der rechten Seite der Halbwinkelformel (1.13) für den Tangens kommt genau der Ausdruck vor, der für die Auslöschung verantwortlich ist. Um die Formel verwenden zu können muss

$$\tan \alpha = \sqrt{\frac{x_n}{2^n}} \quad \Rightarrow \quad \alpha_n = \arctan \sqrt{\frac{x_n}{2^n}}.$$

Die Rekursionsformel besagt dann

$$\frac{x_{n+1}}{2^{n+1}} = \tan \alpha \cdot \tan \frac{\alpha}{2} = \sqrt{\frac{x_n}{2^n}} \tan\left(\frac{1}{2} \arctan \sqrt{\frac{x_n}{2^n}}\right)$$

oder

$$x_{n+1} = 2^{n+1} \sqrt{\frac{x_n}{2^n}} \tan\left(\frac{1}{2} \arctan \sqrt{\frac{x_n}{2^n}}\right). \quad (1.14)$$

In dieser Form ist die Iteration ohne Auslöschung durchführbar, wie man in Tabelle 1.13 sehen kann. Allerdings benötigt die Iteration jetzt zusätzlich die Auswertung eines Arkustangens und eines Tangens, was den Rechenaufwand beträchtlich erhöht, selbst auf modernen Floatingpoint-Hardware, die diese Operationen sehr schnell ausführen kann. \circ

2

Gleichungen lösen

Im Januar 1535 stellten sich Niccolò Tartaglia und Antonio Maria Fior öffentlich je 30 kubische Gleichungen mit der Form $x^3 + px = q$ oder $x^3 = px + q$ und forderten sich gegenseitig heraus, diese Gleichungen innert 50 Tagen zu lösen. Für moderne Leser scheint es zwischen diesen Gleichungen keinen Unterschied zu geben, aber negative Zahlen waren damals noch nicht in Gebrauch. Fior war ein Schüler von Scipione dal Ferro, der ein Lösungsmethode für einige Typen dieser kubischen Gleichungen aufgestellt hatte. Tartaglia strengte sich darauf hin besonders an und fand am 12. Februar 1535 eine Lösungsformel für beide Typen und am darauffolgenden Tag auch eine für die Gleichung $x^3 + q = px$.

Der Wettbewerb ging sehr ungleich aus: mit seiner Lösungsformel konnte Tartaglia alle gestellten Aufgaben lösen, während Fior keine einzige lösen konnte. Solche öffentlichen Wettbewerbe unter Gelehrten waren in der Renaissance durchaus üblich, sie waren Teil des Marketings, mit dem Gelehrte bekannt werden und neue Kunden gewinnen konnten. Tartaglia zum Beispiel verdiente seinen Lebensunterhalt vorwiegend als kaufmännischer Rechner und Privatlehrer. Tartaglia ist auch der Autor eines Buches über Ballistik, seine mathematischen Forschungen waren durchaus auch von konkreten Anwendungen motiviert.

Die Lösung der kubischen Gleichung durch Tartaglia und die spätere Verallgemeinerung durch Gerolamo Cardano (1501–1576) sowie die Lösung der Gleichung vierten Grades durch Lodovico Ferrari (1522–1565) waren Lösungsformeln wie sie heute jeder Schüler für die quadratische Gleichung kennelernt. Wie sieht die Lösungsformel für allgemeine Gleichungen fünften Grades aus? Die überraschende Antwort gab 1824 Niels Henrik Abel, er zeigte, dass es eine allgemeine Lösungsformel nicht geben kann. Dies war eines der ersten Resultate in einer langen Reihe von Unmöglichkeitsaussagen. So wissen wir zum Beispiel heute, dass die Stammfunktion der Funktion e^{-x^2} keine analytische Darstellung mit Hilfe von Potenzfunktionen, Brüchen, Exponential- und Logarithmusfunktionen hat. Es gibt sogar einen Algorithmus¹ von Risch, mit dem man entscheiden kan, ob eine solche Darstellung für einen vorgegebenen Integranden möglich ist.

Diese Beispiel zeigen uns, dass die Lösung einer Gleichung mit einer Lösungsformel eher die Ausnahme als die Regel darstellt. Gefragt sind daher numerische Methoden, die Gleichungen effizient und zuverlässig lösen können. Dieses Kapitel befasst sich mit den besonderen Schwierigkeiten dieser Aufgabenstellung.

¹Eigentlich handelt es sich um einen Pseudo-Algorithmus, denn einzelne Schritte des Algorithmus verlangen, dass entschieden werden muss, ob zwei Terme identisch sind. Auch dies ist ein Problem, welches von einem Computer nicht in voller Allgemeinheit gelöst werden kann, allerdings aus ganz anderem Grund.

2.1 Nullstellen von Funktionen

Die Aufgabe, eine Gleichung der Form $f(x) = g(x)$ zu lösen, also ein $x \in \mathbb{R}$ zu finden derart, dass die Gleichung erfüllt wird, ist gleichbedeutend damit, eine Nullstelle der Funktion $f(x) - g(x)$ zu finden. Es ist also gar nicht nötig, allgemeine Gleichungslösungsverfahren zu entwickeln, es reicht völlig aus, Nullstellen finden zu können.

Es muss davon ausgegangen werden, dass die Funktion f nicht einfach algebraisch invertiert werden kann. Sie wird also als Black-Box behandelt, man kann mit ihr nur Funktionswerte zu vorgegebenen x ermitteln. Bei einem Versuch mit einem Wert x_0 gibt der Funktionswert $f(x_0)$ nur die Information, ob der Versuch erfolgreich war oder nicht. Grundsätzlich können wir nicht einmal schliessen, dass ein grosser Funktionswert bedeutet, dass x_0 weit von einer Nullstelle entfernt liegt. Dazu sind weitere Annahmen über die Funktion notwendig.

In diesem Abschnitt untersuchen wir, wie verschiedene ergänzende Annahmen über die Funktion f die Möglichkeiten erweitern, Nullstellen effizient zu finden. In keinem Fall werden wir allerdings Differenzierbarkeit von f voraussetzen, dies ist Abschnitt 2.2 vorbehalten, wo das Newton-Verfahren behandelt wird..

2.1.1 Intervallhalbierung

Ist die Funktion stetig, sagt die Grösse eines Funktionswertes immer noch nichts darüber aus, wie weit das Argument von der Nullstelle entfernt ist. Zwei Werte mit unterschiedlichem Vorzeichen zeigen dagegen klar an, dass sich eine Nullstelle zwischen Argumenten befinden muss. Dies ist der Inhalt des folgenden Spezialfalls des Zwischenwertsatzes.

Satz 2.1 (Zwischenwertsatz für Nullstellen). *Ist die Funktion $f: [a, b] \rightarrow \mathbb{R}$ stetig mit $f(a) < 0$ und $f(b) > 0$, dann hat f eine Nullstelle im Inneren des Intervalls $[a, b]$.*

Man beachte, dass der Betrag der Funktionswerte an den Intervallenden keine Information darüber liefert, wo im Intervall die Nullstelle zu finden ist. Abbildung 2.1 zeigt zwei Funktionen mit identischen Funktionswerten an den Intervallenden aber völlig verschiedener Lage der Nullstellen. Erst zusätzliche Annahmen über die Steigung oder Krümmung der Kurve können die Lage der Nullstelle besser eingrenzen.

Der Zwischenwertsatz 2.1 liefert trotzdem genug Information, um die Nullstelle zu finden. Für die folgende Diskussion nehmen wir der Einfachheit an, dass $f(a) < 0$ und $f(b) > 0$ ist. Wir wissen bereits, dass die Nullstelle im Intervall $[a, b]$ liegen muss. Sei $m = \frac{1}{2}(a + b)$ der Mittelpunkt des Intervalls. Wenn $f(m) > 0$ ist, können wir schliessen, dass eine Nullstelle im Teilintervall $[a, m]$ liegen muss. Wenn $f(m) < 0$ ist, liegt eine Nullstelle in $[m, b]$. Damit ist ein neues Intervall halber Länge gefunden, welches eine Nullstelle von f enthält. Wir haben ein Bit Information über die Nullstelle gewonnen. Durch Wiederholen dieses Prozesses können wir ein beliebige kleines Intervall erhalten, welches eine Nullstelle enthält. Nach dem Intervallschachtelungsprinzip der Analysis definiert dies die Nullstelle.

Satz 2.2 (Intervallhalbierung). *Sei $f: [a, b] \rightarrow \mathbb{R}$ eine stetige Funktion mit $f(a) < 0$ und $f(b) > 0$. Sei $I_k = [a_k, b_k]$ die Folge von Intervallen rekursiv definiert wie folgt.*

1. Das Startintervall hat Intervallenden $a_0 = a$ und $b_0 = b$.

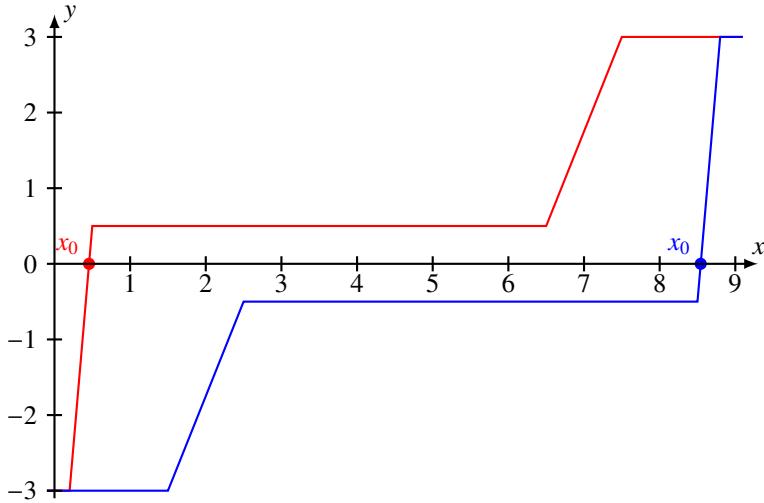


Abbildung 2.1: Die Werte einer stetigen Funktion an den Intervallenden verraten nichts über die Lage einer Nullstelle im Intervall. Die beiden Graphen gehören zu Funktionen, die den gleichen Wert an den Intervallenden haben, aber die Nullstelle x_0 liegt an völlig unterschiedlichen Stellen

2. Sei $m_k = \frac{1}{2}(a_k + b_k)$. Das Intervall I_{k+1} ist

$$I_{k+1} = [a_{k+1}, b_{k+1}] = \begin{cases} [a_k, m_k] & f(m_k) > 0 \\ [m_k, b_k] & f(m_k) < 0. \end{cases}$$

Die Intervalle I_k haben die folgenden Eigenschaften

1. Die Länge der Intervalle halbiert sich in jeder Iteration: $|I_k| = 2^{-k}(b - a)$.
2. Für jedes Intervall gilt $f(a_k) < 0$ und $f(b_k) > 0$.
3. Die Schnittmenge $\bigcap_{k=0}^{\infty} I_k$ enthält nur den Wert

$$x_0 = \lim_{k \rightarrow \infty} a_k = \lim_{k \rightarrow \infty} b_k = \lim_{k \rightarrow \infty} m_k, \quad (2.1)$$

der eine Nullstelle der Funktion f ist.

Beweis. Es ist nur noch die Aussage (2.1) über die Konvergenz der Folgen a_k und b_k zu beweisen. Da aber die Intervall-Länge $|I_k| = 2^{-k}(b - a)$ ist, kann man die Entfernung der Folgenglieder voneinander abschätzen. Ist $k = \min\{m, n\}$, dann folgt

$$|a_m - a_n| < 2^{-k}(b - a), \quad |b_m - b_n| < 2^{-k}(b - a), \quad |m_m - m_n| < 2^{-k}(b - a).$$

Wählt man N so gross, dass $2^{-N}(b - a) < \varepsilon$ ist, dann folgt für jedes beliebige $\varepsilon > 0$, dass

$$|a_m - a_n| < \varepsilon, \quad |b_m - b_n| < \varepsilon, \quad |m_m - m_n| < \varepsilon$$

ist. Die Folgen sind also Cauchy-Folgen. □

k	a_k	b_k	$b_k - a_k$
0	0.00000000	2.00000000	2.00000000
1	1.00000000	2.00000000	1.00000000
2	1.00000000	1.50000000	0.50000000
3	1.00000000	1.25000000	0.25000000
4	1.00000000	1.12500000	0.12500000
5	1.00000000	1.06250000	0.06250000
6	1.00000000	1.03125000	0.03125000
7	1.01562500	1.03125000	0.01562500
8	1.01562500	1.02343750	0.00781250
9	1.01953125	1.02343750	0.00390625
10	1.02148438	1.02343750	0.00195312
11	1.02246094	1.02343750	0.00097656
12	1.02294922	1.02343750	0.00048828
13	1.02319336	1.02343750	0.00024414
14	1.02319336	1.02331543	0.00012207
15	1.02325439	1.02331543	0.00006104
16	1.02328491	1.02331543	0.00003052
17	1.02328491	1.02330017	0.00001526
18	1.02329254	1.02330017	0.00000763
19	1.02329254	1.02329636	0.00000381
20	1.02329254	1.02329445	0.00000191
21	1.02329254	1.02329350	0.00000095
22	1.02329254	1.02329302	0.00000048
23	1.02329278	1.02329302	0.00000024
24	1.02329290	1.02329302	0.00000012

Tabelle 2.1: Bestimmung von $\sqrt[100]{10}$ mit Hilfe des Intervallhalbierungsverfahrens. Die jeweils neue Intervallgrenze ist rot markiert.

Die Konvergenzgeschwindigkeit des Intervall-Halbierungs-Verfahrens ist sehr beschränkt. Der Fehler halbiert sich in jeder Iteration, man gewinnt also genau 1 Bit Genauigkeit. Für die volle Genauigkeit eines Gleitkomma-Typs der Maschine braucht man also etwa soviele Iterationen, wie die Mantisse Binärstellen hat, aber auch nur, wenn das erste Bit und der Exponent bereits richtig sind.

Beispiel. Als Beispiel berechnen wir $\sqrt[100]{10}$. Die Potenzfunktion $p(x) = x^{100}$, die wir dazu invertieren müssen, hat extrem grosse Steigung in der Nähe der Lösung. Es muss eine Nullstelle der Funktion $f(x) = x^{100} - 10$ gefunden werden. Wegen $f(0) = -10$ und $f(2) \approx 1.2677 \cdot 10^{30}$ muss das Intervall $[0, 2]$ eine Nullstelle enthalten. Weil die Funktion monoton wachsend ist, ist es auch die einzige. Der Wert $f(2)$ ist klein genug für den float Typ, daher kann man die Berechnung in float durchführen.

Tabelle 2.1 zeigt den Gang der Berechnung. Die Implementation in C++ verwendet als Abbruchkriterium die Länge des Intervalls. Die Iteration endet, wenn die Intervalllänge $b_k - a_k$ den ε -Wert erreicht, der für den Typ float im Header limits definiert ist. Wie erwartet sind soviele Iterationen nötig, wie die Mantisse des float-Typs Bits hat.

Das Programm `nullstellen.cpp`² implementiert den Algorithmus für jeden Maschinentypen. Damit kann man sehen, dass die berechnete Laufzeit auch für die grösseren Gleitkommatypen durch die Bitlänge der Mantisse gegeben ist.



2.1.2 Sekantenverfahren

Das Intervallhalbierungsverfahren verwendet nicht mehr als die Stetigkeit der Funktion. Dies führt zu sehr langsamer Konvergenz, weil die relative Grösse der Funktionswerte in den Intervallenden keine Information über die Lage der Nullstelle im Intervall gegen kann. Dazu wird Information darüber benötigt, wie schnell sich Funktionswerte ändern können. Insbesondere muss davon ausgegangen werden können, dass die Steigung der Funktion über das betrachtete Intervall nicht zu stark schwankt. Die Ableitung einer differenzierbaren Funktion kann diese Information liefern, es ist aber ausreichend, eine Lipschitz-Bedingung zu verlangen, die wie folgt definiert ist.

Definition 2.3. Die Funktion $f: [a, b] \rightarrow \mathbb{R}$ erfüllt eine Lipschitz-Bedingung zum Exponenten α , wenn es eine Konstante L gibt derart, dass

$$|f(x) - f(y)| < L|x - y|^\alpha$$

ist.

Eine Lipschitz-Bedingung begrenzt, wie schnell sich der Wert einer Funktion ändern kann. Eine stetig differenzierbare Funktion erfüllt automatisch eine Lipschitz-Bedingung für $\alpha = 1$, die Umkehrung gilt allerdings nicht.

Sei jetzt $f: [a, b] \rightarrow \mathbb{R}$ wieder eine Funktion mit $f(a) < 0$ und $f(b) > 0$. Wenn f eine Lipschitz-Bedingung erfüllt, dann geben die Werte $f(a)$ und $f(b)$ zusätzlich Information über die Lage der Nullstelle, die im Intervall liegen muss. Da sich Funktionswerte nicht beliebig schnell ändern können, dürfte die Nullstelle näher beim kleineren Funktionswert sein.

Nehmen wir an, die Funktion f sei linear zwischen den beiden Funktionswerten, dann ist die Nullstelle der Schnittpunkt der Geraden durch $(a, f(a))$ und $(b, f(b))$ mit der x -Achse (Siehe auch Abbildung 2.2). Der Strahlensatz zeigt

$$\begin{aligned} (a - x_0) : f(a) &= (b - x_0) : f(b) \\ \Leftrightarrow a f(b) - b f(a) &= x_0(f(b) - f(a)) \\ \Rightarrow x_0 &= \frac{a f(b) - b f(a)}{f(b) - f(a)}. \end{aligned}$$

Man kann diese Formel für x_0 auch bekommen als mit den Funktionswerten gewichtetes Mittel der Endpunkte wie folgt. Das Gewicht m_a von a ist $f(b)$, das Gewicht m_b von b ist $-f(a)$, hier verwenden wir $f(a) < 0$. Das gewichtete Mittel ist dann

$$\frac{a m_a + b m_b}{m_a + m_b} = \frac{a f(b) - b f(a)}{f(b) - f(a)} = x_0. \quad (2.2)$$

Indem die Intervallhalbierung durch eine Teilung des Intervalls im Punkt x_0 ersetzt wird, kann ein neuer Algorithmus gewonnen werden, der möglicherweise schneller konvergiert, weil Intervalle schneller kleiner werden können. Allerdings deutet die Formel (2.2) auf ein weiteres Problem hin. Ist der Nenner sehr klein, kann die neue Approximation x_0 weit entfernt sein von a und b .

²Das Programm kann im Verzeichnis `buch/chapters/experiments/nullstellen` von [4] gefunden werden.

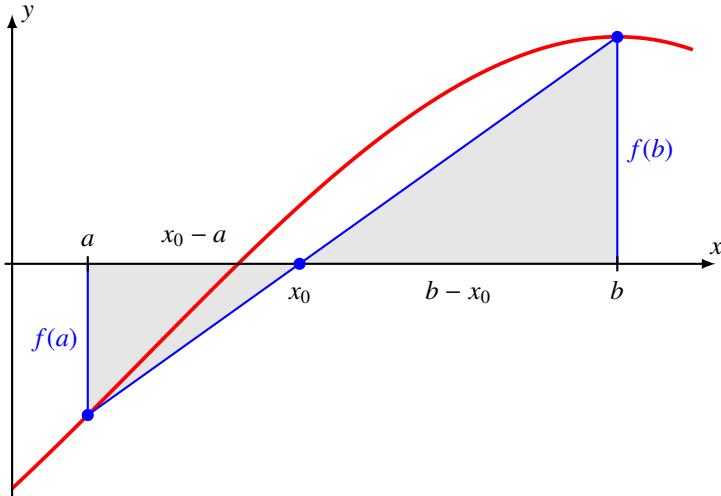


Abbildung 2.2: Bestimmung einer neuen Schätzung x_0 für die Nullstelle mit Hilfe der Sekante. Nach dem Strahlensatz ist $(a - x_0) : f(a) = (b - x_0) : f(b)$, woraus sich x_0 bestimmen lässt (siehe auch (2.2)).

Es ist leider auch nicht so, dass das neue Verfahren unbedingt schneller konvergiert. Im Intervallhalbierungsverfahren ist sichergestellt, dass das neue Intervall höchstens halb so gross ist. Eine solche Garantie gibt es bei Verwendung der Formel (2.2) nicht. Ist zum Beispiel im Intervall $[a, b]$ die erste Ableitung $f'(x) > 0$ und die zweite Ableitung $f''(x) < 0$, dann ist der neue Teilpunkt immer rechts von der Nullstelle. Das neue Intervall hat daher immer den gleichen linken Endpunkt a , die Folge a_k konvergiert nicht.

Beispiel. Die Funktion $f(x) = \sin x - \frac{1}{2}$ hat im Intervall $[0, \frac{\pi}{2}]$ positive Steigung und negative zweite Ableitung. Wie erwartet bleibt a_k konstant, aber b_k konvergiert monoton gegen die Nullstelle. In Tabelle 2.2 sind die Resultate der Rechnung gezeigt. Die Werte der Teilpunkte m_k konvergiert linear gegen den gesuchten Wert $\arcsin \frac{1}{2}$. \circ

Ein gut konvergentes Verfahren kann man also nur erwarten, wenn man sicherstellen kann, dass sich die Funktion im Intervall merklich ändert, dass als $f(x) - f(y)$ nicht zu klein wird. Dies wird erreicht, wenn f eine *untere Lipschitz-Bedingung* erfüllt. Dies bedeutet, dass es eine Zahl $l > 0$ gibt derart, dass

$$|f(x) - f(y)| \geq l \cdot |x - y|^\alpha.$$

Solche Lipschitz-Bedingungen sind automatisch erfüllt, wenn eine Funktion stetig differenzierbar ist, dies ist auch für die Praxis der übliche Fall. Daraus ergibt sich jetzt der folgende Algorithmus.

Algorithmus 2.4 (Sekantenverfahren). *Sei $f: [a, b] \rightarrow \mathbb{R}$ eine stetig differenzierbare Funktion mit $f(a) < 0$ und $f(b) > 0$. Man setzt $x_{-1} = a$ und $x_0 = b$ und konstruiert die Folge*

$$x_{n+1} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}. \quad (2.3)$$

Falls a und b nahe genug bei einer Nullstelle der Funktion f ist, dann konvergiert die Folge x_{n+1} gegen die Nullstelle.

k	m_k
-1	0.00000000
0	1.50000000
2	0.55041450
3	0.52616811
4	0.52383864
5	0.52362108
6	0.52360088
7	0.52359897
8	0.52359879

Tabelle 2.2: Sekantenverfahren zur Bestimmung von $\arcsin \frac{1}{2}$. Die Konvergenz ist ungefähr linear, aber deutlich schneller als beim Intervallhalbierungsverfahren.

Leider garantieren die Bedingungen die Konvergenz nicht unbedingt. Es zeigt sich aber, dass das Verfahren fast immer konvergiert, wenn $f'(x)$ im Suchintervall nicht zu stark schwankt und dem Wert 0 nicht nahe kommt.

Die Iterationsformel (2.3) hat den gravierenden Nachteil, dass in der Nähe der Lösung die beiden Grössen im Zähler und im Nenner fast gleich gross sind und damit starke Auslöschung auftreten wird. Die Umformung

$$x_{n+1} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})} = \frac{x_{n-1}f(x_n) - x_{n-1}f(x_{n-1}) + x_{n-1}f(x_{n-1}) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})} \quad (2.4)$$

$$= x_{n-1} - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \quad (2.5)$$

ist algebraisch identisch, jedoch wird in der letzten Form die Approximation x_{n-1} um einen Betrag korrigiert, der umso kleiner wird, je besser x_{n-1} die Nullstelle bereits approximiert und damit je kleiner der Faktor $f(x_{n-1})$ ist. Der Bruch in (2.5) kann jedoch immer noch erratische Werte annehmen und damit die Konvergenz des Verfahrens gefährden.

Man beachte, dass die Bedingung an die Vorzeichen von $f(a)$ und $f(b)$ nur dazu dient, die Existenz einer Nullstelle im Intervall zu garantieren. Ein anderes Problem dieses Verfahrens ist, dass mit genauer werdender Approximation x_n der Nullstelle die Werte $f(x_n)$ und $f(x_{n-1})$ sehr nahe beieinander liegen und damit die Differenz stark von Auslöschung betroffen sein wird. Im Intervallhalbierungsalgorithmus ist dies kein Problem, weil keine Differenzen von Funktionswerten gebildet werden und ausschliesslich das Vorzeichen des Funktionswertes verwendet wird, die Genauigkeit hat keinen Einfluss auf das Vorzeichen.

2.2 Newton-Verfahren

Die bisher vorgestellten Verfahren zur Bestimmung einer Nullstelle x^* der Funktion f , $f(x^*) = 0$ sind linear konvergent und damit eher langsam. Dafür war nicht mehr als Stetigkeit nötig, um Konvergenz des Verfahrens sicherzustellen.

2.2.1 Analytischer Ansatz für ein quadratisch konvergentes Verfahren

In Abschnitt 1.4.1 wurde dargestellt, dass nach Möglichkeit quadratische Konvergenz angestrebt werden sollte. Quadratische Konvergenz könnte in einer Fixpunktiteration $x_{n+1} = g(x_n)$ erreicht werden, wenn die Ableitung $g'(x^*) = 0$ ist.

Je grösser $f(x_n)$ ist, desto weiter dürfte x_n von der Nullstelle entfernt sein. Wir versuchen daher, die Approximation x_{n+1} proportional zum Wert von $f(x_n)$ zu korrigieren mit Hilfe der Funktion

$$x_{n+1} = g(x_n) = x_n - a(x_n) \cdot f(x_n).$$

Die Funktion $a(x_n)$ muss noch bestimmt werden. Sie soll so gewählt werden, dass die Konvergenz quadratisch wird, was mit $g'(x^*) = 0$ erreicht wird.

Die Ableitung von g ist

$$g'(x) = 1 - a'(x)f(x) - a(x)f'(x).$$

An der Stelle x^* gilt

$$\begin{aligned} g'(x^*) &= 1 - a'(x^*) \underbrace{f(x^*)}_{= 0} - a(x^*)f'(x^*) = 0 \\ \Leftrightarrow 1 &= a(x^*)f'(x^*) \quad \Rightarrow \quad a(x) = \frac{1}{f'(x)}. \end{aligned}$$

Damit finden wir das im folgenden Satz beschriebene Verfahren mit quadratischer Konvergenz.

Satz 2.5 (Newton-Verfahren). *Hat die differenzierbare Funktion f eine Nullstelle x^* und gilt $f'(x^*) \neq 0$, dann konvergiert die Iterationsfolge*

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.6}$$

für Startwerte x_0 genügend nahe bei x^* quadratisch gegen die Nullstelle.

Das Sekantenverfahren hat darunter gelitten, dass die Berechnung der nächsten Approximation mit Hilfe der Formel (2.3) von umso stärker Auslöschung geplagt ist, je näher man bereits an der Lösung ist. Die Iterationsformel (2.6) für die Newton-Iteration hat dieses Problem nicht. In (2.6) wird der aktuelle Wert x_n um einen Korrekturbetrag korrigiert, der proportional zu $f(x_n)$ verkleinert wird. Je genauer der Wert x_n schon ist, desto kleiner wird auch die Korrektur.

2.2.2 Geometrische Interpretation des Newton-Verfahrens

Die Iterationsformel (2.6) lässt sich sehr schön graphisch interpretieren. In Abbildung 2.3 wird die Nullstelle der Funktion $f(x) = e^x - \frac{1}{2}$ mit dem Newton-Verfahren bestimmt. Im Iterationsschritt wird die Approximation x_n korrigiert nach der Formel

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{e^{x_n} - \frac{1}{2}}{e^{x_n}} = x_n - \left(1 - \frac{e^{-x_n}}{2}\right)$$

Die Korrektur für $n = 0$ ist in Abbildung 2.3 als Grundseite des rechtwinkligen Dreiecks ABC erkennbar. Die Hypotenuse hat die Steigung $f'(x_0)$, daher ist

$$\overline{AC} \cdot f'(x_0) = f(x_0) \quad \Rightarrow \quad \overline{AC} = \frac{f(x_0)}{f'(x_0)}.$$

Die vom Newton-Verfahren berechnete Korrektur ist also die optimale Korrektur, die sich aus Funktionswert und erster Ableitung an der Stelle x_n berechnen lässt.

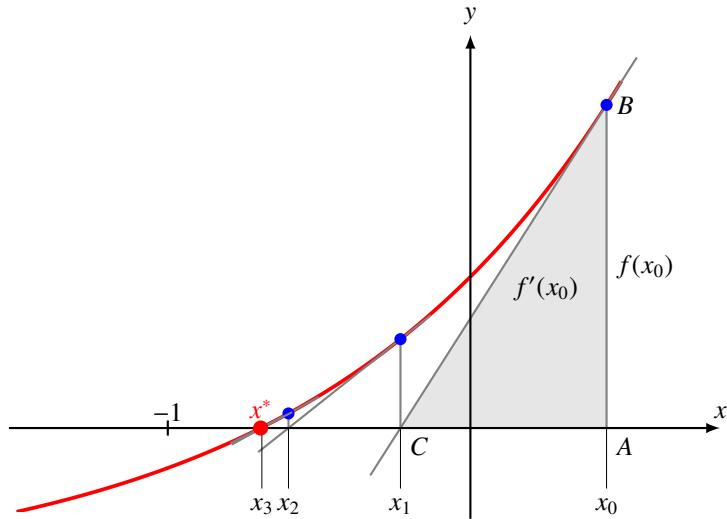


Abbildung 2.3: Graphische Interpretation des Newton-Verfahrens. In jedem Iterationsschritt wird die bisherige Approximation A mit Hilfe einer Tangente vom Funktionswert B zum Punkt C korrigiert, $\overline{AC} = f(x_0)/f'(x_0)$.

2.2.3 Wurzeln

Als Beispiel für die Anwendung des Newton-Verfahrens berechnen wir die k -te Wurzel einer positiven reellen Zahl a , wir lösen also die Gleichung $x^k = a$. Dies ist gleichbedeutend damit, eine Nullstelle der Funktion $f(x) = x^k - a$ zu bestimmen. Die Ableitung von f ist $f'(x) = kx^{k-1}$, woraus wir die Iterationsformel des Newton-Verfahrens ablesen können:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^k - a}{kx_n^{k-1}} = \frac{1}{k} \left((k-1)x_n + \frac{a}{x_n^{k-1}} \right).$$

Im Falle $n = 2$ finden wir das bereits in Abschnitt 1.4.1 untersuchte, quadratisch konvergente Verfahren zur Bestimmung der Quadratwurzel wieder. In Tabelle 2.3 ist das Verfahren für $a = 10$, $k = 5$ und $x_0 = a$ gezeigt. Quadratische Konvergenz stellt sich allerdings erst bei x_{10} ein, der Startwert x_0 ist zu weit von der Lösung entfernt.

2.2.4 Newton-Verfahren in \mathbb{R}^n

In der bisher beschriebenen Form erlaubt das Newton-Verfahren, Nullstellen von reellwertigen Funktion zu finden. Es eignet sich nicht, Vektorgleichungen zu lösen.

Sei daher im folgenden $f: \mathbb{R}^k \rightarrow \mathbb{R}^k$ eine Vektorfunktion mit einer Nullstelle x^* , $f(x^*) = 0$, die numerisch gefunden werden soll. Wie in Abschnitt 2.2.1 soll eine Approximation x_n für die Nullstelle x^* proportional zur Grösse von $f(x_n)$ korrigiert werden. Eine skalare Funktion $a(x)$ wird aber im allgemeinen zu wenig allgemein für eine performante Lösung des Problem sein, daher wird für $A(x)$ eine matrixwertige Funktion mit Werten in $GL_k(\mathbb{R})$ gewählt. Wir setzen also an

$$x_{n+1} = g(x_n) = x_n - A(x_n)f(x_n)$$

und versuchen wie früher $A(x)$ so zu wählen, dass die Ableitung von g an der Stelle x^* verschwindet.

n	x_n
0	10.000000000000000
1	8.000200000000000
2	6.40064823242493
3	5.12171019598693
4	4.10027465454472
5	3.28729556684556
6	2.64696320430731
7	2.15831219143923
8	<u>1.81881622015378</u>
9	<u>1.63781027109793</u>
10	<u>1.58820394873794</u>
11	<u>1.58490696686523</u>
12	<u>1.58489319270054</u>
13	<u>1.58489319246111</u>
∞	$\sqrt[5]{10} = 1.58489319246111$

Tabelle 2.3: Berechnung von $\sqrt[5]{10}$ mit dem Newton-Verfahren. Der Startwert $x_0 = 10$ ist sehr weit von der Lösung entfernt, so dass es einige Iterationen braucht, bis die Konvergenz quadratisch wird.

Die Ableitung von $g(x) = x - A(x)f(x)$ ist eine lineare Abbildung, die auf dem Vektor $h \in \mathbb{R}^k$ den Wert

$$Dg(x) \cdot h = h - (DA(x) \cdot h)f(x) - A(x)Df(x) \cdot h$$

hat. An der Stelle $x = x^*$ verschwindet der mittlere Term wegen $f(x^*) = 0$, so dass als Gleichung für $A(x)$

$$0 = h - A(x)Df(x) \cdot h \quad \Rightarrow \quad h = A(x)Df(x) \cdot h \quad \forall h \in \mathbb{R}^n$$

übrigbleibt. Dies ist nur möglich, wenn $A(x)$ die inverse Matrix von $Df(x)$ ist, was den folgenden Satz motiviert.

Satz 2.6 (Newton-Verfahren für Vektorgleichungen). *Hat die Funktion $f: \mathbb{R}^k \rightarrow \mathbb{R}^k$ eine Nullstelle $x^* \in \mathbb{R}^n$, dann ist die Folge*

$$x_{n+1} = x_n - Df(x_n)^{-1} \cdot f(x_n)$$

für Startwerte x_0 nahe genug an x^ quadratisch konvergent mit Grenzwert x^* .*

Beweis. Es ist klar, das x^* ein Fixpunkt der Abbildung

$$g(x) = x - Df(x)^{-1} \cdot f(x)$$

ist. Wir müssen nur noch zeigen, dass der Fehler der Iteration quadratisch abnimmt. Dazu entwickeln wir f um den Punkt x^* in eine Taylor-Reihe

$$f(x^* + \delta) = f(x^*) + Df(x^*) \cdot \delta + O(|\delta|^2), \quad \delta \in \mathbb{R}^k$$

$$= Df(x^*) \cdot \delta + O(|\delta|^2)$$

wegen $f(x^*) = 0$. Die Iteration ist

$$\begin{aligned} x_{n+1} &= x^* + \delta_{n+1} = g(x^* + \delta_n) = x^* + \delta_n - Df(x_n)^{-1} \cdot f(x_n) \\ &= x^* + \delta_n - Df(x_n)^{-1} \cdot f(x^* + \delta_n) \\ &= x^* + \delta_n - Df(x_n)^{-1} \cdot (Df(x^*) \cdot \delta_n + O(|\delta_n|^2)). \end{aligned} \quad (2.7)$$

Um (2.7) zu berechnen, muss man auch $Df(x_n)$ in eine Taylor-Reihe entwickeln, sie ist

$$Df(x_n) = Df(x^* + \delta_n) = Df(x^*) + D^2 f(x^*) \cdot \delta_n.$$

Setzt man dies in (2.7) ein, erhält man

$$\begin{aligned} x_{n+1} &= x^* + \delta_{n+1} = x^* + \delta_n - (Df(x^*) + D^2 f(x^*) \cdot \delta_n)^{-1} (Df(x^*) \cdot \delta_n + O(|\delta_n|^2)) \\ &= x^* + \delta_n - (Df(x^*)^{-1} + O(|\delta_n|)) \cdot (Df(x^*) \cdot \delta_n + O(|\delta_n|^2)) \\ &= x^* + \delta_n - \underbrace{Df(x^*)^{-1} Df(x^*)}_{= E} \cdot \delta_n + O(|\delta_n|^2) \\ &= x^* + O(|\delta_n|^2). \end{aligned}$$

Der Fehler $\delta_{n+1} = O(|\delta_n|^2)$ nimmt somit quadratisch ab und damit ist gezeigt, dass die Iterationsfolge quadratisch konvergiert. \square

Beispiel. Es sollen die Polarkoordinaten des Punktes (x, y) als Lösung der Gleichung

$$\begin{pmatrix} r \cos \varphi \\ r \sin \varphi \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \Rightarrow \quad f(r, \varphi) = \begin{pmatrix} r \cos \varphi - x \\ r \sin \varphi - y \end{pmatrix} = 0$$

bestimmt werden. Dies ist ein Beispiel für die Dimension $k = 2$.

Die Ableitungsmatrix von f ist

$$Df(r, \varphi) = \begin{pmatrix} \cos \varphi & -r \sin \varphi \\ \sin \varphi & r \cos \varphi \end{pmatrix} \quad \Rightarrow \quad Df(r, \varphi)^{-1} = \begin{pmatrix} \cos \varphi & \sin \varphi \\ -\frac{1}{r} \sin \varphi & \frac{1}{r} \cos \varphi \end{pmatrix}.$$

Die Iterationsformel wird jetzt

$$\begin{aligned} \begin{pmatrix} r_{n+1} \\ \varphi_{n+1} \end{pmatrix} &= \begin{pmatrix} r_n \\ \varphi_n \end{pmatrix} - \begin{pmatrix} \cos \varphi_n & \sin \varphi_n \\ -\frac{1}{r_n} \sin \varphi_n & \frac{1}{r_n} \cos \varphi_n \end{pmatrix} \begin{pmatrix} r_n \cos \varphi_n - x \\ r_n \sin \varphi_n - y \end{pmatrix} \\ &= \begin{pmatrix} x \cos \varphi_n + y \sin \varphi_n \\ \varphi_n - \frac{x}{r_n} \sin \varphi_n + \frac{y}{r_n} \cos \varphi_n \end{pmatrix} \end{aligned} \quad (2.8)$$

Die Resultate der Iteration (2.8) für den Punkt $(x, y) = (\cos 1, \sin 1)$ ist in Tabelle 2.4 gegeben. Die quadratische Konvergenz ist wieder deutlich erkennbar. \circlearrowright

2.2.5 Der Fall $f'(x^*) = 0$

Im Fall $f'(x^*) = 0$ versagt die Iterationsformel des Newton-Verfahrens. Es ist damit zu rechnen, dass das Verfahren sehr langsam oder gar nicht konvergiert. Wir untersuchen dies mit Hilfe einer Entwicklung der Funktion f um den Punkt x^* :

$$f(x^* + \delta) = x^* + \frac{1}{2} f''(x^*) \delta^2 + \frac{1}{6} f'''(x^*) \delta^3 + O(\delta^4).$$

n	r_n	φ_n
0	3.1415926535897931	0.3678794411714423
1	0.8067763763745672	0.5559550343664228
2	0.9030213557712843	1.0884392177149671
3	0.9960918007116625	0.9906298199545209
4	0.9999561001841604	1.0000366265580796
5	0.999999993292477	0.999999983920385
6	1.0000000000000000	1.0000000000000000
∞	1.0000000000000000	1.0000000000000000

Tabelle 2.4: Quadratische Konvergenz des Iterationsverfahrens (2.8) zur Bestimmung der Polarkoordinaten

Für den Fehler δ_n der Approximation $x_n = x^* + \delta_n$ folgt die Iteration

$$\begin{aligned}
x^* + \delta_{n+1} &= x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x^* + \delta_n - \frac{\frac{1}{2}f'''(x^*)\delta_n^2 + O(\delta_n^4)}{f''(x^*)\delta_n + \frac{1}{2}f'''(x^*)\delta_n^2 + O(\delta^3)} \\
&= x^* + \delta_n - \frac{1}{2}\delta_n \frac{1 + O(\delta_n)}{1 + O(\delta_n)} = x^* + \delta_n - \frac{1}{2}\delta_n(1 + O(\delta_n)) \\
&= x^* + \frac{1}{2}\delta_n + O(\delta_n^2) \\
\Rightarrow \quad \delta_{n+1} &= \frac{1}{2}\delta_n + O(\delta_n^2)
\end{aligned}$$

Der Fehler halbiert sich in jeder Iteration. Die Folge $(x_n)_{n \in \mathbb{N}}$ konvergiert also immer noch, aber die Konvergenz ist nur noch linear.

2.2.6 Vergleich mit dem Sekantenverfahren

Die Ähnlichkeit des Newton-Verfahrens mit dem Sekantenverfahren ist unübersehbar. Um dies deutlich zu machen, berechnen wir den Grenzfall $x_{n-1} \rightarrow x_n$ mit Hilfe der Form (2.5). Beim Grenzübergang $x_{n-1} \rightarrow x_n$ geht der Quotient auf der rechten Seite in den Kehrwert der Ableitung $f'(x_n)$ über. Der Grenzfall des Sekantenverfahrens ist daher

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

also das Newton-Verfahren.

Der Vorteil des Newton-Verfahrens gegenüber dem Sekantenverfahren ist jedoch, dass die Ableitung nicht nur mit Hilfe eines Differenzenquotienten approximiert wird, sondern exakt zur Verfügung steht. Damit ist das Newton-Verfahren nicht anfällig auf die Auslöschung, die die Zuverlässigkeit des Sekantenverfahrens beeinträchtigt.

2.2.7 Nullstellen von Polynomen

Das Newton-Verfahren verlangt, dass die Ableitung $f'(x_n)$ genau berechnet werden kann. In einigen Fällen kann dies ein Hindernis für die Anwendung des Verfahrens sein. Polynome sind jedoch

einfach genug, dass die Ableitung immer berechnet werden kann. Somit ist das Newton-Verfahren besonders gut geeignet, Nullstellen von Polynomen zu finden. In diesem Abschnitt sei daher

$$f(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_2 X^2 + a_1 X + a_0 \quad (2.9)$$

ein Polynom mit reellen Koeffizienten, $a_k \in \mathbb{R}$. Wir gehen davon aus, dass f eine reelle Nullstelle x^* hat und dass x_0 eine ausreichend genaue Schätzung für die Nullstelle ist.

Berechnung von Funktionswerten

Die übliche Darstellung (2.9) ist nicht die effizienteste Form zur Berechnung des Polynomwertes. Die Berechnung der Potenzen x^k für $1 \leq k \leq n$ benötigt bereits $n - 1$ Multiplikationen, dazu kommen $n - 1$ Multiplikationen mit Koeffizienten und n Additionen. Zudem besteht die Gefahr von Verschmierung.

Durch Ausklammern möglichst vieler Faktoren x findet man die Formel

$$f(x) = ((\dots((a_n x + a_{n-1})x + a_{n-2})x + \dots)x + a_1)x + a_0, \quad (2.10)$$

welche den Funktionswert in genau n Multiplikationen und n Additionen zu berechnen gestattet.

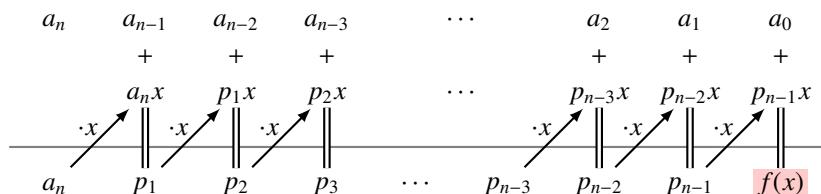
Wir bezeichnen die Teilprodukte in (2.10) mit

$$(\dots((a_n x + a_{n-1})x + a_{n-2})x + \dots)x + a_k = p_{n-k},$$

d. h.

$$\begin{aligned} p_0 &= a_n \\ p_1 &= a_n x + a_{n-1} = p_0 x + a_{n-1} \\ p_2 &= (a_n x + a_{n-1})x + a_{n-2} = p_1 x + a_{n-2} \\ &\vdots \quad \vdots \\ f(x) &= p_n = p_{n-1} x + a_0. \end{aligned} \quad (2.11)$$

Diese Berechnung lässt sich in der folgenden, *Horner-Schema* genannten Tabelle zusammenfassen.



Beispiel. Wir berechnen den Wert des Polyoms

$$f(X) = X^6 - X^5 + X^4 - X^3 + X^2 - X + 1$$

an der Stelle $X = 2$ mit Hilfe des Horner-Schemas

$$\begin{array}{rccccccccc} 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ & 2 & 2 & 6 & 10 & 22 & 42 \\ \hline 1 & 1 & 3 & 5 & 11 & 21 & 43 \end{array}$$

Der Wert in der rechten unteren Ecke stimmt überein mit

$$\begin{aligned} f(2) &= 2^6 - 2^5 + 2^4 - 2^3 + 2^2 - 2 + 1 \\ &= 64 - 32 + 16 - 8 + 4 - 2 + 1 \\ &= 32 + 8 + 2 + 1 = 43. \end{aligned}$$

○

Deflation

Die Bedeutung der Werte p_0, \dots, p_{n-1} lässt sich verstehen, wenn man den Polynomdivisionsalgorithmus für $f(X)/(X - x)$ ausschreibt. Die Rekursionsformeln (2.11) zeigen, dass die Teilreste der Division die Koeffizienten p_k haben:

$$\begin{array}{r} (a_n X^n + a_{n-1} X^{n-1} + a_{n-2} X^{n-2} + a_{n-3} X^{n-3} + \dots) : (X - x) = a_n X^{n-1} + p_1 X^{n-2} + p_2 X^{n-3} + \dots \\ \underline{a_n X^n - a_n x X^{n-1}} \\ p_1 X^{n-1} + a_{n-2} X^{n-2} \\ \underline{p_1 X^{n-1} - p_1 x X^{n-2}} \\ p_2 X^{n-2} + a_{n-3} X^{n-3} \\ \underline{p_2 X^{n-2} - a_{n-4} x X^{n-3}} \\ p_3 X^{n-3} \quad \dots \\ \dots \quad \dots \end{array} \tag{2.12}$$

Die Koeffizienten p_k sind daher auch die Koeffizienten des Quotienten

$$q(X) = p_0 X^{n-1} + p_1 X^{n-2} + p_2 X^{n-3} + \dots + p_{n-2} X + p_{n-1}.$$

Es gilt daher

$$f(X) = (X - x) \cdot (p_0 X^{n-1} + p_1 X^{n-2} + p_2 X^{n-3} + \dots + p_{n-2} X + p_{n-1}) + f(x).$$

Ist x ein Nullstelle, dann ist das Polynom $f(X)$ durch $X - x$ teilbar und $q(X)$ ist der andere Faktor, also $f(X) = (X - x) \cdot q(X)$. Das Polynom $q(X)$ hat Grad $n-1$, die Suche nach weiteren Nullstellen wird also vereinfacht. Man nennt den Prozess, eine Nullstelle aus dem Polynom $f(X)$ herauszudividieren, *Deflation*.

Beispiel. Das Polynom

$$f(x) = x^4 - 25x^2 + 144$$

hat eine Nullstelle $x = 3$ und $x = 4$ als Nullstellen. Man finde zwei weitere Nullstellen.

Die Polynomdivision mit dem Horner-Schema für $x = 3$

$$\begin{array}{r} 1 \quad 0 \quad -25 \quad 0 \quad 144 \\ \quad 3 \quad 9 \quad -48 \quad -144 \\ \hline 1 \quad 3 \quad -16 \quad -48 \quad 0 \end{array}$$

ergibt

$$q_1(x) = f(x)/(x - 3) = x^3 + 3x^2 - 16x - 48.$$

Weiter bekommt man

$$a_2(x) = f(x)/((x - 3)(x - 4)) = (x^2 + 7x + 12) = (x + 3)(x + 4)$$

aus

$$\begin{array}{r} 1 \quad 3 \quad -16 \quad -48 \\ \quad \quad 4 \quad 28 \quad 48 \\ \hline 1 \quad 7 \quad 12 \quad 0 \end{array}$$

Insbesondere schliesst man, dass $x = -3$ und $x = -4$ die verbleibenden Nullstellen von $f(x) = (x - 3)(x - 4)(x + 4)(x + 3)$ sind. \circlearrowright

Berechnung der Ableitung

Für das Newton-Verfahren wird ausser dem Funktionswert auch die Ableitung benötigt. Der Funktionswert $r = f(x)$ wird mit dem Horner-Schema sofort gefunden, ebenso der Quotient $q(x)$. Es gilt also

$$f(X) = q(X)(X - x) + r,$$

was wir dazu verwenden können, die Ableitung von f mit Hilfe der Produktregel zu berechnen:

$$f'(X) = q'(X)(X - x) + q(X).$$

An der Stelle $X = x$ ist daher $f'(x) = q(x)$. Da die Koeffizienten von $q(X)$ bereits mit dem Horner-Schema berechnet worden sind, kann $f'(x)$ durch Iteration des Horner-Schemas berechnet werden.

Beispiel. Man berechne den Funktionswert und die Ableitung des Polynoms

$$f(x) = 2x^3 + x + 9$$

an der Stelle $x = 4$. Zweimalige Anwendung des Horner-Schemas ergibt

$$\begin{array}{r} 2 \quad 0 \quad 1 \quad 9 \\ \quad \quad 8 \quad 32 \quad 132 \\ \hline 2 \quad 8 \quad 33 \quad 141 \\ \quad \quad 8 \quad 64 \\ \hline 2 \quad 16 \quad 97 \end{array}$$

Man liest $f(4) = 141$ und $f'(4) = 97$ ab. \circlearrowright

Ist x eine doppelte Nullstelle des Polynoms $f(x)$, dann ist $f'(x) = 0$. Das Horner-Schema kann daher auch dazu verwendet werden, doppelte Nullstellen zu erkennen und damit die Faktorisierung zu vereinfachen, wie das folgende Beispiel zeigt.

Beispiel. Wir betrachten das Polynom

$$f(x) = x^4 - 13x^3 + 41x^2 - 47x + 18.$$

Es hat die Nullstelle $x = 1$, das Horner-Schema liefert für den Quotienten

$$\begin{array}{r} 1 \quad -13 \quad 41 \quad -47 \quad 18 \\ \quad \quad 1 \quad -12 \quad 29 \quad -18 \\ \hline 1 \quad -12 \quad 29 \quad -18 \quad 0 \\ \quad \quad 1 \quad -11 \quad 18 \\ \hline 1 \quad -11 \quad 18 \quad 0 \\ \quad \quad 1 \quad -10 \\ \hline 1 \quad -10 \quad 8 \end{array}$$

n	x_n	$f(x_n)$	$f'(x_n)$	$q_n(X)$
0	-10.000000	-182.00000	129.00000	$X^2 - X + 19$
1	-8.589148	-38.99232	75.71571	$X^2 + 0.4108524323X + 5.47112751$
2	-8.074164	-4.31028	59.24143	$X^2 + 0.9258356094X + 1.524651051$
3	-8.001407	-0.08021	57.04221	$X^2 + 0.9985933304X + 1.009848595$
4	-8.000000	-0.00005	57.00003	$X^2 + 0.9999990463X + 1.000006676$
5	-8.000000	0.00000	57.00000	$X^2 + X + 1$

Tabelle 2.5: Newton-Verfahren für das Polynom $f(X) = X^3 + 9X^2 + 9X + 8$ mit Hilfe des Horner-Schemas. Als Nebeneffekt bestimmt das Horner-Schema in jeder Iteration auch den Quotienten $q_n(x) = f(x)/(x - x_n)$.

Daraus kann man ablesen, dass $x = 1$ eine doppelte aber nicht eine dreifache Nullstelle ist und dass sich das Polynom schreiben lässt als

$$f(x) = (x - 1)^2 \cdot (x - 11x + 18) = (x - 1)^2(x - 2)(x - 9).$$

Damit ist das Polynom $f(x)$ vollständig faktorisiert. \circ

Newton-Verfahren für Nullstellen von Polynomen

Da mit dem Horner-Schema sowohl Funktionswerte wie auch Ableitungen effizient berechnet werden können, kann es dazu verwendet werden, das Newton-Verfahren für Polynomnullstellen zu implementieren.

Das Horner-Schemas liefert zu jeder Nullstelle x auch immer gleich den Quotienten $q(X) = f(X)/(X - x)$, welches für die Suche nach weiteren Nullstellen verwendet werden kann (Deflation).

Beispiel. Die reellen Nullstellen von $f(X) = X^3 + 9X^2 + 9X + 8$ sollen mit Hilfe des Newton-Verfahrens gefunden werden. Die Tabelle 2.5 zeigt die vom Horner-Schema berechneten Funktions- und Ableitungswerte sowie das Quotientenpolynom. Die Konvergenz ist quadratisch und liefert die Nullstelle $x = -8$ sowie den Quotienten $q(X) = X^2 + X + 1$, tatsächlich ist

$$(X + 8)q(X) = (X + 8)(X^2 + X + 1) = X^3 + 9X^2 + 9X + 8 = f(X).$$

Die Diskriminante von $q(X)$ ist $b^2 - 4ac = 1^2 - 4 \cdot 1 \cdot 1 = -3 < 0$, $q(X)$ hat also keine weiteren reellen Nullstellen. \circ

2.2.8 Inverse der Normalverteilungsfunktion

Das Integral der Standardnormalverteilungsdichte

$$\Phi(x) = \int_{-\infty}^x e^{-t^2/2} dt$$

kann nicht in geschlossener Form berechnet werden und erst recht nicht invertiert werden. Für die Anwendung wird jedoch die Umkehrfunktion benötigt, zu einem Wert $p \in [0, 1]$ ist dasjenige x zu finden, für welches $F(x) = p$ gilt. Im Beispiel auf Seite 21 wurde gezeigt, wie die Fehlerfunktion

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

dazu verwendet werden kann, die Normalverteilungsfunktion

$$\Phi(x) = \frac{1}{2} \left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) \right)$$

zu berechnen. In diesem Abschnitt soll untersucht werden, wie zu gegebenen Funktionswert p das zugehörige x bestimmt werden kann. Es soll also die Gleichung

$$\Phi(x) = p \quad \Rightarrow \quad f(x) = \frac{1}{2} + \frac{1}{2} \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) - p = 0$$

gelöst werden.

Sekantenverfahren

Die mit dem Sekantenverfahren gewonnenen Iterationsfolge ist in der rechten Spalte in Tabelle 2.6 dargestellt. Da die erste Ableitung von f relativ langsam ändert, ist die Konvergenz zwar zunächst offenbar nur linear, beschleunigt sich dann aber auf fast quadratische Konvergenz, weil die Sekante die Tangente sehr gut approximiert, sich das Sekantenverfahren in ihrem Konvergenzverhalten also dem Newton-Verfahren anzunähern beginnt. Wegen der unvermeidbaren Auslöschung bei der Berechnung der Sekantensteigung wird das Verfahren dann aber instabil, die Iteration bricht ab. Es können für den Typ `long double` nur etwa 18 signifikante Stellen gefunden werden, während das Newton-Verfahren noch drei weitere Stellen ermitteln kann.

Newton-Verfahren

Das Newton-Verfahren benötigt außer dem Funktionswert auch noch die Ableitung

$$f'(x) = \frac{d}{dx} \frac{1}{\sqrt{\pi}} \int_0^{x/\sqrt{2}} e^{-t^2} dt = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

Damit wird die Iterationsformel für das Newton-Verfahren:

$$x_{n+1} = x_n - \sqrt{2\pi} e^{x_n^2/2} \left(\frac{1}{2} + \operatorname{erf}\left(\frac{x_n}{\sqrt{2}}\right) - p \right). \quad (2.13)$$

Wie erwartet konvergiert die Iterationsfolge quadratisch für geeignete Startwerte (siehe Tabelle 2.6). Der Startwert $x_0 = 0$ funktioniert für jedes beliebige p . Bei weiter von 0 entfernten Starwerten läuft man Gefahr, dass die Iteration zu betragsmäßig grossen Werten x springt, was dann zu einem Überlauf führt.

2.3 Homotopie-Verfahren

Der Erfolg des Newton-Verfahrens hängt entscheidend von der Qualität der Anfangsschätzung x_0 ab. Allerdings ist es oft nicht einfach, eine solche Schätzung zu produzieren. Die folgende Idee kann dabei helfen.

Oft ist ein schwieriges Problem ein “deformierte” Variante eines weniger schwierigen Problems. Der Begriff der *Homotopie* gibt dieser Idee eine klare Bedeutung.

k	x_n nach Newton-Verfahren	x_n nach Sekantenverfahren
0	0.00000000000000000000000000000000	0.00000000000000000000000000000000
1	1.12798272358394999736	1.00000000000000000000000000000000
2	1.50523868934241237280	1.31831529614238960517
3	1.63077255164383121513	1.53246084663801306026
4	1.64469272791792957300	1.62023672225157423321
5	1.64485360566334308020	1.64272141855681471658
6	1.64485362695147202343	1.64481100638619422225
7	1.64485362695147239662	1.64485355229014184382
8	1.64485362695147239662	1.64485362694885539842
9		1.64485362695147239597

Tabelle 2.6: Newton-Iteration und Sekantenverfahren zur Bestimmung der Inversen der Verteilungsfunktion $\Phi(x)$ der Normalverteilung, berechnet mit dem Typ `long double`. Das Sekantenverfahren ist auch sehr schnell, da die Sekante bereits sehr gut mit der vom Newton-Verfahren verwendeten Tangente übereinstimmt. Wegen Auslöschung kann es allerdings nicht die volle Genauigkeit erreichen.

Definition 2.7. Zwei Funktionen $f_0(x)$ und $f_1(x)$ heißen homotop, wenn es eine stetige Funktion

$$F: \mathbb{R} \times I : (x, t) \mapsto F(x, t)$$

mit $I = [0, 1]$ gibt derart, dass $f_0(x) = F(x, 0)$ und $f_1(x) = F(x, 1)$. Die partielle Funktion $x \mapsto F(x, t)$ für $t \in I$ wird auch mit f_t bezeichnet: $f_t(x) = F(x, t)$.

Die Funktionen $f_t(x) = F(x, t)$ sind Funktionen “zwischen” $f_0(x)$ und $f_1(x)$. Lässt man den Parameter t von 0 nach 1 laufen, wird der Graph von $f_0(x)$ deformiert in den Graphen von $f_1(x)$.

Beispiel. Die Kepler-Gleichung ist

$$M = E - e \sin E,$$

wobei M gegeben und E gesucht ist. Dazu gehört die Funktion

$$f(E) = M - E + e \sin E.$$

Der Fall $e = 0$ ist ein trivial einfaches Problem, $E = M$ ist Nullstelle der Funktion

$$f_0(E) = M - E.$$

Eine Homotopie zwischen f_0 und $f_1 = f$ ist

$$F(x, t) = M - E + et \sin E.$$

○

Eine Homotopie kann dazu verwendet werden, Startwerte für das Newton-Verfahren zu liefern. Ist $x_0(t)$ eine Nullstelle der partiellen Funktion $x \mapsto F(x, t)$, dann kann $x_0(t)$ als Startwert zur Bestimmung einer Nullstelle von der partiellen Funktion $F(x, t')$ für $|t - t'| < \varepsilon$ dienen. Ist F differenzierbar bezüglich x , dann können einige Iterationen des Newton-Verfahrens aus dem Startwert $x_0(t)$ eine gute Lösung für $x_0(t')$ sein. Damit lässt sich der folgende Algorithmus konstruieren:

1. Starte mit der exakten Lösung $x_0 = x_0(0)$ und $t = 0$

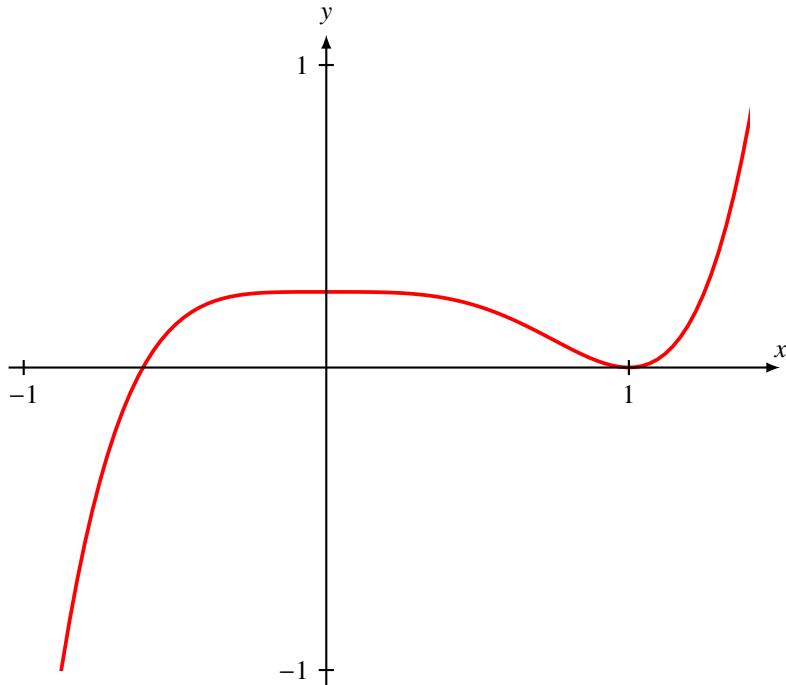


Abbildung 2.4: Graph zur Aufgabe 2.1, Nullstelle des Polynoms (2.14).

2. Inkrementiere t um Δt
3. verbessere x_0 durch einige Iterationen des Newton-Verfahrens zu einer Nullstelle von $f_t(x)$.
4. Wiederhole Schritte 2 und 3 bis $t = 1$.

Auf diese Weise kann sichergestellt werden, dass jede Iteration des Newton-Verfahrens mit einem guten Schätzwert startet, wenn auch nur für eine immer bessere Approximation des eigentlichen Problems.

Übungsaufgaben

2.1. Finden Sie eine Lösung der Gleichung

$$x^5 - \frac{5}{4}x^4 + \frac{1}{4} = 0 \quad (2.14)$$

mit Hilfe des Newton-Verfahrens. Welche Konvergenzgeschwindigkeit stellen Sie fest?

Lösung. Die Iterationsformel für das Newton-Verfahren braucht die Ableitung von $f(x) = x^5 - \frac{5}{4}x^4 + \frac{1}{4}$, also

$$f'(x) = 5x^4 - 5x^3 = 5(x - 1)x^3,$$

insbesondere hat $f'(x)$ eine Nullstelle bei $x = 1$, was in der Newton-Iteration zu Schwierigkeiten führen könnte. Ebenso ist 0 eine Nullstelle der Ableitung, so dass 0 und 1 ganz bestimmt keine guten Startwerte für die Newton-Iteration sind.

Die Iteration lautet jetzt

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^5 - \frac{5}{4}x_n^4 + \frac{1}{4}}{5(x_n - 1)x_n^3}.$$

Um eine Lösung zu finden, braucht man jetzt noch einen guten Startwert. Der Graph in Abbildung 2.4 zeigt, dass 1 eine doppelte Nullstelle ist, und dass es noch eine weitere Nullstelle in der Nähe von $x_0 = -0.5$ gibt.

Die Iteration liefert die folgenden Werte:

n	x_n
0	-0.5000000000000000
1	-0.6500000000000000
2	-0.610646335912608
3	-0.605893367985182
4	-0.605829597525286
5	-0.605829586188268
6	-0.605829586188268

Man hat also in 5 Iterationsschritten ein Resultat mit 15 Stellen Genaugigkeit erhalten, quadratische Konvergenz ist klar sichtbar.

Die Nullstelle bei $x = 1$ macht dem Newton-Algorithmus dagegen etwas Mühe. Die Iteration mit Startwert $x_0 = 0.5$ liefert

n	x_n
0	0.500000000000000
1	1.150000000000000
2	1.08416125585600
3	1.04522243974522
4	1.02356853334768
5	1.01205249984377
6	1.00609759001936
7	1.00306721671745
8	1.00153829071860
9	1.00077032580427
10	1.00038545926066
11	1.00019280387677
12	1.00009642051980
13	1.00004821490799
14	1.00002410861546
15	1.00001205459851
16	1.00000602736919
17	1.00000301369634
18	1.00000150685215
19	1.00000075342128
20	1.00000037677627

Die Konvergenz ist wie erwartet nur linear.

Die Ableitung $f'(x) = 4x^3(x - 1)$ der Funktion $f(x)$ hat an der Stelle $x = 0$ eine dreifache Nullstelle, was auch daran erkennbar ist, dass $f(x)$ abgesehen von der Konstante von vierter Ordnung in x ist. Der Graph von $f(x)$ ist daher in einer Umgebung von 0 sehr flach. Startwerte x_0 in der Umgebung von 0 führen daher automatisch zu einem Wert x_1 sehr weit weg vom Nullpunkt. Zum Beispiel führt $x_0 = -10^{-4}$ auf $x_1 = -49995000499.9501$. Es braucht dann 114 Iterationen, bis x_n nahe genug bei der negativen Nullstelle liegt, dass man wieder quadratische Konvergenz beobachten kann. \circ

3

Interpolation

Der Satz von Stone-Weierstrass garantiert, dass jede stetige Funktion auf einem Intervall beliebig genau durch Polynome approximiert werden kann. Polynome sind effizient berechenbar, es ist daher naheliegend, komplizierte Funktionen durch Polynome zu approximieren, die möglichst viele für die vorliegende Anwendung relevante Eigenschaften mit der Funktion gemeinsam habe.

Leider sagt der Satz von Stone-Weierstrass nichts darüber, wie solche Polynome gefunden werden könnten. Ziel dieses Kapitels ist daher, einige Möglichkeiten zusammenzustellen, solche Approximationenpolynome zu finden und insbesondere auch ihre Fehler abzuschätzen.

3.1 Lineare Interpolation und Polygonzüge

Oft sind von einer Funktion nur einzelne Werte bekannt, doch meist reicht dies, den ungefähren Verlauf ihres Graphen zu erahnen. Unter der Annahme, dass sich die Funktion zwischen den bekannten Werten nicht zu “wild” verhält, können Werte zwischen den bekannten Werten abgeschätzt werden. In diesem Abschnitt soll daher die folgende Aufgabe gelöst werden.

Aufgabe 3.1. *Gegeben $n + 1$ sogenannte Stützstellen*

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

im Intervall $[a, b]$ und bekannte Funktionswerte f_i , $0 \leq i \leq n$, finde eine stetige Funktion $f: [a, b] \rightarrow \mathbb{R}$ derart, dass $f(x_j) = f_j$ für $0 \leq j \leq n$.

3.1.1 Lineare Interpolation

Die Aufgabe (3.1) ist zu wenig präzise gestellt, es gibt unendlich viele Lösungen. Es müssen daher zusätzliche Bedingungen an die gesuchte Funktion f gestellt werden, damit die Lösung eindeutig bestimmt wird. Ein mögliche solche Bedingung ist, dass die Funktion f in jedem Teilintervall zwischen aufeinanderfolgenden Stützstellen linear ist. In diesem Fall haben die Werte außerhalb des Intervalls offenbar keinen Einfluss auf den Verlauf im Inneren des Intervalls, es genügt also das Problem mit nur zwei Stützstellen zu lösen.

Satz 3.2. Die einzige auf dem Intervall $[x_i, x_{i+1}]$ definierte lineare Funktion $f(x)$ mit Funktionswerten $f(x_i) = f_i$ und $f(x_{i+1}) = f_{i+1}$ hat im Punkt $x \in [x_i, x_{i+1}]$ den Wert

$$f(x) = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}(x - x_i) + f_i = \frac{x - x_{i+1}}{x_i - x_{i+1}}f_i + \frac{x_i - x}{x_i - x_{i+1}}f_{i+1} = f_il_i(x) + f_{i+1}l_{i+1}(x) \quad (3.1)$$

mit den linearen Funktionen

$$l_i(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} \quad \text{und} \quad l_{i+1}(x) = \frac{x_i - x}{x_i - x_{i+1}}. \quad (3.2)$$

Beweis von Satz 3.2. Die beiden linearen Funktion $l_i(x)$ und $l_{i+1}(x)$ haben an den Intervallenden die speziellen Werte

$$\begin{array}{ll} l_i(x_i) = 1 & l_{i+1}(x_i) = 0 \\ l_i(x_i) = 0 & l_{i+1}(x_i) = 1, \end{array}$$

wie man durch Einsetzen unmittelbar bestätigen kann. Die zweite Form in (3.1) ist daher als Linear-kombination

$$f(x) = f_il_i(x) + f_{i+1}l_{i+1}(x)$$

linearer Funktionen wieder linear und dank der speziellen Werte von l_i und l_{i+1} folgt unmittelbar, dass

$$\begin{aligned} f(x_i) &= f_il_i(x_i) + f_{i+1}l_{i+1}(x_i) = f_i \cdot 1 + f_{i+1} \cdot 0 = f_i \\ f(x_{i+1}) &= f_il_i(x_{i+1}) + f_{i+1}l_{i+1}(x_{i+1}) = f_i \cdot 0 + f_{i+1} \cdot 1 = f_{i+1} \end{aligned}$$

gilt.

Der Bruch

$$m = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$$

ist die Steigung der Geraden durch die Punkte (x_i, f_i) und (x_{i+1}, f_{i+1}) . Der erste Ausdruck auf der rechten Seite in (3.1) ist also $f(x) = m(x - x_i) + f_i$, dies ist die Gleichung einer Geraden mit Steigung m durch den Punkt (x_i, f_i) , sie verläuft natürlich auch durch den Punkt (x_{i+1}, f_{i+1}) . \square

3.1.2 Polygonzüge

(3.2) Wendet man das Resultat von Satz 3.2 auf jedes Teilintervall an, entsteht eine Interpolationsfunktion, deren Graph ein Polygonzug ist. Eine solche Funktion lässt sich einfacher beschreiben mit Hilfe der Interpolationsfunktionen mit den speziellen Werten

$$l_i(x_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{sonst.} \end{cases} \quad (3.3)$$

Die Funktionen sind in Abbildung 3.1 für die Stützstellen $x_0 = 0, x_1 = 1, \dots, x_6 = 6$ dargestellt. Die lineare Interpolationsfunktion kann jetzt als Linearkombination der Funktionen l_i geschrieben werden:

$$f(x) = \sum_{j=0}^n f_j l_j(x). \quad (3.4)$$

Diese Lösung des Interpolationsproblems kann für alle weiteren Interpolationsansätze in diesem Kapitel als Vorlage dienen. Es geht nämlich nicht darum, eine Interpolationsfunktion in geschlossener

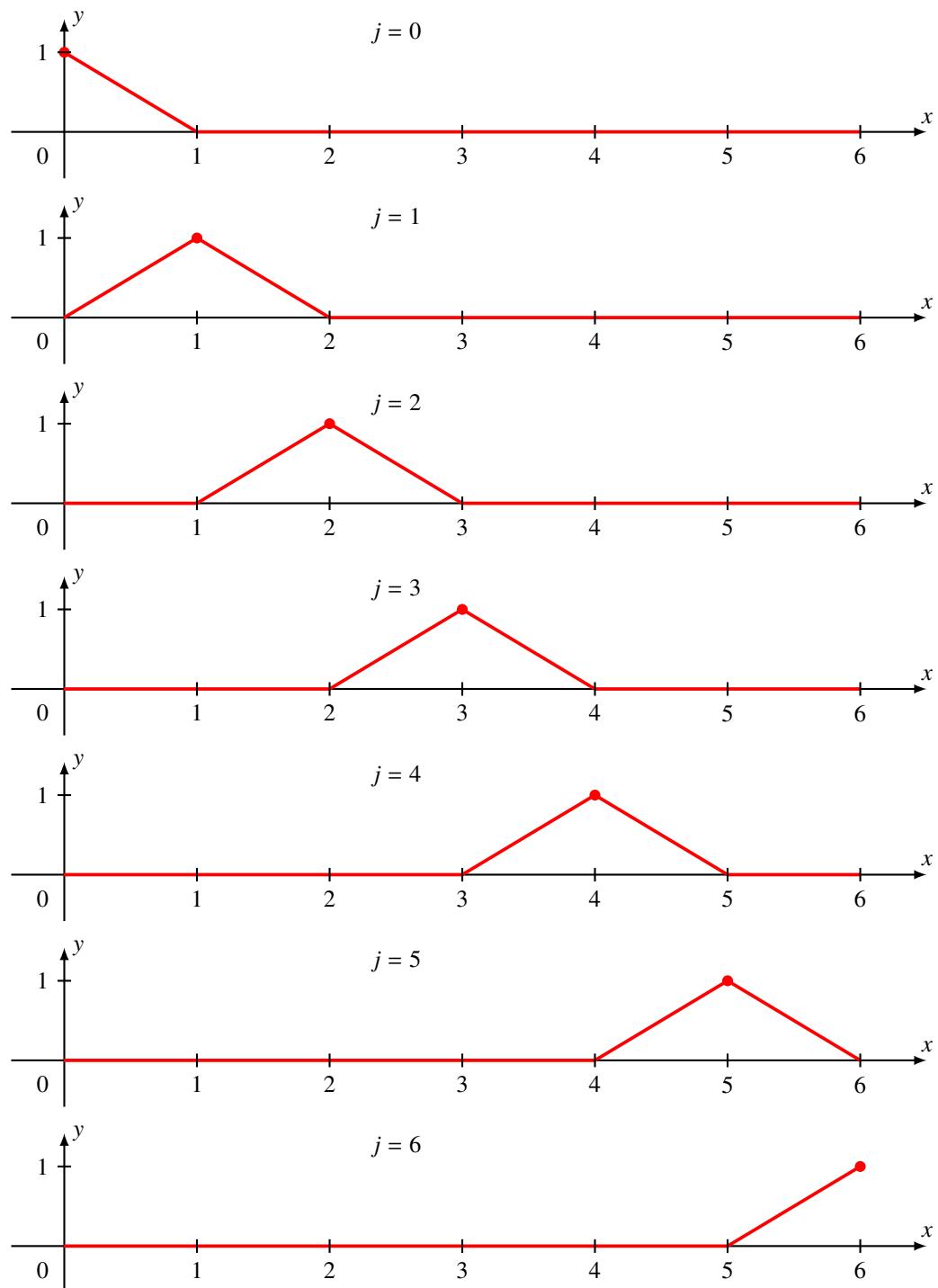


Abbildung 3.1: Basisfunktionen für die lineare Interpolation einer Funktion $f: [0, 6] \rightarrow \mathbb{R}$ mit Stützstellen $0, 1, \dots, 6$

Form hinzuschreiben, dies ist für die Funktionen $l_j(x)$ ohnehin nicht möglich. Es ist nur nötig, dass Funktionswerte $f(x)$ effizient berechnet werden können. Die letztere Aufgabe ist gelöst, wenn man die Funktionen $l_j(x)$ effizient berechnen kann. Es ist dann nur noch die Linearkombination (3.4) zu bilden.

Die Wahl der Funktionen $l_j(x)$, die natürlich die Bedingungen (3.3) erfüllen müssen, bestimmt die Eigenschaften der Interpolationsfunktion, die nach (3.4) gebildet wird. In diesem Abschnitt wählen die l_j stückweise lineare Funktionen, also auch die Funktion $f(x)$. Im nächsten Abschnitt sollen die Funktionen Polynome sein, also wird die Interpolationsfunktion ebenfalls ein Polynom sein.

3.2 Interpolationspolynom

In diesem Abschnitt wird die folgende Aufgabe gelöst.

Aufgabe 3.3 (Interpolationspolynom). *Gegeben Stützstellen*

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

und Funktionswerte $f_i, 0 \leq i \leq n$, finde ein Polynom $l(x)$ mit der Eigenschaft $l(x_i) = f_i$ für alle $i = 0, 1, \dots, n$.

Gegeben sind also $n + 1$ Bedingungen, die das Polynom erfüllen muss. Abgesehen von trivialen Fällen wie dem Null-Polynom, muss ein Polynom im Allgemeinen mindestens den Grad n haben, damit alle Bedingungen durch geeignete Wahl der $n + 1$ Koeffizienten erfüllt werden können. Man könnte das Polynom nämlich in der Form

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

ansetzen und die Stützstellen einsetzen. Lösung des Gleichungssystems

$$\begin{aligned} a_N x_0^N + a_{N-1} x_0^{N-1} + \dots + a_1 x_0 + a_0 x_0 &= f_0 \\ a_N x_1^N + a_{N-1} x_1^{N-1} + \dots + a_1 x_1 + a_0 x_1 &= f_1 \\ \vdots &\quad \vdots \quad \ddots \quad \vdots \quad \vdots \quad \vdots \\ a_N x_n^N + a_{N-1} x_n^{N-1} + \dots + a_1 x_n + a_0 x_n &= f_n \end{aligned} \tag{3.5}$$

für die Koeffizienten a_k liefert dann die gesuchten Koeffizienten. Dieser Weg ist allerdings sehr aufwendig, die Lösung eines linearen Gleichungssystems mit dem Gauss-Algorithmus benötigt $O(n^3)$ Operationen. Die sehr spezielle Struktur des Gleichungssystems sollte ermöglichen, das Polynom $l(x)$ auf direkterem Weg zu ermitteln.

3.2.1 Bestimmung des Interpolationspolynoms

Das allgemeine Interpolationsproblem kann leicht gelöst werden, wenn das folgende spezielle Interpolationsproblem gelöst ist.

Aufgabe 3.4 (Spezielle Interpolationspolynome). *Gegeben die Stützstellen*

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b,$$

finde Polynome l_j vom Grad n derart, dass

$$l_j(x_i) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{sonst.} \end{cases}$$

Jedes der Interpolationspolynome l_j hat Grad n , also hat auch eine beliebige Linearkombination den Grad höchstens n . Die Linearkombination

$$p(x) = \sum_{j=0}^n f_j l_j(x)$$

ist das gesuchte Interpolationspolynom, wie Einsetzen von x_i in

$$p(x_i) = \sum_{j=0}^n f_j l_j(x_i) = \sum_{j=0}^n f_j \delta_{ij} = f_i$$

bestätigt.

Beispiel. Ein besonders einfacher Fall ist $n = 1$. Gesucht ist eine lineare Funktion $l(x) = a_1 x + a_0$ derart, dass $l(x_0) = f_0$ und $l(x_1) = f_1$. Polynome l_0 und l_1 können leicht angegeben werden:

$$l_0(x) = \frac{x_1 - x}{x_1 - x_0} \quad \text{und} \quad l_1(x) = \frac{x - x_0}{x_1 - x_0}$$

haben die geforderten Eigenschaften. Die gesuchte Interpolationsfunktion ist daher

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

Der Koeffizient von x ist wie erwartet die Steigung der Geraden durch die Punkte (x_0, f_0) und (x_1, f_1) .



Ein Polynom vom Grad $n + 1$, welches in *allen* Stützstellen verschwindet, ist leicht zu finden, es ist

$$l(x) = (x - x_0)(x - x_1)(x - x_2) \cdots (x - x_{n-1})(x - x_n).$$

Ein Polynom, welches nur an der Stützstelle x_j *nicht* verschwindet, entsteht, indem man den Faktor $(x - x_j)$ weglässt, es hat den Grad n . Wir führen dafür die Notation

$$(x - x_0)(x - x_1)(x - x_2) \cdots (\widehat{x - x_j}) \cdots (x - x_{n-1})(x - x_n),$$

der Hut bedeutet, dass dieser Faktor weggelassen werden soll. Allerdings hat dieses Polynom nicht den geforderten Wert 1, man muss es also noch mit einer geeigneten Konstante multiplizieren. Das gesuchte Polynom $l_j(x)$ hat daher die Form

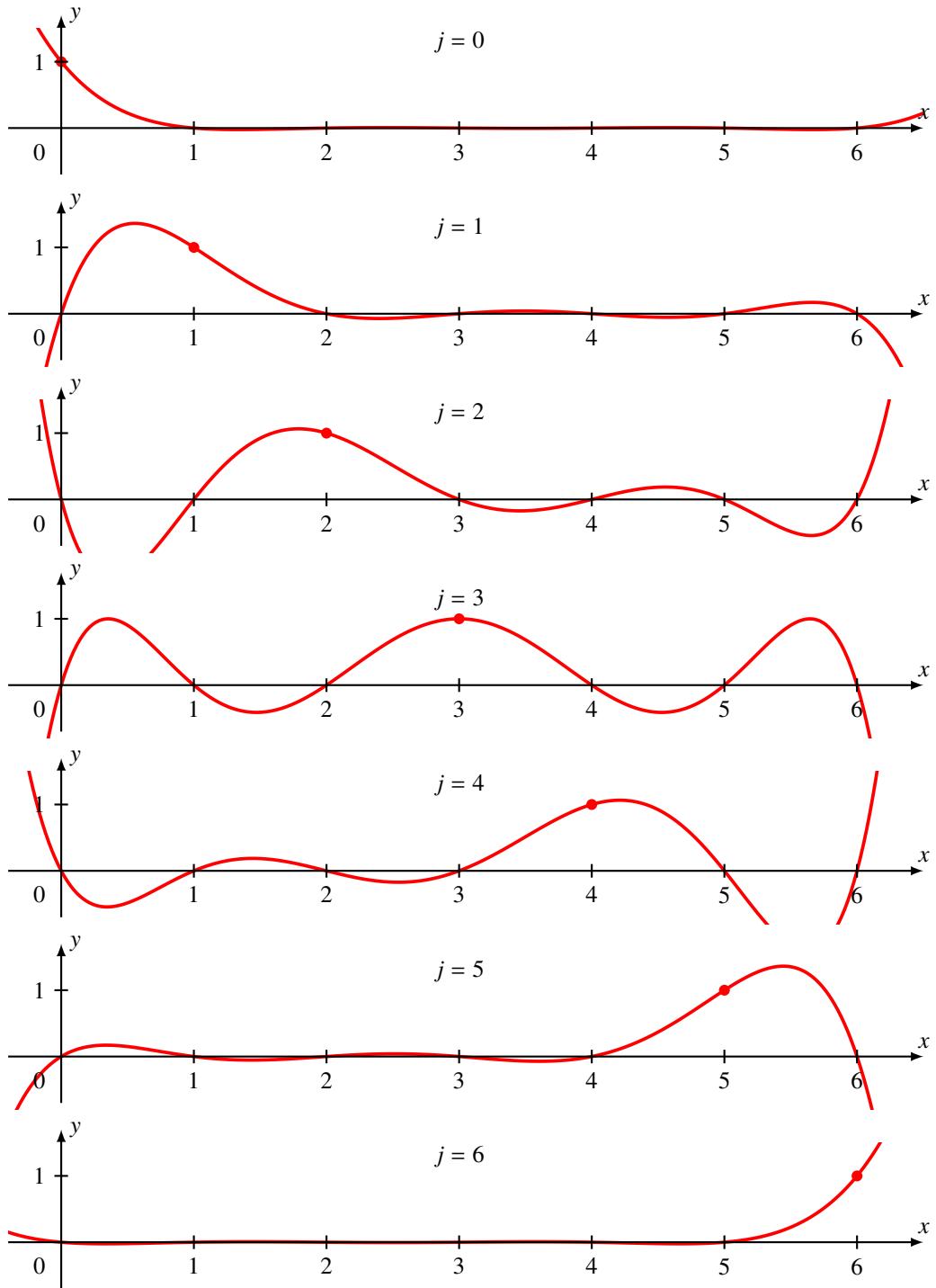
$$l_j(x) = c_j (x - x_0)(x - x_1)(x - x_2) \cdots (\widehat{x - x_j}) \cdots (x - x_{n-1})(x - x_n).$$

Einsetzen von x_j ergibt

$$l_j(x_j) = 1 = c_j (x_j - x_0)(x_j - x_1)(x_j - x_2) \cdots (\widehat{x_j - x_j}) \cdots (x_j - x_{n-1})(x_j - x_n),$$

die Konstante c_j ist daher

$$c_j = \prod_{\substack{i=0 \\ i \neq j}}^n \frac{1}{x_j - x_i}.$$

Abbildung 3.2: Polynome $l_j(x)$, welche des spezielle Interpolationsproblem 3.4 lösen.

Beispiel. Man finde ein Polynom, welches $l(0) = l(1) = 0$ und $l(\frac{1}{2}) = 1$ erfüllt. Wegen $f_0 = f_2 = 0$ ist nur das Polynom l_1 zu ermitteln, es ist

$$l(x) = l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{x(x - 1)}{\frac{1}{2}(\frac{1}{2} - 1)} = \frac{1}{4}x(1 - x).$$
○

3.2.2 Fehler von Approximationsspolynomen

Getreu der Maxime, dass wir zu jeder numerischen Lösungsformel auch Informationen über die zu erwartenden Fehler brauchen, entwickeln wir in diesem Abschnitt die Theorie des Fehlers der Approximationsspolynome. Wir müssen zu diesem Zweck einen kleinen Ausflug in die Analysis unternehmen in einen Bereich, der im Unterricht manchmal etwas zu kurz kommt.

Wenn die Ableitung einer Funktion in einem Intervall klein ist, dann werden auch die Funktionswerte im Inneren dieses Intervalls nicht gross von den Werten am Rand abweichen können. Eine grosse Abweichung würde ja automatisch eine Steigung einer Sekanten und damit auch eine grosse Steigung einer Tangenten zur Folge haben. Dies ist die Idee, die den nachfolgend entwickelten Fehlerabschätzungen zu Grunde liegt.

Der Zwischenwertsatz

Der Ausgangspunkt aller nachfolgenden Überlegungen ist die intuitiv anschauliche Tatsache, dass eine stetige Funktion keine Sprünge macht.

Satz 3.5 (Zwischenwertsatz). *Eine auf dem Intervall $[a, b]$ stetige Funktion nimmt jeden Wert im Intervall $[f(a), f(b)]$ an. Anders ausgedrückt, für jedes y zwischen $f(a)$ und $f(b)$ gibt es ein x zwischen a und b derart, dass $y = f(x)$.*

Dieser Satz war natürlich bereits die Grundlage des Verfahrens der Intervallhalbierung, mit welchem wir in Abschnitt 2.1.1 Gleichungen gelöst haben. Wenn die Funktion an den Intervallenden verschiedene Vorzeichen hat, dann muss es eine Nullstelle im Inneren des Intervalls geben. Die Intervallhalbierung hat in jedem Schritt ein neues Intervall konstruiert, das die Nullstelle enthält.

Der Satz von Rolle

Der Satz von Rolle erweitert den Zwischenwertsatz auf die Ableitung einer differenzierbaren Funktion an (Abbildung 3.3).

Satz 3.6 (Rolle). *Sei f eine auf dem Intervall $[a, b]$ nicht konstante, stetig differenzierbare Funktion mit $f(a) = f(b)$, dann gibt es einen Punkt $\xi \in (a, b)$ im Inneren des derart, dass $f'(\xi) = 0$.*

Der Satz von Rolle ist eine Selbstverständlichkeit, wenn die Ableitung $f'(x)$ stetig ist, doch dies wird nicht vorausgesetzt, es wird nur verlangt, dass die Ableitung existiert. Außerdem macht der Satz eine Aussage darüber, dass die Zwischenstelle ξ im Inneren des Intervalls sei.

Beweis. Eine stetige Funktion hat auf dem kompakten Intervall $[a, b]$ mindestens ein Maximum und ein Minimum. Da die Funktion nicht konstant ist, ist das Maximum oder das Minimum von $f(a)$ verschieden. Wir nehmen an $\xi \in [a, b]$ sei ein Maximum mit dieser Eigenschaft, das Argument für das Minimum ist völlig analog. Wegen $f(\xi) > f(a)$ ist ξ ein Punkt im Inneren des Intervalls, also kann ξ nicht $= a$ sein, folglich ist $\xi \in (a, b)$.

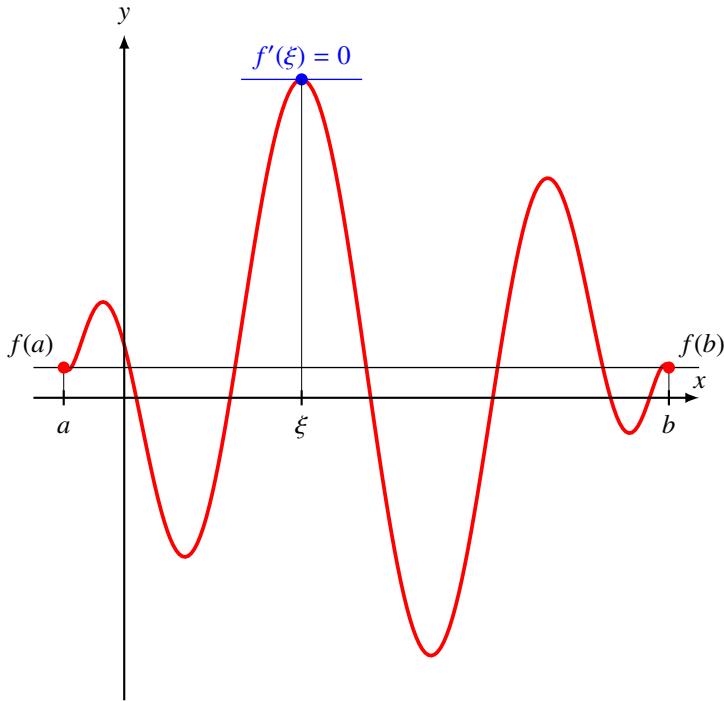


Abbildung 3.3: Satz von Rolle: eine nicht konstante differenzierbare Funktion, die an den Enden eines Intervalls den gleichen Funktionswert hat, hat im Inneren des Intervalls eine Stelle ξ mit Ableitung 0.

Wegen $f(\xi) \geq f(x) \forall x \in [a, b]$ folgt dann

$$\begin{aligned}f'(\xi) &= \lim_{h \rightarrow 0^+} \frac{f(\xi + h) - f(\xi)}{h} \leq 0 \\f'(\xi) &= \lim_{h \rightarrow 0^-} \frac{f(\xi + h) - f(\xi)}{h} \geq 0.\end{aligned}$$

Da f differenzierbar ist, müssen diese beiden Grenzwerte übereinstimmen, also ist $f'(\xi) = 0$. \square

Nullstellen und der Satz von Rolle

Ist $p(x)$ ein Interpolationspolynom für die Funktion $f(x)$, dann ist $f(x_i) - p(x_i) = 0$ für alle Stützstellen. Insbesondere macht der Satz von Rolle Aussagen über die Differenz $f(x) - p(x)$ zwischen den Stützstellen. Diese Frage wird im folgenden Satz genauer untersucht.

Satz 3.7. Ist f eine differenzierbare Funktion auf dem Intervall $[a, b]$ mit Nullstellen

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b,$$

die auf keinem Teilintervall $[x_i, x_{i+1}]$ konstant ist, dann hat f' im Inneren jedes Teilintervalls $[x_i, x_{i+1}]$ eine Nullstelle.

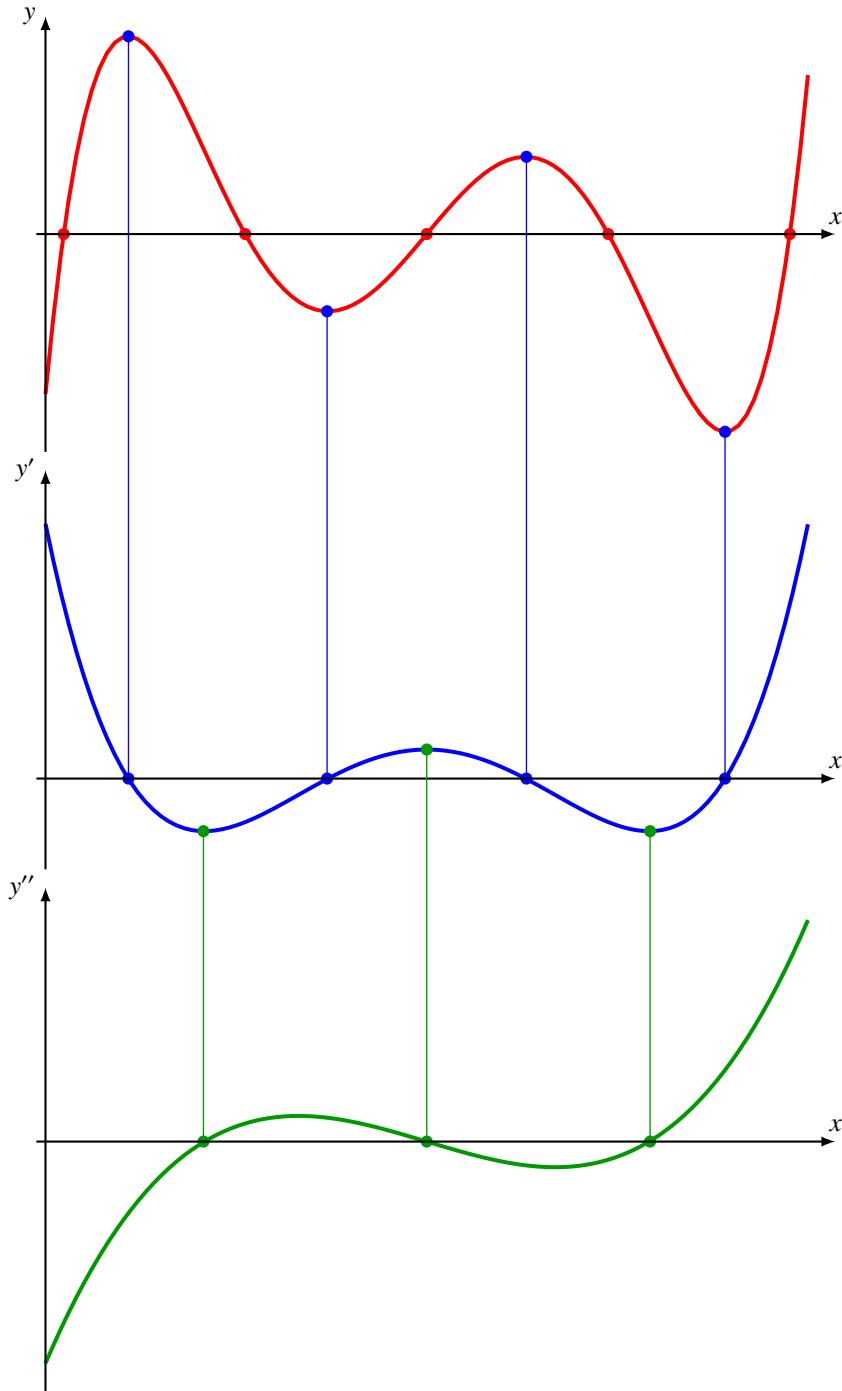


Abbildung 3.4: Schachtelung der Nullstellen von $f(x)$, $f'(x)$ und $f''(x)$. Der Satz von Rolle 3.6 impliziert, dass sich zwischen zwei Nullstellen von f immer eine Nullstelle von f' befindet und ebenso zwischen zwei Nullstellen von f' eine von f'' .

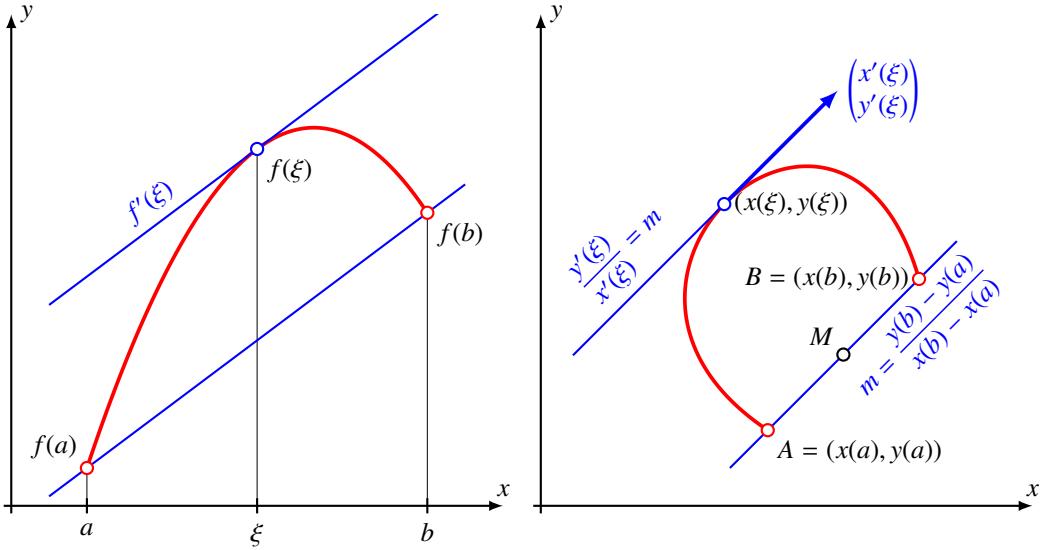


Abbildung 3.5: Der Mittelwertsatz 3.8 besagt, dass die Sekante des Graphen einer differenzierbaren Funktion immer eine Tangente mit der gleichen Steigung hat. Diese Eigenschaft kann man auch für eine beliebige ebene Kurven erwarten (rechts), dies ist der Inhalt des verallgemeinerten Mittelwertsatzes 3.10.

Das Polynom

$$l(x) = (x - x_0)(x - x_1) \dots (x - x_{n-1})(x - x_n),$$

welches für die Konstruktion des Interpolationspolynoms verwendet wurde, hat genau die Nullstellen $x_0, x_1, \dots, x_{n-1}, x_n$. Nach dem Satz 3.7 muss es zwischen je zwei aufeinanderfolgenden Nullstellen von l eine Nullstelle der Ableitung geben. Diese Situation ist in Abbildung 3.4 für den Fall $l(x) = (x + 2)(x + 1)x(x - 1)(x - 2)$ dargestellt.

Die höheren Ableitungen $f^{(k)}$ haben ihre Nullstellen natürlich auch wieder zwischen den Nullstellen der Ableitung $f^{(k-1)}$. Die n -te Ableitung ist konstant und hat keine Nullstellen.

Der Mittelwertsatz der Differentialrechnung

Der Satz von Rolle sagt etwas über die Nullstellen von der Ableitung einer differenzierbaren Funktion. Zwischen zwei Argumentwerten mit gleichem Funktionswert gibt es immer eine Nullstelle der Ableitung. Der Mittelwertsatz der Differentialrechnung verallgemeinert diese Aussage auf verschiedene Funktionswerte.

Satz 3.8 (Mittelwertsatz). *Zu einer auf dem Intervall $[a, b]$ differenzierbaren Funktion $f(x)$ gibt es immer ein $\xi \in [a, b]$ mit*

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}$$

(Abbildung 3.5 links).

Beweis. Man betrachtet die Funktion

$$g(x) = f(x) - \frac{f(b) - f(a)}{b - a}(x - a),$$

sie hat an den Intervallenden die Werte

$$\begin{aligned} g(a) &= f(a) - \frac{f(b) - f(a)}{b - a}(a - a) = f(a), \\ g(b) &= f(b) - \frac{f(b) - f(a)}{b - a}(b - a) = f(b) - (f(b) - f(a)) = f(a). \end{aligned}$$

Die Funktionswerte an den Intervallenden sind also gleich, nach dem Satz 3.6 von Rolle hat also die Ableitung g' eine Nullstelle $\xi \in [a, b]$:

$$g'(\xi) = f'(\xi) - \frac{f(b) - f(a)}{b - a} = 0 \quad \Rightarrow \quad f'(\xi) = \frac{f(b) - f(a)}{b - a}. \quad \square$$

Mittelwertsatz und ebene Kurven

Die von Abbildung 3.5 vermittelte Intuition lässt sich auch für beliebige parametrisierte Kurven $\gamma: [a, b] \rightarrow \mathbb{R} : t \mapsto (x(t), y(t))$ in der Ebene verallgemeinern (Abbildung 3.5 rechts). Sind die Punkte $A = (\gamma(a), \gamma(b))$ und $B = (y(a), y(b))$ verschieden, so dass die Richtung von A nach B wohldefiniert ist, dann erwarten wir einen Zwischenstelle ξ mit einer Tangente mit derselben Richtung.

Satz 3.9. Sei $\gamma: [a, b] \rightarrow \mathbb{R} : t \mapsto (x(t), y(t))$ eine stetig differenzierbare ebene Kurve mit $|\dot{\gamma}(t)| \neq 0$ für alle $t \in [a, b]$ und $A = \gamma(a) \neq \gamma(b) = B$. Dann gibt es ein $\xi \in [a, b]$ derart, dass $\dot{\gamma}(\xi)$ die gleiche Richtung hat wie \overrightarrow{AB} .

Beweis. Zwei Vektoren haben die gleiche Richtung, wenn die Determinante mit den Vektoren als Spaltenvektoren verschwindet. Der Mittelpunkt der Strecke AB ist $M = (x_M, y_M) = (\frac{1}{2}(x(a) + x(b)), \frac{1}{2}(y(a) + y(b)))$. Wir definieren die Funktion

$$h(t) = \det(\overrightarrow{M\gamma(t)}, \overrightarrow{AB}) = \begin{vmatrix} x(t) - x_M & x(b) - x(a) \\ y(t) - y_M & y(b) - y(a) \end{vmatrix}$$

Da die Strecken AB und MA bzw. MB die gleiche Richtung haben, gilt an den Stellen $t = a$ und $t = b$

$$h(a) = \det(\overrightarrow{MA}, \overrightarrow{AB}) = 0 \quad \text{and} \quad h(b) = \det(\overrightarrow{MB}, \overrightarrow{AB}) = 0.$$

Nach dem Satz von Rolle muss es daher ein $\xi \in [a, b]$ geben mit

$$0 = h'(\xi) = \det(\dot{\gamma}(\xi), \overrightarrow{AB}) = \begin{vmatrix} x'(\xi) & x(b) - x(a) \\ y'(\xi) & y(b) - y(a) \end{vmatrix}, \quad (3.6)$$

was natürlich wieder bedeutet, dass die Tangente $\dot{\gamma}(\xi)$ parallel ist zur Strecke AB . \square

Satz 3.10 (Verallgemeinerter Mittelwertsatz). Sind $x(t)$ und $y(t)$ stetig differenzierbare Funktionen auf dem Intervall $[a, b]$ derart, dass $x(a) \neq x(b)$ oder $y(a) \neq y(b)$ und ausserdem $y'(t) \neq 0$ für $t \in [a, b]$. Dann gibt es ein $\xi \in [a, b]$ mit

$$\frac{y'(\xi)}{x'(\xi)} = \frac{y(b) - y(a)}{x(b) - x(a)}.$$

Beweis. Die Bedingungen bedeuten, dass die die Kurve $\gamma(x(t), y(t))$ den Voraussetzungen des Satzes 3.9 genügt. Es gibt daher einen Punkten $\xi \in [a, b]$ wo die Tangente parallel zur Strecke AB ist. Nach 3.6 bedeutet dies

$$y'(\xi) \cdot (x(b) - x(a)) - x'(\xi) \cdot (y(b) - y(a)) = 0 \quad \Rightarrow \quad \frac{x'(\xi)}{y'(\xi)} = \frac{y(b) - y(a)}{x(b) - x(a)}. \quad \square$$

Mittelwertsatz in \mathbb{R}^n

Der Mittelwertsatz verspricht die Existenz eines Arguments $\xi \in [a, b]$, gibt aber keinerlei Hinweise darauf, wie ξ berechnet werden könnte. Ausserdem lässt sich der Satz in dieser Form nicht auf höherdimensionale Situationen verallgemeinern. In vielen Anwendungen des Mittelwertsatzes ist der genaue Wert von ξ gar nicht nötig. Meistens wird nur verwendet, dass die Ableitung eine Abschätzung dafür gibt, wie gross der Unterschied zwischen $f(b)$ und $f(a)$ sein kann. Dies kann zum Beispiel wie im folgenden Satz formuliert werden.

Satz 3.11. *Für eine in $[a, b]$ stetig differenzierbare Funktion $f: [a, b] \rightarrow \mathbb{R}^n$ gibt es ein $\xi \in [a, b]$ mit*

$$|f(b) - f(a)| \leq |f'(\xi)| \cdot |b - a|.$$

Beweis. [1, (8.5.1)] □

Diese Form ist ausreichend, um zum Beispiel Fehlerabschätzungen in der Numerik durchzuführen.

Taylor-Reihe mit Lagrange-Restterm

Die Ableitung $f'(a)$ einer Funktion $f(x)$ an der Stelle a ist definiert als die beste lineare Approximation

$$f(x) = f(a) + f'(a) \cdot (x - a) + o(|x - a|).$$

Der Mittelwertsatz besagt, dass

$$f(x) = f(a) + f'(\xi) \cdot (x - a)$$

für ein geeignetes $\xi \in [a, x]$. Man kann also sagen, dass $f(a)$ eine Approximation für $f(x)$ mit einem Fehler der Form $f'(\xi) \cdot (x - a)$ für ein geeignetes $\xi \in [a, x]$ ist. Das Taylor-Polynom verallgemeinert diese Beobachtung auf eine Approximation höherer Ordnung.

Satz 3.12. *Sei $f(x)$ eine $n + 1$ -mal stetig differenzierbare Funktion in einer Umgebung von a . Dann ist das Taylor-Polynom*

$$\begin{aligned} T_{a,n}f(x) &= f(a) + f'(a) \cdot (x - a) + f''(x) \frac{(x - a)^2}{2!} + f'''(x) \frac{(x - a)^3}{3!} + \cdots + f^{(n)}(a) \frac{(x - a)^n}{n!} \\ &= \sum_{k=0}^n f^{(k)}(a) \frac{(x - a)^k}{k!} \end{aligned}$$

eine Approximation von $f(x)$ derart, dass

$$f(a) = T_{a,n}f(a), \quad f'(a) = T_{a,n}f'(a), \quad f''(a) = T_{a,n}f''(a), \quad \dots, \quad f^{(n)}(a) = T_{a,n}f^{(n)}(a).$$

Ausserdem gibt es einen Wert $\xi \in [a, x]$ derart, dass der Fehler

$$R_{n,a}f(x) = f(x) - T_{n,a}f(x) = f^{(n+1)}(\xi) \frac{(x - a)^{n+1}}{(n + 1)!}$$

ist.

$R_{n,a}f(x)$ heisst das *Lagrange-Restglied* des Taylor-Polynoms.

Beweis. Zunächst kann die Behauptung über die Ableitungen von $T_{n,a}f(x)$ durch die Rechnung

$$\begin{aligned} T_{n,a}f'(x) &= f'(a) + f''(a) \cdot (x - a) + f'''(a) \frac{(x - a)^2}{2!} + \dots &\Rightarrow T_{n,a}f'(a) &= f'(a) \\ T_{n,a}f''(x) &= f''(a) + f'''(a) \cdot (x - a) + \dots &\Rightarrow T_{n,a}f''(a) &= f''(a) \\ &\vdots &&\vdots \\ T_{n,a}f^{(n)}(x) &= f^{(n)}(a) &\Rightarrow T_{n,a}f^{(n)}(a) &= f^{(n)}(a) \end{aligned}$$

direkt bestätigt werden. Daraus folgt, dass die Funktion $g(x) = f(x) - T_{n,a}f(x)$ die Eigenschaft

$$g(a) = g'(a) = g''(a) = \dots = g^{(n)}(a) = 0 \quad (3.7)$$

hat. Es muss als nur gezeigt werden, dass es unter den Voraussetzungen (3.7) eine Zahl ξ gibt derart, dass

$$g(x) = g^{(n+1)}(\xi) \frac{(x - a)^{n+1}}{(n + 1)!}$$

gilt. Der Fall $n = 0$ ist der Mittelwertsatz 3.8.

Für $n > 0$ kann die Behauptung mit vollständiger Induktion bewiesen werden. Sei also bereits bewiesen, dass für eine Funktion, deren Ableitungen bis zur Ordnung $n-1$ an der Stelle a verschwinden, eine Zahl $\xi \in [a, x]$ existiert mit

$$g(x) = g^{(n)}(\xi) \frac{(x - a)^n}{n!}$$

und müssen jetzt zeigen, dass dies auch für $n + 1$ gilt.

Wir müssen $g(x)$ mit $(x - a)^{n+1}$ vergleichen und untersuchen dazu den Quotienten

$$\frac{g(x)}{(x - a)^{n+1}}.$$

Zur Abkürzung schreiben wir $h(x) = (x - a)^{n+1}$, es ist also der Quotient

$$\frac{g(x)}{h(x)} = \frac{g(x) - g(a)}{h(x) - h(a)}$$

zu berechnen. Nach dem verallgemeinerten Mittelwertsatz 3.10 gibt es $\xi_1 \in [a, x]$ mit

$$\frac{g'(\xi_1)}{h'(\xi_1)} = \frac{g(x) - g(a)}{h(x) - h(a)}.$$

Da die Ableitung $h'(x) = (n + 1)(x - a)^n$ ist, folgt

$$\frac{g'(\xi_1)}{h'(\xi_1)} = \frac{g'(\xi_1)}{(n + 1)(\xi_1 - a)^n} = \frac{g(x) - g(a)}{h(x) - h(a)} = \frac{g(x)}{(x - a)^{n+1}}. \quad (3.8)$$

Die Ableitungen der Ordnung $\leq n - 1$ der Funktion $g'(x)$ verschwinden im Punkt a , nach Induktionsvoraussetzung gibt es also ein ξ derart, dass

$$g'(\xi_1) = (g')^{(n)}(\xi) \frac{(\xi_1 - a)^n}{n!} = g^{(n+1)}(\xi) \frac{(\xi_1 - a)^n}{n!}$$

gilt. Setzt man dies in (3.8) ein, erhält man

$$\frac{g(x)}{(x-a)^{n+1}} = \frac{g'(\xi_1)}{(n+1)(\xi_1-a)^n} = g^{(n+1)}(\xi) \frac{(\xi_1-a)^n}{n!} \frac{1}{(n+1)(\xi_1-a)^n} = g^{(n+1)}(\xi) \frac{1}{(n+1)!}.$$

Durch Multiplikation mit $(x-a)^{n+1}$ erhalten wir

$$g(x) = g^{(n+1)}(\xi) \frac{(x-a)^{n+1}}{(n+1)!}.$$

Damit ist der Induktionsschritt vollzogen und die Restformel bewiesen. \square

Beispiel. Die Funktion $f(x) = x^N$ hat an der Stelle $a = 0$ verschwindende Ableitungen bis zur Ordnung $N - 1$. Das Taylor-Polynom $T_{0,n}f(x)$ der Ordnung n dieser Funktion verschwindet daher für $n < N$. Nach der Restformel für das Taylor-Polynom gibt es Zahlen ξ_n derart

$$\begin{aligned} R_{n,0}f(x) &= f(x) = x^N = f^{(n+1)}(\xi_n) \frac{x^{n+1}}{(n+1)!} \\ &= \frac{N \cdot (N-1) \cdot (N-2) \cdots (N-n)}{1 \cdot 2 \cdot 3 \cdots (n+1)} \xi_n^{N-n-1} = \binom{N}{n+1} \xi_n^{N-n-1} \\ \Rightarrow \quad \xi_n &= \binom{N}{n+1}^{\frac{-1}{N-n-1}} x^{\frac{N}{N-n-1}}. \end{aligned}$$
 \circlearrowright

Fehlerabschätzungen

Im Hinblick auf die nachfolgende Diskussion des Fehlers des Interpolationspolynoms formulieren wir dies noch als eine Fehlerabschätzung. Das Taylor-Polynom approximiert die Funktion $f(x)$ durch ein Polynom vom Grad n . Der Fehler des Taylor-Polynoms ist

$$|f(x) - T_{n,a}f(x)| = \left| f^{(n+1)}(\xi) \frac{(x-a)^{n+1}}{(n+1)!} \right| = |f^{(n+1)}(\xi)| \cdot \left| \frac{(x-a)^{n+1}}{(n+1)!} \right| \leq \max_{\xi \in [a,x]} |f^{(n+1)}(\xi)| \left| \frac{(x-a)^{n+1}}{(n+1)!} \right|,$$

er ist also beschränkt einerseits durch ein Polynom mit Grad $n+1$ und den grössten Wert der $(n+1)$ -te Ableitung im Intervall $[a, x]$. Die Fehlerformel für das Interpolationspolynom wird von der genau gleichen Art sein.

Fehler des Lagrange-Interpolationspolynoms

Der folgende Satz gibt vollständige Auskunft über den Fehler des Interpolationspolynoms.

Satz 3.13. Sei p ein Polynom vom Grad n , welches mit der $n+1$ -mal differenzierbaren Funktion f an den $n+1$ Stellen

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

übereinstimmt. Dann gibt es für jedes $x \in [a, b]$ ein $\xi_x \in [a, b]$ mit

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} l(x). \tag{3.9}$$

Beweis. An den Stützstellen x_i ist $f(x_i) - p(x_i) = 0$ und $l(x_i) = 0$, die Gleichung (3.9) ist also trivialerweise erfüllt.

Sei jetzt also $x \in [a, b]$ verschieden von allen x_i . Da $l(x) \neq 0$ ist, gibt es eine Zahl c derart, dass

$$f(x) - p(x) = cl(x) \quad \Leftrightarrow \quad f(x) - p(x) - cl(x) = 0. \quad (3.10)$$

Die Funktion $g(x) = f(x) - p(x) - cl(x)$ verschwindet in allen Stützstellen x_i und zusätzlich auch noch im Punkt x , sie hat also $n + 1$ Nullstellen.

Nach dem Nullstellen-Schachtelungssatz 3.7 hat die $n + 1$ -te Ableitung von g eine Nullstelle im Intervall. Es gibt also eine Zahl $\xi_x \in [a, b]$ mit $g^{(n+1)}(\xi_x) = 0$.

Da p Grad n hat, ist die $n + 1$ -te Ableitung 0. Das Polynom $l(x)$ hat die Form

$$l(x) = x^{n+1} - (x_0 + x_1 + \cdots + x_{n-1} + x_n)x^{n-1} + \cdots + (-1)^{n+1}x_0x_1 \dots x_{n-1}x_n,$$

seine $n + 1$ -Ableitung ist die Konstanten $(n + 1)!$.

Die Folgerung $g^{(n+1)}(\xi_x) = 0$ wird damit zu

$$0 = f^{(n+1)}(\xi_x) - c(n + 1)! \quad \Rightarrow \quad c = \frac{f^{(n+1)}(\xi_x)}{(n + 1)!}.$$

Einsetzen dieses Wertes für c in (3.10) ergibt

$$f(x) - p(x) = cl(x) = \frac{f^{(n+1)}(\xi_x)}{(n + 1)!} l(x),$$

wie behauptet. □

Dieser Satz erlaubt den Fehler eines Interpolationspolynoms abzuschätzen, wenn die $n + 1$ -te Ableitung der Funktion f bekannt ist. Wir bezeichnen mit

$$\|g\| = \sup_{a \leq x \leq b} |g(x)|$$

die *Supremum-Norm* der Funktion g im Intervall $[a, b]$.

Korollar 3.14. Ist p ein Interpolationspolynom vom Grad n , welches mit der Funktion f in den Stellen $a = x_0 < x_1 < \cdots < x_{n+1} < x_n = b$ übereinstimmt, dann ist

$$|f(x) - p(x)| \leq \frac{\|f^{(n+1)}\|}{(n + 1)!} \|l(x)\|.$$

Der Fehler des Interpolationspolynoms vom Grad n ist also beschränkt durch das Polynom $l(x)$ vom Grad $n + 1$ und den grössten Wert der $(n + 1)$ -ten Ableitung von $f(x)$.

Beispiel. Die Funktion $f(x) = \sin x$ soll mit den Stützstellen $x_0 = 0$, $x_1 = \frac{\pi}{2}$ und $x_2 = \pi$ interpoliert werden. Das Interpolationspolynom ist ein quadratisches Polynom mit Nullstellen x_0 und x_2 , der Funktionswert bei x_1 muss 1 sein. Man kann sich davon überzeugen, dass das Polynom

$$p(x) = \frac{4}{\pi^2} x(\pi - x)$$

diese Eigenschaft hat (Abbildung 3.6). Wie gross ist der Fehler dieses Interpolationspolynoms?

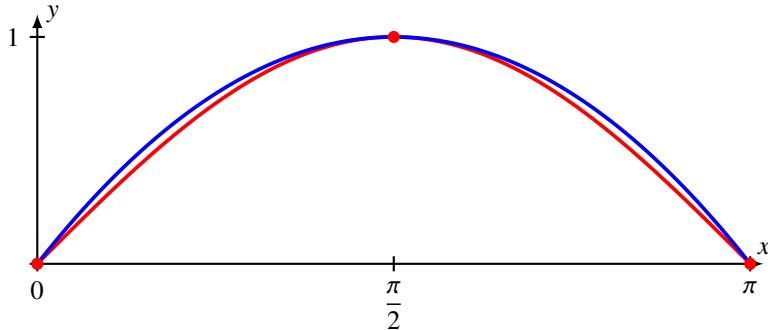


Abbildung 3.6: Interpolation der Funktion $f(x) = \sin x$ mit nur drei Stützstellen $x_0 = 0$, $x_1 = \frac{\pi}{2}$ und $x_2 = \pi$. Der Fehler ist deutlich kleiner als die Abschätzung mit Satz 3.13 erwarten lässt.

Die dritten Ableitungen der Funktion $f(x) = \sin x$ sind, bekannt, es ist $f^{(3)}(x) = -\cos x$. Der Betrag von $f^{(3)}(x)$ wird also nie grösser als 1. Es folgt, dass

$$|f(x) - p(x)| \leq \frac{1}{3!} l(x) = \frac{1}{6} |x(x - \frac{\pi}{2})(x - \pi)|$$

Die Ableitung des Polynoms auf der rechten Seite hat Nullstellen bei $\frac{\pi}{2} \pm \frac{\pi}{2\sqrt{3}}$, durch Einsetzen erhält man den maximalen Wert

$$\|f^{(3)}\| = \frac{\pi^3}{12\sqrt{3}} \approx 1.49179.$$

Wir schliessen, dass das Interpolationspolynom niemals um mehr als 0.24863 vom Funktionswert abweichen kann. \circ

3.2.3 Runges Phänomen

Das Korollar 3.14 besagt, dass der Fehler des Interpolationspolynoms unter anderem durch den Betrag von $l(x)$ begrenzt ist. Für äquidistante Stützstellen mit Abstand h kann man beobachten, dass die Oszillationen des Polynoms $l(x)$ gegen den Rand des Intervalls immer grösser werden. Für einen Punkt in der Mitte jedes Teilintervalls ist $h/2$ der kleinste mögliche Faktor in $l(x)$. Den grössten Faktor findet man für x im ersten oder letzten Teilintervall, er ist $b - a - h$. Außerdem treten mehrere ähnlich grosse Faktoren auf. Für x in einem Intervall nahe $(a + b)/2$ sind die Faktoren dagegen nur halb so gross. Diese Phänomen, dass die Amplitude der Oszillationen des Interpolationspolynoms $l(x)$ gegen den Rand des Intervalls immer grösser wird, ist auch als *Runge's Phänomen* bekannt.

3.2.4 Wahl der Stützstellen und Tschebyscheff-Interpolationspolynom

Die als Runge's Phänomen beschriebenen Oszillationen am Rande des Intervalls können verkleinert werden, wenn man dafür sorgt, dass mit grösserem Abstand von der Mitte des Intervalls der Abstand der Stützstellen untereinander ebenfalls verkleinert. Dies garantiert, dass neben den grossen Faktoren in der Nähe von $b - a$ auch wesentlich kleinere Faktoren auftreten, so dass die extremen Werte nahe den Intervallenden vergleichbar mit den Werten im Inneren des Intervalls werden.

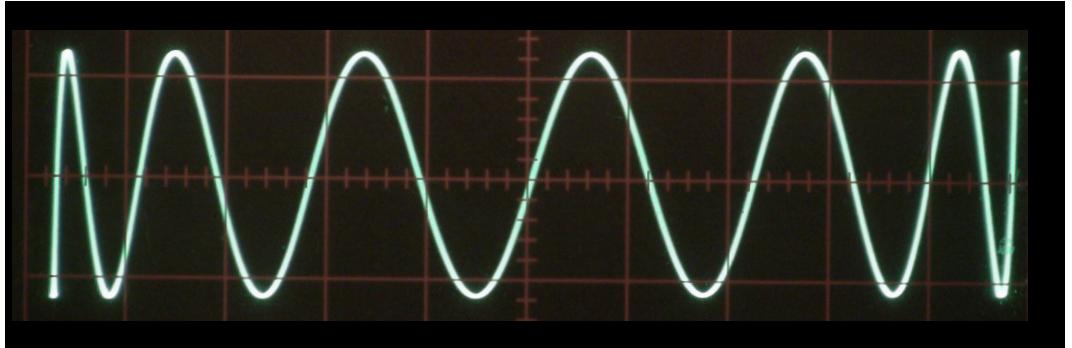


Abbildung 3.7: Diese Lissajous-Figur suggeriert eine mögliche Lösung für eine Interpolationspolynom mit besonders kleinem Fehler. Wenn sich diese Kurve als Polynom ausdrücken lässt, bleibt der Fehler über das ganze Definitionsintervall gleichmäßig beschränkt.

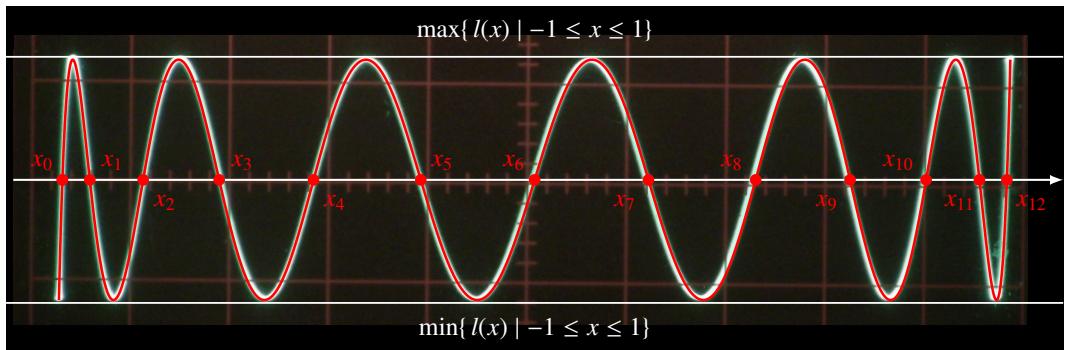


Abbildung 3.8: Lissajous-Figur von Abbildung 3.7 mit eingezeichnetem Tschebyscheff-Polynom und Nullstellen desselben. Das Maximum des Betrags des Tschebyscheff-Polynoms ist 1 und liefert damit eine obere Schranke für den Fehler des zugehörigen Interpolationspolynoms.

Die beste Approximation durch ein Interpolationspolynom kann man also erwarten, wenn $l(x)$ im Intervall $[a, b]$ keine besonders grossen Werte annimmt. Eine Funktion ähnlich wie $\sin x$ oder $\cos x$, die unendlich viele Extremewerte ± 1 haben, würde dieses Kriterium erfüllen, aber $\sin x$ und $\cos x$ sind keine Polynome. Sie sind auch auf einem viel grösseren Intervall als nötig definiert, nämlich ganz \mathbb{R} . Ein verwandtes Beispiel sind Lissajous-Figuren. Abbildung 3.7 suggeriert, dass eine geeignete Lissajous-Figur als Graph für ein Interpolationspolynom mit sehr geringem Fehler dienen könnte, wenn man sie als Polynom darstellen kann. Eine solche Lissajous-Figur entsteht als Kurve $\gamma_n: t \mapsto (\cos t, \cos nt)$ oder eine anderes trigonometrisches Polynom als zweite Komponente. Die Abbildung 3.8 zeigt diese Kurve $\gamma_n(t)$ für den Fall $n = 13$ der Lissajous-Figur von Abbildung 3.7 überlagert. Die gute Übereinstimmung bestätigt die obige Beobachtung.

Die Tschebyscheff-Polynome

Es stellt sich also die Frage, ob $\cos nt$ als Polynom in $z = \cos t$ ausgedrückt werden kann. Sei also

$$T_n(x) = T_n(\cos t) = \cos nt.$$

Für kleine n kann man unmittelbar verifizieren, dass $T_n(z)$ ein Polynom ist:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= \cos 2t = 2 \cos^2 t - 1 = 2x^2 - 1 \quad \text{und} \\ T_3(x) &= \cos 3t = 4 \cos^3 t - 3 \cos t = 4x^3 - 3x. \end{aligned} \tag{3.11}$$

Für kleine Werte von n ist also direkt nachprüfbar, dass $T_n(x)$ ein Polynom ist.

Wir benötigen eine Formel, mit der sich Werte der Polynome $T_n(x)$ effizient berechnen lassen, ohne dass das Polynom in expliziter Form ermittelt werden muss. Im Folgenden soll dazu eine Rekursionsformel hergeleitet werden.

Aus den Additionstheoremen für den Kosinus folgt die Formel für die Summe von zwei Konsinus-Funktionen

$$\begin{aligned} \cos(n+1)t + \cos(n-1)t &= 2 \cos \frac{(n+1)t + (n-1)t}{2} \cos \frac{(n+1)t - (n-1)t}{2} = 2 \cos nt \cos t \\ T_{n+1}(x) + T_{n-1}(x) &= 2T_n(x)x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x). \end{aligned} \tag{3.12}$$

Wenn $T_n(x)$ und $T_{n-1}(x)$ Polynome sind, dann ist auch $T_{n+1}(x)$ ein Polynom. Aus den bereits bekannten Polynomen (3.11) und der Rekursionsformel (3.12) folgt jetzt mit vollständiger Induktion, dass alle $T_n(x)$ Polynome sind. Sie heißen *Tschebyscheff-Polynome*. Die Rekursionsformel kann dazu verwendet werden, die Polynome explizit zu berechnen. Zum Beispiel folgt für die nächsten drei Polynome

$$\begin{aligned} T_4(x) &= 8x^4 - 8x^2 + 1, \\ T_5(x) &= 16x^5 - 20x^3 + 5x \quad \text{und} \\ T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1. \end{aligned}$$

Da das Interpolationspolynom den führenden Koeffizienten 1 haben muss, muss $l(x) = 2^{1-n}T_n(x)$ gewählt werden.

Tschebyscheff-Stützstellen

Die Polynome $T_n(x)$ sind eigentlich nicht nötig, da für die Konstruktion des Interpolationspolynoms nur die Nullstellen nötig sind. Wegen $T_n(x) = \cos nt$ liegt eine Nullstelle genau dann vor, wenn $nt = \frac{\pi}{2} + k\pi$, $k \in \mathbb{Z}$. Die zugehörigen Werte von z sind

$$x_k = \cos t = \cos \frac{\pi(2k+1)}{2n}.$$

In Abbildung 3.9 sind die Polynome $2^{n-1}l(x)$ vom Grad n oben für äquidistante Stützstellen und unten für Tschebyscheff-Stützstellen im Vergleich dargestellt. Wie in der Einleitung zu diesem Abschnitt angekündigt, oszillieren die Polynome für äquidistante Stützstellen nahe den Intervallenden. Für Tschebyscheff-Stützstellen wird $2^{n-1}l(x)$ betragsmäßig nie grösser als 1.

Abbildung 3.10 zeigt die Basispolynome vom Grad 7 $l_j(x)$ für Tschebyscheff-Stützstellen. Da bei Verwendung von Tschebyscheff-Stützstellen die Polynome $l(x)$ keine ausgeprägten Oszillationen an den Intervallenden aufweisen, sind auch die Basispolynome $l_j(x)$ vor allem in der Nähe der jeweiligen Stützstelle x_j wesentlich von 0 verschieden.

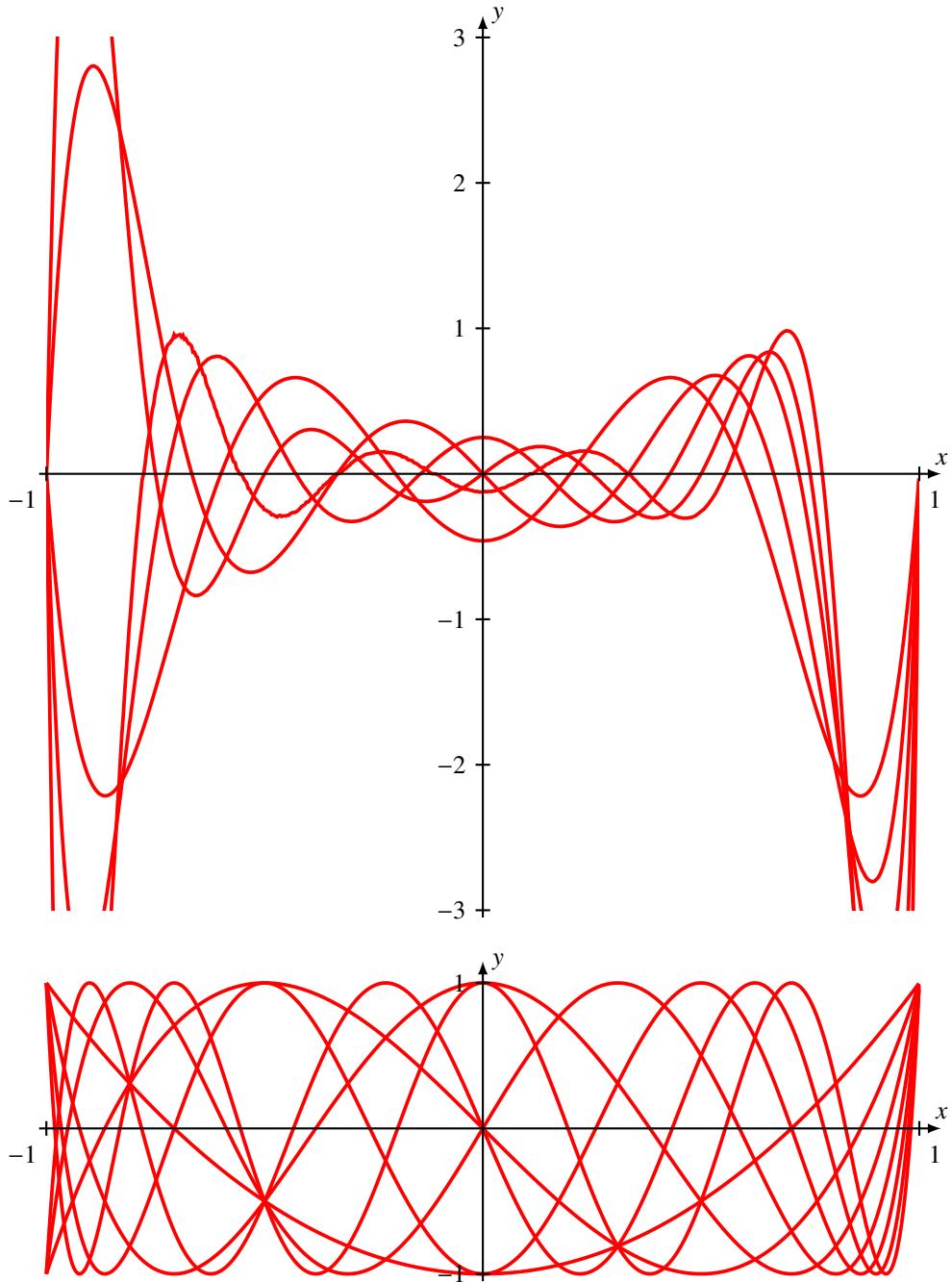


Abbildung 3.9: Vergleich der Oszillationen bei äquidistanten Stützstellen (oben) und bei Tschebyscheff-Stützstellen. Damit die Abweichungen sichtbar werden, sind die Polynome $l(x)$ vom Grad n mit dem Faktor 2^{n-1} skaliert. Bei Verwendung von Tschebyscheff-Stützstellen wächst $2^{n-1}l(x)$ nie über 1 an, während bei äquidistanten Stützstellen die in der Einleitung zu diesem Abschnitt diskutierten Oszillationen nahe den Intervallenden auftreten.

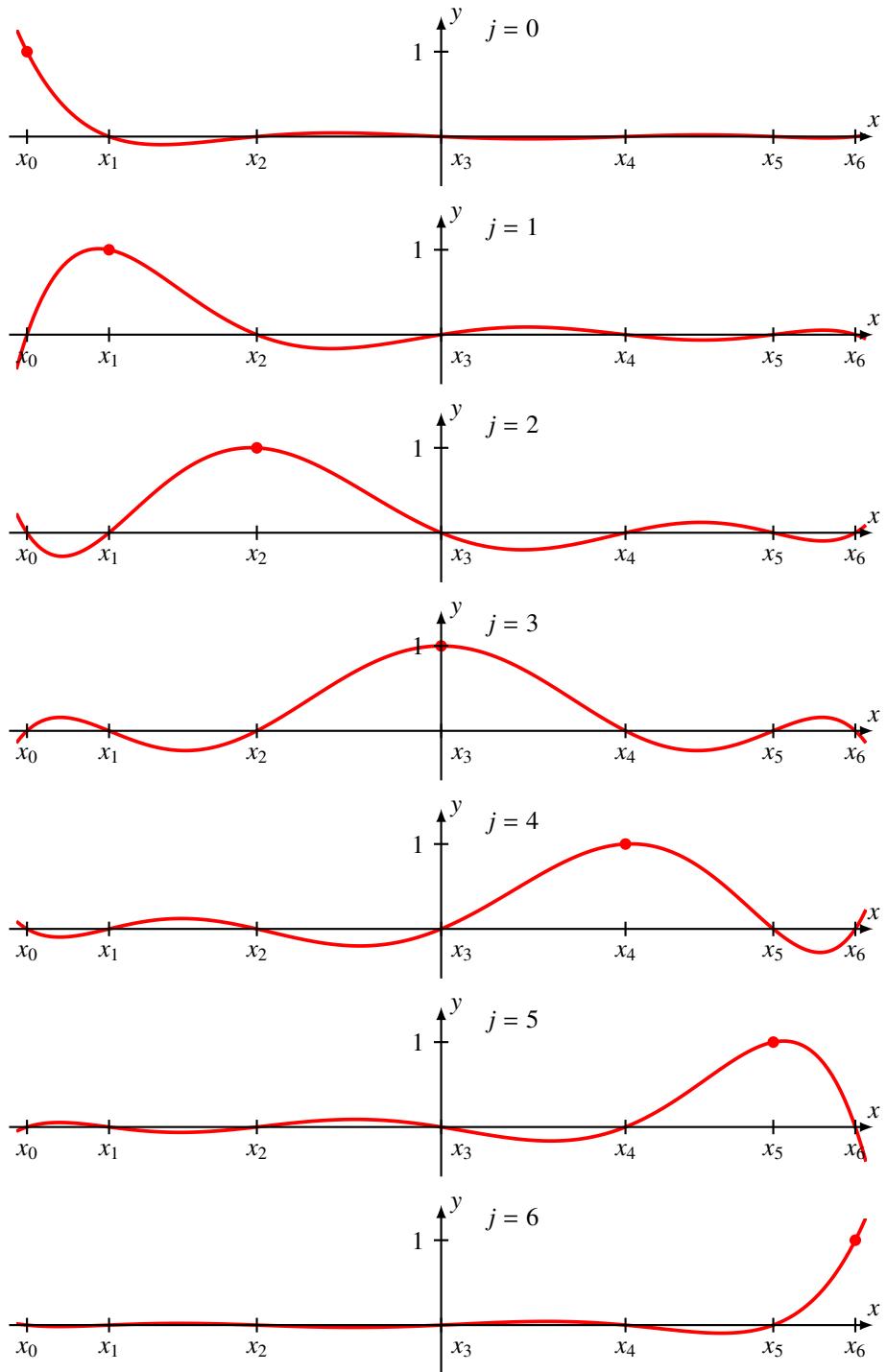


Abbildung 3.10: Basisinterpolationspolynome vom Grad 7 für Tschebyscheff-Stützstellen

Vergleich der Fehler

In diesem Abschnitt vergleichen wir die Fehler der Interpolationspolynome für die Funktion

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

nach Lagrange mit äquidistanten Stützstellen und mit Tschebyscheff-Stützstellen.

Die Resultate sind in Abbildung 3.11 für das Lagrange-Interpolationspolynom und Abbildung 3.12 für das Tschebyscheff-Interpolationspolynom dargestellt. In beiden Abbildungen wird die gleiche Anzahl Stützstellen verwendet. Beim Lagrange-Interpolationspolynom nimmt der Fehler zunächst schnell ab. Er ist, wie das Runge Phänomen erwarten lässt, immer am grössten im äussersten Teilintervall. Für grösser werdende Anzahl von Stützstellen wird die Berechnung des Interpolationspolynoms in diesen äussersten Teilintervallen instabil und wächst zum Beispiel von $n = 30$ zu $n = 34$ um eine Größenordnung an. Für eine grosse Zahl von Stützstellen kann das Lagrange-Interpolationspolynom also mindestens am Rand des Intervalls keine zuverlässigen Approximation sein.

Die Verwendung von Tschebyscheff-Stützstellen verbessert die Genauigkeit schon bei $n = 16$ Stützstellen um zwei Größenordnungen. Außerdem sind die Fehler über das ganze Intervall gleichmässig verteilt. Bei einer grösseren Anzahl von Stützstellen scheint der Fehler vor allem linken Rand stark anzuwachsen. Da die Funktionswerte aber ungefähr 1 sind, ist ein Fehler von 10^{-16} genau der typische Rundungsfehler des `double` Datentyps. Man sieht hier als nicht den Fehler des Interpolationspolynoms sondern die Grenzen der Maschinengenauigkeit.

3.3 Hermite-Interpolation

Das Lagrange-Interpolationspolynom nimmt zwar in unmittelbarer Nähe der Stützstellen zuverlässig Funktionswerte nahe den gegeben Werten an, doch insbesondere gegen den Rand des Intervalls können die oft beobachteten Oszillationen eine schlechte Approximation bewirken. Im Gegensatz zur Taylor-Reihe, deren Ableitung mindestens in der Nähe des Entwicklungspunktes auch mit der Ableitung der zu approximierenden Funktion übereinstimmt, gibt es für das Interpolationspolynom keine solche Garantie. Beide Schwierigkeiten könnten gemildert werden, indem gefordert wird, dass das Polynom nicht nur die gleichen Funktionswerte, sondern auch die gleichen Ableitungen bis zu einer bestimmten Ordnung haben soll. Dies ist die Idee der *Hermite-Interpolation*, die in diesem Abschnitt vorgestellt werden soll.

3.3.1 Aufgabenstellung

Das Hermite-Interpolationspolynom löst die folgende Approximationsaufgabe.

Aufgabe 3.15 (Hermite-Interpolationspolynom). *Gegeben Stützstellen*

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

und Funktionswerte f_i , $0 \leq i \leq n$, und Werte $s_i^{(k)}$ der k -ten Ableitungen bis zur m -ten Ordnung, $1 \leq k \leq m$, finde ein Polynom h , mit

$$h(x_i) = f_i, \quad h^{(k)}(x_i) = s_i^{(k)}, \quad 0 \leq i \leq n, 1 \leq k \leq m. \quad (3.13)$$

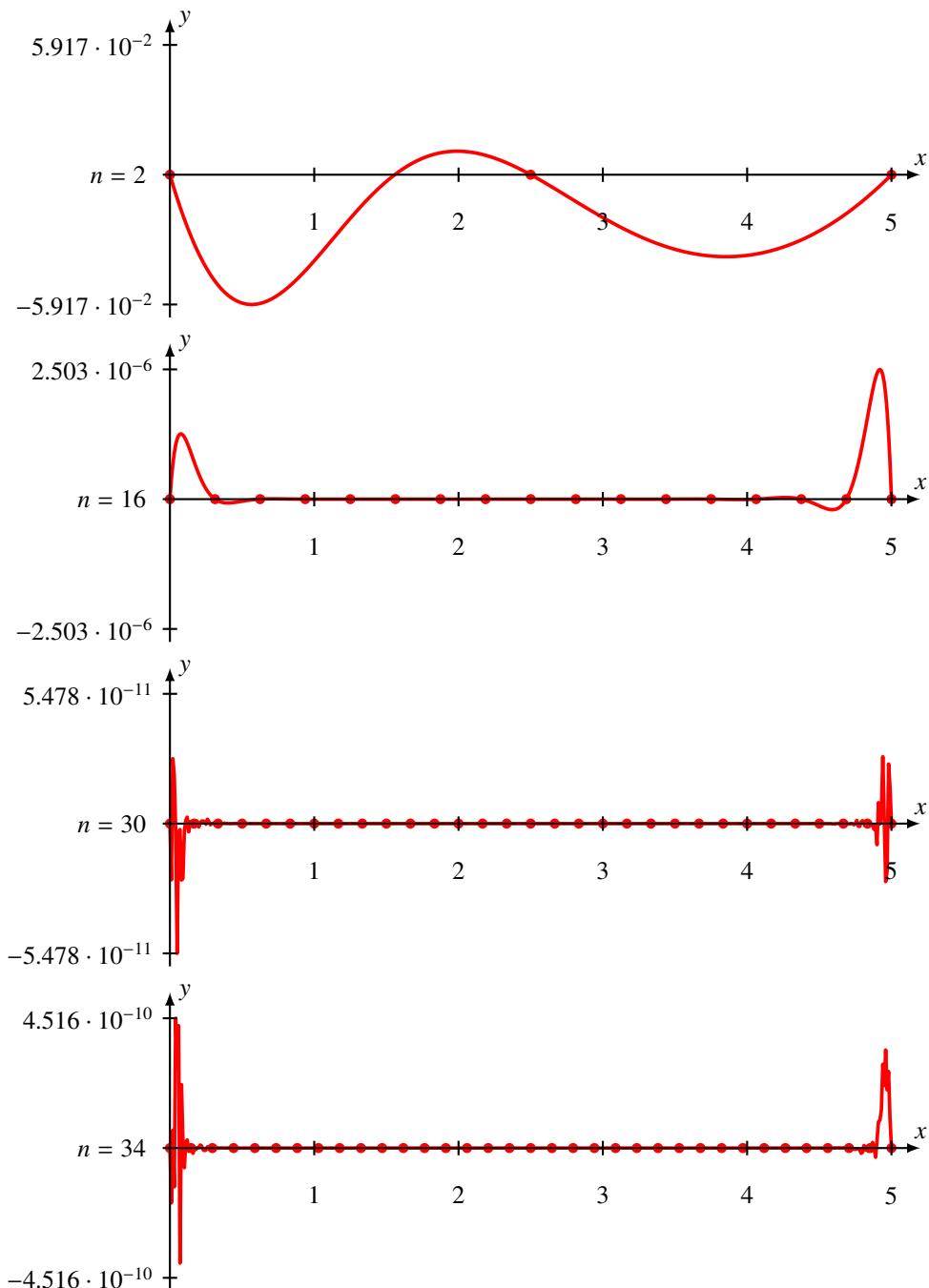


Abbildung 3.11: Fehler des Lagrange-Interpolationspolynoms für die Funktion $f(x) = e^{-x^2/2} / \sqrt{2\pi}$. Der Fehler nimmt mit der Anzahl der Stützstellen bis $n = 30$ ab, danach wird die Berechnung instabil und der Fehler nimmt wieder zu.

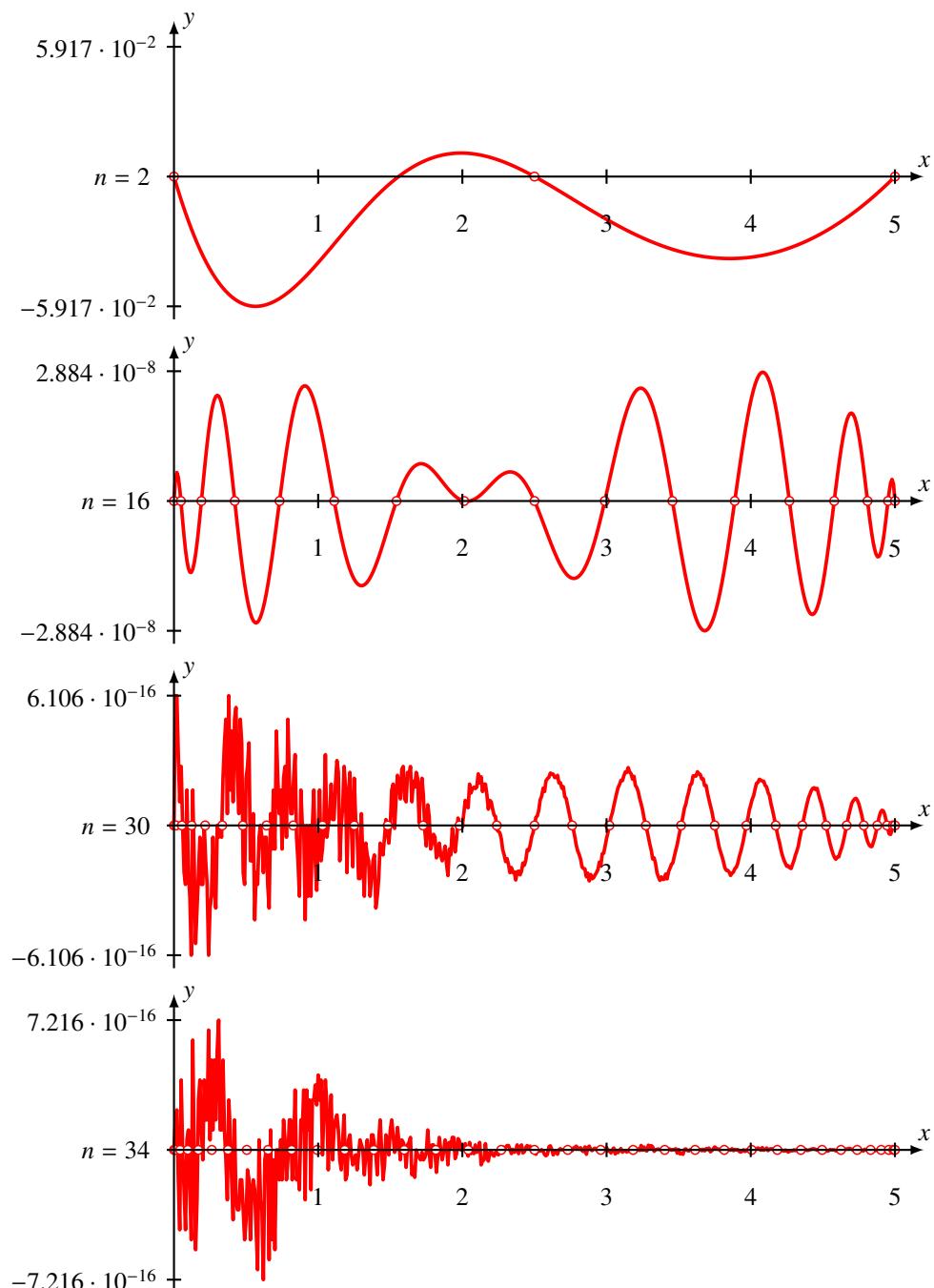


Abbildung 3.12: Fehler des Interpolationspolynomes für die Funktion $f(x) = e^{-x^2/2} / \sqrt{2\pi}$ mit Stützstellen nach Tschebyscheff. Der Fehler bleibt über das ganze Intervall gleichmässig. Für eine grosse Zahl von Stützstellen erreicht die Interpolation die Maschinengenauigkeit.

Die Aufgabenstellung formuliert $N = (n + 1)(k + 1)$ Bedingungen für das Polynom h , es braucht also im Allgemeinen ein Polynom mindestens vom Grade $N = (n + 1)(k + 1)$, um alle diese Bedingungen erfüllen zu können. Ein elementarer Ansatz könnte sein, eine Polynom in der Form $a_N x^N + a_{N-1} x^{N-1} + \dots + a_1 x + a_0$ anzusetzen, die Bedingungen (3.13) als lineare Gleichungen für die Koeffizienten auszuschreiben und das Gleichungssystem zu lösen. Dieses Vorgehen ist allerdings sehr aufwendig und numerisch nicht besonders stabil. Ein Weg analog zur Bestimmung des Lagrange-Interpolationspolynomes in Abschnitt 3.2.1 ist daher angezeigt.

3.3.2 Bestimmung des Hermite-Interpolationspolynom

Wir führen die Konstruktion nur für den Fall $m = 1$ durch, also für Interpolationspolynome, die den Funktionswerten und ersten Ableitungen übereinstimmen. Wie in Abschnitt 3.2.1 suchen wir zunächst wieder eine Lösung des folgenden *speziellen Interpolationsproblems*.

Aufgabe 3.16 (Spezielles Hermite-Interpolationsproblem). *Gegeben Stützstellen*

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

finde Polynome h_j und h_j^1 vom Grad höchstens $2n + 1$ derart, dass

$$\left. \begin{array}{l} h_j(x_i) = \delta_{ij} \\ h'_j(x_i) = 0 \end{array} \right\} \forall i, j \quad \text{und} \quad \left. \begin{array}{l} h_j^1(x_i) = 0 \\ h_j^{1'}(x_i) = \delta_{ij} \end{array} \right\} \forall i, j$$

Lösung. Ein Polynom vom Grad $2n + 2$, welches in allen Stützstellen eine doppelte Nullstelle hat, ist das Produkt

$$(x - x_0)^2(x - x_1)^2(x - x_2)^2 \dots (x - x_{n-1})^2(x - x_n)^2.$$

Die gesuchten Polynome h_j^1 haben in jeder Stützstelle außer in x_j ein doppelte Nullstelle, die Nullstelle in x_j muss einfach sein. Ein solches Polynom kann man erhalten, indem man einen der Faktoren $(x - x_j)$ weglässt, oder zu

$$p_j(x) = (x - x_0)^2(x - x_1)^2 \dots (\widehat{x - x_k})^2 \dots (x - x_{n-1})^2(x - x_n)^2$$

einen solchen Faktor hinzufügt:

$$h_j^1(x) = c_j^2(x - x_j)p_j(x),$$

die Konstante c_j^2 muss passend gewählt werden, damit die Ableitung

$$h_j^{1'}(x) = c_j^2 \underbrace{\frac{d}{dx}(x - x_j)}_{= 1} + c_j^2(x - x_j) \frac{d}{dx} p_j(x)$$

den richtigen Wert bekommt. An der Stelle $x = x_j$ fällt der zweite Term weg und es bleibt

$$h_j^{1'}(x_j) = c_j^2 p_j(x_j).$$

und damit ist $c_j^2 = 1/p_j(x_j)$. Dies ist das Quadrat des entsprechenden Normierungsfaktors, der beim Lagrange-Interpolationspolynom zur Anwendung kam.

Die Polynome h_j haben in allen Stützstellen außer x_j eine doppelte Nullstelle. Das Produkt $p_j(x)$ teilt diese Eigenschaft. Da es vom Grad $2n$ ist, haben wir nur die Freiheit, einen Linearfaktor

der Form $(u_j(x - x_j) + v_j)$ hinzuzufügen, um h_j zu erhalten. Es müssen also u_j und v_j so gewählt werden, dass für

$$h_j(x) = (u_j(x - x_j) + v_j)p_j(x) \quad \text{die Gleichungen} \quad \begin{cases} h_j(x_j) = v_j p_j(x_j) = 1 \\ h'_j(x_j) = u_j p_j(x_j) + v_j p'_j(x_j) = 0 \end{cases} \quad (3.14)$$

gelten. Aus der ersten Gleichung folgt $v_j = 1/p_j(x_j) = c_j^2$, aus der zweiten

$$u_j = -\frac{p'_j(x_j)}{p_j(x_j)^2} = -c_j^4 p'_j(x_j).$$

Einsetzen in die (3.14) ergibt

$$h_j(x) = \left(-\frac{p'_j(x_j)}{p_j(x_j)^2}(x - x_j) + \frac{1}{p_j(x_j)} \right) p_j(x) = \frac{p'_j(x_j)(x - x_j) + p_j(x_j)}{p_j(x_j)^2} p_j(x)$$

Andererseits ist $p_j(x)(x - x_j) = h_j^1(x)/c_j^1$, man kann also auch

$$h_j(x) = \frac{p_j(x)}{p_j(x_j)} - \frac{p'_j(x_j)}{c_j^1 p_j(x_j)^2} h_j^1(x) \quad (3.15)$$

schreiben. Der erste Term in (3.15) ist das Quadrat des Lagrange-Interpolationspolynoms $l_j(x)$. \square

Mit der Lösung des speziellen Interpolations-Problems findet man jetzt auch eine Lösung für das allgemeine Problem. Das gesuchte Interpolationspolynom ist

$$h(x) = \sum_{j=0}^n f_j h_j(x) + \sum_{j=0}^n s_j h_j^1(x).$$

3.3.3 Zwei Stützstellen

Der Fall zweier Stützstellen x_0 und x_1 ist von einiger praktischer Bedeutung. Er wird zum Beispiel im Abschnitt 3.5 zum Einsatz kommen. Die Polynome h und h^1 sollen daher für diesen Fall explizit berechnet werden.

Die Polynome p_0 und p_1 sind

$$p_0(x) = (x - x_1)^2 \quad \text{und} \quad p_1(x) = (x - x_0)^2,$$

woraus $c_0^2 = c_1^2 = (x_1 - x_0)^2$ folgt.

Die Polynome h_0^1 und h_1^1 entstehen durch geeignete Normierung der Polynome $(x - x_0)p_0(x)$ und $(x - x_1)p_1(x)$, also

$$h_0^1(x) = \frac{(x - x_0)(x - x_1)^2}{(x_0 - x_1)^2} \quad \text{beziehungsweise} \quad h_1^1(x) = \frac{(x - x_0)^2(x - x_1)}{(x_0 - x_1)^2}.$$

Für die Polynome h_0^1 und h_1^1 sind die Konstanten u_0 und u_1 zu bestimmen. Die Ableitung der Polynome p_j sind

$$p'_0(x) = 2(x - x_1) \quad \text{und} \quad p'_1(x) = 2(x - x_0)$$

und damit ist

$$u_0 = -\frac{p'_0(x_0)}{(x_0 - x_1)^4} = -2 \frac{x_0 - x_1}{(x_0 - x_1)^4} = -\frac{2}{(x_0 - x_1)^3} \quad \text{und} \quad u_1 = -\frac{p'_1(x_1)}{(x_0 - x_1)^4} = \frac{2}{(x_0 - x_1)^3}.$$

Aus (3.14) folgt jetzt

$$\begin{aligned} h_0(x) &= (u_0(x - x_0) + v_0)(x - x_1)^2 = -\frac{2}{(x_0 - x_1)^3}(x - x_0)(x - x_1)^2 + \frac{(x - x_1)^2}{(x_0 - x_1)^4} \\ h_1(x) &= (u_1(x - x_1) + v_1)(x - x_0)^2 = \frac{2}{(x_0 - x_1)^3}(x - x_1)(x - x_0)^2 + \frac{(x - x_0)^2}{(x_0 - x_1)^2}. \end{aligned}$$

Der Spezialfall $x_0 = 0$

In diesem Fall schreiben wir $m = x_1$ für die Intervalllänge und erhalten die Polynome

$$\begin{aligned} h_0^1(x) &= \frac{x(x - m)^2}{m^2} & h_1^1(x) &= \frac{x^2(x - m)}{m^2} \\ h_0(x) &= \frac{2x(x - m)^2}{m^3} + \frac{(x - m)^2}{m^4} & h_1(x) &= -\frac{2x^2(x - m)}{m^3} + \frac{x^2}{m^4}. \end{aligned} \tag{3.16}$$

Der Spezialfall $x_0 = 0, x_1 = 1$

Eine besonders einfache Form nehmen die Polynome h_j^0 und h_j^1 an, wenn man sie auf das Intervall $[0, 1]$ spezialisiert. Wir bezeichnen diese Polynome mit grossen Buchstaben, sie sind

$$\begin{aligned} H_0^1(x) &= x(1 - x)^2 = x^3 - 2x^2 + x & H_1^1(x) &= (1 - x)x^2 = x^3 - x^2 \\ H_0(x) &= (1 + 2x)(1 - x)^2 = 2x^3 - 3x^2 + 1 & H_1(x) &= (3 - 2x)x^2 = -2x^3 + 3x^2 \end{aligned} \tag{3.17}$$

Graphen dieser Polynome sind in Abbildung 3.13 dargestellt.

Diese Polynome können auch verwendet werden, die Polynome für ein beliebiges Intervall wieder zu gewinnen. Dazu setzen wir $x = (x - x_0)/m$ in die Polynome ein. Die Polynome $h_0(x) = H_0((x - x_0)/m)$ und $h_1(x) = H_1((x - x_0)/m)$ haben die Werte

$$\begin{aligned} h_0(x) &= H_0((x - x_0)/m) \Big|_{x=x_0} = H_0(0) = 1 \quad \text{und} \quad h_0(x) = H_0((x - x_0)/m) \Big|_{x=x_1} = H_0(1) = 0 \\ h_1(x) &= H_1((x - x_0)/m) \Big|_{x=x_0} = H_1(0) = 0 \quad \text{und} \quad h_1(x) = H_1((x - x_0)/m) \Big|_{x=x_1} = H_1(1) = 1 \end{aligned}$$

und Ableitungen

$$h_i''(x_0) = \frac{d}{dx} H_i((x - x_0)/m) \Big|_{x=x_0} = H_i'((x - x_0)/m) \frac{1}{m} \Big|_{x=x_0} = \frac{H_i'(0)}{m} = 0$$

an den Intervallenden.

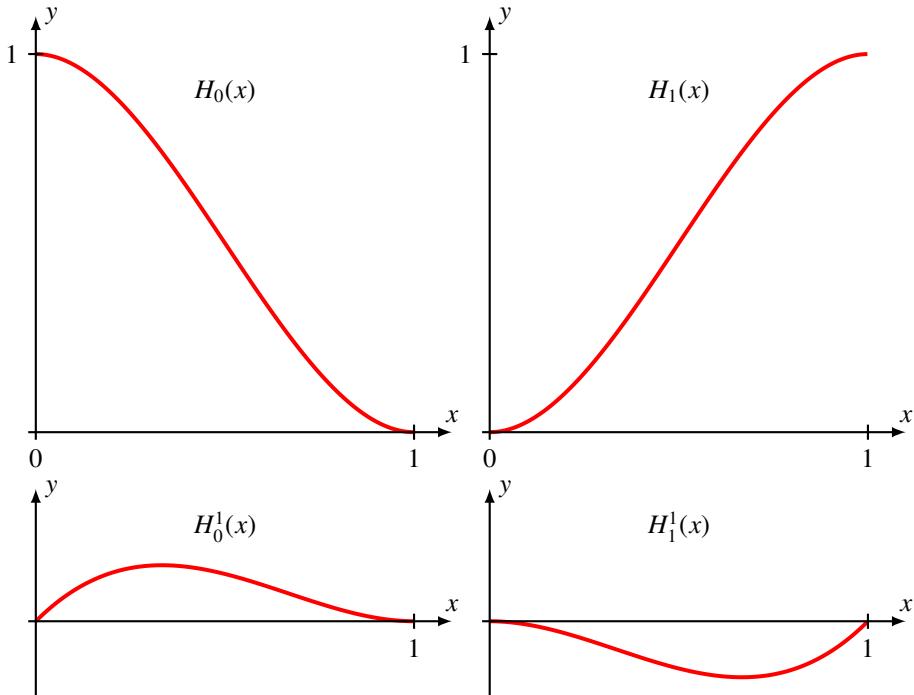
Tun wir dasselbe für die Polynome H_0^1 und H_1^1 , erhalten wir

$$h_j^{1'}(x_i) = \frac{d}{dx} H_j^1((x - x_0)/m) \Big|_{x=x_i} = H_j^{1'}((x - x_0)/m) \Big|_{x=x_i} \frac{1}{m} = \frac{1}{m} H_j^{1'}(i) = \frac{1}{m} \delta_{ij},$$

dies ist bis auf den Faktor $1/m$ korrekt. Daraus lesen wir ab, dass wir die Polynome

$$h_j^1(x) = m H_j^1((x - x_0)/m)$$

für die Ableitungen verwenden müssen.

Abbildung 3.13: Hermite-Basispolynome für das Intervall $[0, 1]$ nach (3.17)

Zweite Ableitungen

Für die spätere Anwendung bei der Spline-Interpolation untersuchen wir auch noch die zweiten Ableitung des Hermite-Interpolationspolynoms im Fall zweier Stützstellen am Rande des Intervalls. Wir tun dies für die Polynome (3.17) und kümmern uns später darum, was auf anderen Intervallen passiert. Wir erhalten die Werte

$$\begin{array}{ll} H_0''(0) = -6 & H_0''(1) = 6 \\ H_1''(0) = 6 & H_1''(1) = -6 \\ H_0^{1''}(0) = -4 & H_0^{1''}(1) = 2 \\ H_1^{1''}(0) = -2 & H_1^{1''}(1) = 4. \end{array}$$

Unter Verwendung der Substitution $x \rightarrow (x - x_0)/m$ können wir jetzt auch die Werte für die zweiten Ableitungen an den Intervallenden für h_j und h_j^1 bestimmen. Dazu berechnen wir erst die zweite Ableitung einer Funktion $f((x - x_0)/m)$:

$$\frac{d^2}{dx^2} f((x - x_0)/m) = \frac{d}{dx} f'((x - x_0)/m) \frac{1}{m} = f''((x - x_0)/m) \frac{1}{m^2}.$$

Angewendet auf die oben gefundenen Polynome bedeutet dies,

$$\begin{aligned} h_0''(x_0) &= -6/m^2 & \text{und} & \quad h_0''(x_1) = 6/m^2 \\ h_1''(x_0) &= 6/m^2 & \text{und} & \quad h_1''(x_1) = -6/m^2 \\ h_0^{1''}(x_0) &= -4/m & \text{und} & \quad h_0^{1''}(x_1) = 2/m \\ h_1^{1''}(x_0) &= -2/m & \text{und} & \quad h_1^{1''}(x_1) = 4/m. \end{aligned}$$

3.4 Baryzentrische Formeln für Interpolationspolynome

Die Interpolationspolynome von Lagrange und Hermite haben in der bis jetzt gezeigten Form das folgende grundlegende Problem. Sie sind definiert über das Produkt

$$l(x) = (x - x_0)(x - x_1) \dots (x - x_{n-1})(x - x_n).$$

Ist die Zahl der Stützstellen gross und erstrecken sich die Stützstellen über einen grossen Bereich, dann sind einzelne Faktoren $(x - x_i)$ immer gross, wie wir bei der Diskussion von Runges Phänomen bereits diskutiert haben. Zudem tritt bei der Berechnung eines Wertes in unmittelbarer Nähe der Stützstelle x_j in dem Faktor $(x - x_j)$ Auslöschung auf. Der grosse relative Fehler dieses Faktors wird durch die anderen Faktoren zu einem grossen absoluten Fehler aufgeblasen.

Andererseits ist klar, dass sich das Interpolationspolynom vor allem in der Nähe einer Stützstelle ändern sollte, wenn man den Wert an der Stützstelle ändert. Die anderen Stützstellen sollten also nur einen geringen Einfluss auf den Wert des Interpolationspolynoms haben. Dies geht aus der bisherigen Form des Interpolationspolynoms ebenfalls nicht hervor.

Gesucht ist also eine Form des Interpolationspolynoms, welche einsichtig macht, dass Änderungen von Stützwerten sich vor allem in der Nähe der betroffenen Stützstelle auswirken und die auch bei einer grossen Zahl von Stützstellen stabil sind.

Früher wurde gezeigt, dass das Interpolationspolynom für Funktionswerte f_j an den Stützstellen x_j durch die Linearkombination

$$p(x) = \sum_{j=0}^n f_j l_j(x)$$

gegeben ist. Für die Polynome $l_j(x)$ wurde

$$l_j(x) = \frac{(x - x_0)(x - x_1) \cdots (\widehat{x - x_j}) \cdots (x - x_n)}{(x_j - x_0)(x_j - x_1) \cdots (\widehat{x_j - x_j}) \cdots (x_j - x_n)}$$

gefunden. Schreibt man

$$w_j = \frac{1}{\prod_{\substack{k=1 \\ k \neq j}}^n (x_j - x_k)},$$

dann kann man die Faktoren $l_j(x)$ auch als

$$l_j(x) = \frac{l(x)}{(x - x_j)} \cdot w_j$$

ausdrücken. Damit wird das Interpolationspolynom jetzt

$$p(x) = l(x) \sum_{j=0}^n \frac{w_j f_j}{x - x_j}. \quad (3.18)$$

Die Zahlen w_j hängen nur von den Stützstellen ab, nicht von den Funktionswerten f_j . Sie können also nach Festlegung der Stützstellen einmalig berechnet werden und verursachen danach keinen weiteren Berechnungsaufwand.

Das Interpolationspolynom wird besonders einfach, wenn alle Funktionswerte $f_j = 1$ sind. Da das konstante Polynom $p(x) = 1$ genau diese Werte annimmt, muss

$$1 = l(x) \sum_{j=0}^n \frac{w_j}{x - x_j}$$

gelten. Damit erhalten wir eine neue Darstellung für

$$l(x) = \frac{1}{\sum_{j=0}^n \frac{w_j}{x - x_j}}. \quad (3.19)$$

In dieser Form wird vermieden, dass zur Berechnung von $l(x)$ eine grosse Anzahl Produkte mit potentiell grossen Faktoren gebildet werden muss. Sorgen bereiten in der Produktdarstellung vor allem die Faktoren $x - x_j$ für x weit entfernt von x_j . Stattdessen wird in (3.19) eine Summe von Summanden gebildet, die klein sind, wenn x weit von x_j entfernt ist.

Die vorteilhafte Formulierung (3.19) kann nun dazu verwendet werden, auch eine verbesserte Formulierung für das Interpolationspolynom aufzustellen. Dazu ersetzen wir den Faktor $l(x)$ in (3.18) durch (3.19) und erhalten

$$p(x) = \frac{\sum_{j=0}^n \frac{w_j f_j}{x - x_j}}{\sum_{j=0}^n \frac{w_j}{x - x_j}}. \quad (3.20)$$

Diese Form des Interpolationspolynoms ist ein gewichtetes Mittel der Werte f_j , gewichtet mit den Gewichten $w_j/(x-x_j)$. Diese Gewichte sind klein für x weit weg von x_j , die grössten Gewichte haben die Funktionswerte f_j nahe bei x . Die Formel (3.20) ist daher eine numerisch besonders vorteilhafte Form der Auswertung eines Interpolationspolynoms.

3.5 Spline-Interpolation

Die Hermite-Interpolation ermöglicht Approximationspolynome zu finden, die sowohl Funktionswerte als auch Ableitungen an den Stützstellen mit der zu approximierenden Funktion gemeinsam haben. Dadurch wird der Fehler der Approximationspolynome zwar kleiner, aber es entsteht das zusätzliche Problem, dass die Ableitungen der Funktion bestimmt werden müssen.

Die Spline-Interpolation umgeht dieses Problem, indem sie an den Stützstellen nicht die gleichen Steigungen verlangt, sondern Steigungen, die zu einem möglichst “wenig gekrümmten” Graphen

des Approximationspolynoms führen, welches natürlich immer noch in den Stützstellen die vorgegebenen Werte annehmen soll. Die Steigungen in den Stützstellen sind also Lösungen eines Optimierungsproblems, welches nicht die “am besten passende”, sondern die “schönste” Kurve durch die Stützstellen sucht. Die so gefundene Interpolationsfunktion heisst *Spline-Funktion* und wird in diesem Abschnitt bestimmt.

3.5.1 Anforderungen and die interpolierende Funktion

Gegeben seien wie früher Punkte

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

auf und Funktionswerte f_i einer im übrigen unbekannten, aber ausreichend glatten Funktion $f: [a, b] \rightarrow \mathbb{R}: x \mapsto f(x)$, es ist also $f(x_i) = f_i$.

Gesucht ist eine stetige Funktion $g: [a, b] \rightarrow \mathbb{R}: x \mapsto g(x)$, die die folgenden natürliche Eigenarten haben soll:

1. Die Funktion g nimmt in allen Stützstellen die Werte der Funktion f an, es ist also $g(x_i) = f_i \forall 0 \leq i \leq n$.
2. Die Funktion g ist stetig differenzierbar im ganze Intervall. Insbesondere existiert die Ableitung $g'(x)$ in jedem Punkt x des Intervalls $[a, b]$, der Graph von g kann also keine “Knicke” haben.
3. Im Inneren jedes Teilintervall $[x_i, x_{i+1}]$ ist die Funktion g beliebig oft stetig differenzierbar und die einseitigen Grenzwerte an den Enden der Teilintervalle existieren:

$$\exists \lim_{x \rightarrow x_i^+} g^{(k)}(x) \quad \forall 0 \leq i < n \quad \text{und} \quad \exists \lim_{x \rightarrow x_i^-} g^{(k)}(x) \quad \forall 0 < i \leq n.$$

Es wird nicht verlangt, dass die rechts- und linksseitigen Grenzwerte an den inneren Stützstellen x_1, \dots, x_{n-1} übereinstimmen müssen.

4. Der Graph von g soll möglichst wenig gekrümmt sein. Da die zweite Ableitung einer Funktion ein Mass für die Krümmung des Graphen ist, kann dieses Kriterium dadurch realisiert werden, dass die Funktion g unter allen Funktionen, die die Bedingungen 1–3 erfüllen, das Integral

$$J(g) = \int_a^b (g''(x))^2 dx$$

minimieren soll.

Man beachte, dass nirgends verlangt wird, dass die Ableitungen von g an den Stützstellen irgendwie mit Werten der Funktion f oder ihrer Ableitung f' in Verbindung steht.

3.5.2 Das Optimierungsproblem

Zunächst ist nicht klar, ob das eben gestellte Optimierungsproblem überhaupt eine Lösung hat. In jedem Teilintervall $[x_i, x_{i+1}]$ geht es um ein Problem der folgenden Art. Gesucht ist eine Funktion,

die an den Intervallenden die vorgegebene Werte $g(x_i) = f_i$ und $g(x_{i+1}) = f_{i+1}$ annimmt, im Inneren des Intervalls beliebig oft stetig differenzierbar ist und zudem einen Integralausdruck

$$\int_{x_i}^{x_{i+1}} (g''(x))^2 dx$$

minimiert.

Diese Art von Problemen hat bereits Leonhard Euler in recht allgemeiner Form untersucht und zu diesem Zweck das Gebiet der Variationsrechnung geschaffen. Sie tauchen in der Physik zum Beispiel in der folgenden Form auf.

Beispiel. Ein Teilchen der Masse m bewegt sich entlang der y -Achse. Zur Zeit a befindet es sich bei f_0 , zur Zeit b bei f_n . Auf das Teilchen wirkt außerdem eine Kraft, die durch ihr Potential $V(y)$ beschrieben werden kann. Die Geschwindigkeit zur Zeit t ist $\dot{y}(t)$. Die Differenz von kinetischer und potentieller Energie ist die sogenannte Lagrange-Funktion

$$L(t, y, \dot{y}) = \frac{1}{2}m\dot{y}(t)^2 - V(y(t)). \quad (3.21)$$

In der Physik wird gezeigt, dass die Bewegung des Teilchens durch diejenige Funktion $y(t)$ beschrieben wird, welche das sogenannte Wirkungsintegral

$$\int_a^b L(t, y(t), \dot{y}(t)) dt$$

minimiert (Maupertuis-Prinzip, Prinzip der kleinsten Wirkung). ○

Die Euler-Lagrange-Gleichungen

Um zu zeigen, dass die Interpolationsfunktion g existiert, lösen wir daher das folgende, wesentlich allgemeinere Problem.

Satz 3.17. Sei $L(x, y, y_1)$ eine in allen Argumenten beliebig oft stetig differenzierbare Funktion auf $[a, b] \times \mathbb{R} \times \mathbb{R}$. Es gibt eine glatte Funktion $y(x)$, die in den Intervallenden vorgegebene Werte $y(a) = y_a$ und $y(b) = y_b$ annimmt und außerdem das Integral

$$J(y) = \int_a^b L(x, y(x), y'(x)) dx$$

minimiert, sie ist Lösung der Euler-Lagrange-Differentialgleichung

$$\frac{d}{dx} \frac{\partial L}{\partial y_1}(x, y(x), y'(x)) - \frac{\partial L}{\partial y}(x, y(x), y'(x)) = 0. \quad (3.22)$$

Richtungsableitung

Die Verbindung zwischen den Extrema von $J(y)$ und der Euler-Lagrange-Differentialgleichung entsteht durch eine Art "Ableitung" von $J(y)$ nach y , die in einem Minimum verschwinden soll. Zunächst ist allerdings zu klären, was diese Ableitung nach einer Funktion überhaupt bedeuten soll. Die Ableitung soll in linearer Näherung wiedergeben können, was passiert, wenn die Funktion $y(x)$ verändert wird. Es gibt natürlich unendlich viele Stellen, in denen die Funktion modifiziert werden

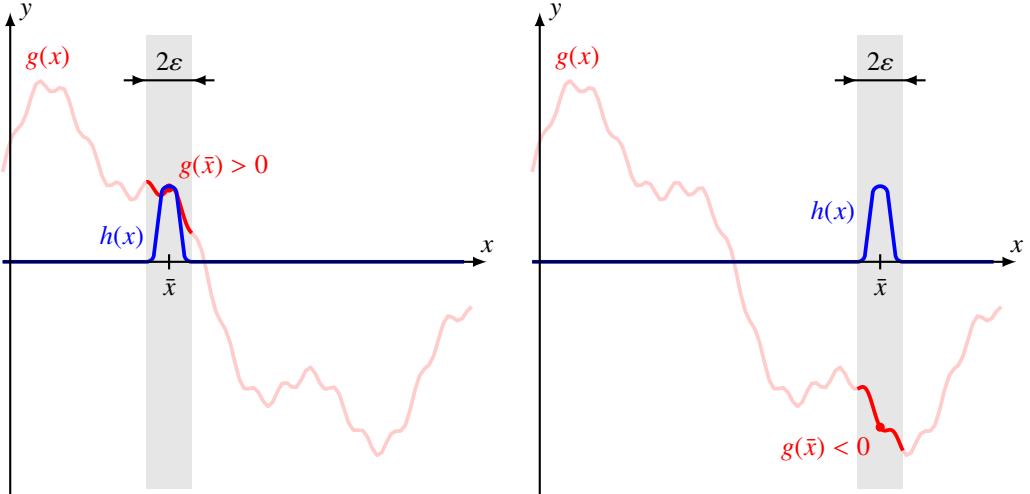


Abbildung 3.14: Wenn das Integral von $g(x)h(x)$ für jede Funktion $h(x)$ verschwinden, dann muss auch $g(x)$ verschwinden. In einer Umgebung eines Punktes \bar{x} , wo $g(\bar{x}) \neq 0$ ist, kann man $h(x)$ derart von 0 verschieden wählen (blaue Kurve), dass das Integral nicht verschwindet. Der Widerspruch zeigt, dass $g(\bar{x}) = 0$ sein muss.

kann, es handelt sich also um einen Ableitungsbegriff in einem unendlichdimensionalen Vektorraum. Die geometrische Intuition, die für den Begriff des Gradienten verwendet wird, kann in der unendlichdimensionalen Situation versagen, wir müssen also etwas vorsichtiger sein.

Ungefährlich ist, die Veränderung von $J(y)$ nur in einer Richtung zu untersuchen. Dazu ersetzen wir $y(x)$ durch $y_\varepsilon(x) = y(x) + \varepsilon h(x)$ und planen, den Parameter ε zu verändern. Es ist natürlich $y(x) = y_0(x)$. Damit $y_\varepsilon(x)$ immer noch ein möglicher Kandidat für das Minimum von $J(y)$ ist, müssen die Endpunkte von $y_\varepsilon(x)$ gleich sein, es muss also

$$\begin{aligned} y_\varepsilon(a) &= y(a) \\ y_\varepsilon(b) &= y(b) \end{aligned} \quad \Rightarrow \quad h(a) = h(b) = 0 \quad (3.23)$$

gelten.

Die Funktion $J(\varepsilon) = J(y + \varepsilon h) = J(y_\varepsilon)$ hängt jetzt nur noch vom Parameter ε ab, die Ableitung nach ε ist wohldefiniert. Ist außerdem y ein Minimum, dann muss die Ableitung von $J(y_\varepsilon)$ nach ε für $\varepsilon = 0$ verschwinden, also

$$0 = \frac{d}{d\varepsilon} J(\varepsilon) \Big|_{\varepsilon=0} = \frac{d}{d\varepsilon} J(y + \varepsilon h) \Big|_{\varepsilon=0}.$$

Dies muss für jede beliebige Funktion gelten, welche die Randbedingung (3.23) erfüllt.

Integralprinzip

Die Möglichkeit, die Funktion nur unter der minimalen Bedingung (3.23) frei zu wählen, schränkt die Funktion $y(x)$ ein. Dies ist der Inhalt des folgenden Prinzips

Lemma 3.18 (Integralprinzip). *Sei $g(x)$ eine stetige Funktion auf dem Intervall $[a, b]$. Ausserdem gilt*

$$\int_a^b g(x)h(x) dx = 0$$

für jede auf $[a, b]$ definierte stetige Funktion mit $h(a) = h(b) = 0$. Dann ist $g(x) = 0$.

Beweis. Wir führen die Annahme, dass es einen Punkt \bar{x} gibt mit $g(\bar{x}) \neq 0$, zu einem Widerspruch.

Nehmen wir also an, dass $g(\bar{x}) > 0$ ist. Dann ist $g(x) > 0$ wegen der Stetigkeit von g auch noch in einer ε -Umgebung von \bar{x} von 0 verschieden. Wir wählen $h(x)$ als stetige Funktion mit folgenden Eigenschaften

1. Die Funktion wird nirgends negativ: $h(x) \geq 0$.
2. $h(x)$ verschwindet ausserhalb der ε -Umgebung von \bar{x} .
3. $h(\bar{x}) = 1$.

Eine solche Funktion ist in Abbildung 3.14 links blau dargestellt. Da $g(x)$ in der Umgebung > 0 ist, folgt

$$\int_a^b g(x)h(x) dx = \int_{\bar{x}-\varepsilon}^{\bar{x}+\varepsilon} \underbrace{g(x)}_{> 0} \underbrace{h(x)}_{\geq 0} dx > 0,$$

im Widerspruch zu den Voraussetzungen.

Der Fall $g(\bar{x}) < 0$ wird analog dazu behandelt, wie in Abbildung 3.14 rechts dargestellt. \square

Beweis von Satz 3.17

Wir gehen wie folgt vor: wir zeigen zunächst, dass eine solche Funktion eine Differentialgleichung erfüllen muss. Dann beziehen wir uns auf bekannte Sätze der Theorie der gewöhnlichen Differentialgleichungen, die besagen, dass die Gleichung eine glatte Lösung hat.

Sei jetzt also $y(x)$ eine Funktion mit $y(a) = y_a$ und $y(b) = y_b$, die das Integral $J(y)$ minimiert. Ändern wir die Funktion ein klein wenig, dann muss der Wert von J zunehmen. Wir vollziehen die Änderung, indem wir eine Funktion $h(x)$ mit $h(a) = 0$ und $h(b) = 0$ addieren. Die Funktionen $y_\varepsilon = y + \varepsilon h$ erfüllen dann alle die Bedingung $y_\varepsilon(a) = y_a$ und $y_\varepsilon(b) = y_b$, insbesondere müssen sie alle einen Wert $J(y + \varepsilon h)$ ergeben, der grösser ist als $J(y)$. Insbesondere muss die Ableitung von $J(y + \varepsilon h)$ nach ε an der Stelle $\varepsilon = 0$ verschwinden.

Wir berechnen die Ableitung von $J(y + \varepsilon h)$ nach ε :

$$\begin{aligned} 0 = \frac{d}{d\varepsilon} J(y + \varepsilon h) \Big|_{\varepsilon=0} &= \frac{d}{d\varepsilon} \int_a^b L(x, y(x) + \varepsilon h(x), y'(x) + \varepsilon h'(x)) dx \Big|_{\varepsilon=0} \\ &= \int_a^b \frac{\partial L}{\partial y}(x, y(x), y'(x)) h(x) + \frac{\partial L}{\partial y_1} L(x, y(x), y'(x)) h'(x) dx \\ &= \int_a^b \frac{\partial L}{\partial y}(x, y(x), y'(x)) h(x) dx + \int_a^b \frac{\partial L}{\partial y_1} L(x, y(x), y'(x)) h'(x) dx \quad (3.24) \end{aligned}$$

Das zweite Integral enthält die Ableitung $h'(x)$, über die wir nicht viel wissen. Wir können diese aber durch partielle Integration los werden:

$$\int_a^b \frac{\partial L}{\partial y_1} L(x, y(x), y'(x)) h'(x) dx = \left[\frac{\partial L}{\partial y_1} L(x, y(x), y'(x)) h(x) \right]_a^b - \int_a^b \frac{d}{dx} \frac{\partial L}{\partial y_1} L(x, y(x), y'(x)) h(x) dx$$

h war so gewählt, dass die Werte, an den Intervallenden verschwinden, also $h(a) = h(b) = 0$. Der erste Terme verschwindet daher und es bleibt

$$= - \int_a^b \frac{d}{dx} \frac{\partial L}{\partial y_1} L(x, y(x), y'(x)) h(x) dx.$$

Einsetzen in (3.24) ergibt die Gleichung

$$0 = - \int_a^b \left(\frac{d}{dx} \frac{\partial L}{\partial y_1}(x, y(x), y'(x)) - \frac{\partial L}{\partial y}(x, y(x), y'(x)) \right) h(x) dx. \quad (3.25)$$

Gleichung (3.25) muss für jede beliebige Funktion $h(x)$ gelten. Wir möchten zeigen, dass das nur möglich ist, wenn die grosse Klammer im Integral verschwindet. Dies ist aber genau die Situation des Integralprinzips von Lemma 3.18. Es gestattet uns zu schliessen, dass das Integral (3.25) nur dann für alle möglichen Funktionen $h(x)$ verschwindet, wenn die grosse Klammer im Integranden verschwindet. Es gilt daher die Gleichung

$$\frac{d}{dx} \frac{\partial L}{\partial y_1}(x, y(x), y'(x)) - \frac{\partial L}{\partial y}(x, y(x), y'(x)) = 0. \quad (3.26)$$

Damit ist der Beweis von Satz 3.17 vollständig. ○

Beispiel. Wir wenden die Euler-Lagrange-Gleichung auf die Lagrange-Funktion (3.21) an, dabei erhalten wir

$$\left. \begin{array}{l} \frac{\partial L}{\partial y} = -V'(y) \\ \frac{\partial L}{\partial \dot{y}} = m\ddot{y} \end{array} \right\} \Rightarrow \frac{d}{dt} \frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = \frac{d}{dt} m\ddot{y} + V'(y) = 0 \Rightarrow m\ddot{y} = -V'(y).$$

Dies ist das 2. Newtonsche Gesetz.

○

3.5.3 Lösung des Optimierungsproblems

Leider lässt sich der Satz 3.22 nicht direkt auf das Interpolationsproblem anwenden, weil im Ausdruck $J(g)$ die zweite Ableitung von g vorkommt. Wir führen daher die Rechnung, die auf die Euler-Lagrange-Differentialgleichung geführt hat, nochmals in diesem Spezialfall durch. Wieder sei h eine Funktion, die in jeder Stützstelle verschwindet. Die Minimalitätsbedingung ist dann

$$\begin{aligned} 0 &= \frac{d}{d\varepsilon} \int_{x_i}^{x_{i+1}} (g''(x) + \varepsilon h''(x))^2 dx \Big|_{\varepsilon=0} \\ &= \int_{x_i}^{x_{i+1}} 2g''(x)h''(x) + 2\varepsilon h''(x)^2 dx \Big|_{\varepsilon=0} \\ &= \int_{x_i}^{x_{i+1}} 2g''(x)h''(x) dx. \end{aligned} \quad (3.27)$$

Wie bei der Euler-Lagrange-Gleichung können wir durch partielle Integrieren die zweite Ableitung der Funktion h los werden:

$$\begin{aligned} 0 &= \left[g''(x)h'(x) \right]_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} g'''(x)h'(x) dx \\ &= \left[g''(x)h'(x) \right]_{x_i}^{x_{i+1}} - \left[g'''(x)h(x) \right]_{x_i}^{x_{i+1}} + \int_{x_i}^{x_{i+1}} g^{(4)}(x)h(x) dx. \end{aligned} \quad (3.28)$$

Auf Grund der Definition von h verschwindet der mittlere Term.

Bedingungen im Inneren der Teilintervalle

Jetzt nutzen wir wieder die freie Wahlmöglichkeit der Funktion h aus. Wir können die Funktion so wählen, dass $h(x_i) = h(x_{i+1}) = h'(x_i) = h'(x_{i+1}) = 0$ ist, dann verschwinden die ersten beiden Terme. Das Integral verschwindet nur dann immer, wenn der Integrand verschwindet, wenn also im Inneren jedes Teilintervalls $[x_i, x_{i+1}]$ gilt, dass $g^{(4)}(x) = 0$. Es folgt, dass in jedem Teilintervall die Funktion g ein kubisches Polynom sein muss.

Bedingungen an den Stützstellen

Aus dem verbleibenden ersten Term von Gleichung (3.28) lässt sich noch mehr über die zweiten Ableitungen der Funktion g schliessen. Die Summe dieser Terme muss ja ebenfalls 0 ergeben, also

$$\begin{aligned} 0 &= \sum_{i=0}^{n-1} \left[g''(x)h'(x) \right]_{x_i}^{x_{i+1}} = \sum_{i=0}^{n-1} (g''(x_{i+1}-)h'(x_{i+1}) - g''(x_i+)h'(x_i)) \\ &= -g''(x_0+)h'(x_0) + \sum_{i=1}^{n-1} h'(x_i)(g''(x_i-) - g''(x_i+)) + g''(x_n-)h'(x_n). \end{aligned}$$

Indem man für h eine Funktion wählt, die an allen Stützstellen verschwindet und in genau einer Stützstelle Ableitung 1 hat, was mit einem Hermite-Interpolationspolynom sicher möglich ist, schliesst man

$$g''(x_i-) = g''(x_i+) \quad \forall 1 \leq i < n. \quad (3.29)$$

Die Funktion ist also zweimal stetig differenzierbar. Schliesslich müssen auch die Terme an den Enden der Summe verschwinden. Eine Funktion h , die in allen Stützstellen zusammen mit der ersten Ableitung in den inneren Stützstellen verschwindet und deren erste Ableitung in genau einem der Endpunkte 1 ist zeigt, dass ausserdem

$$g''(x_0+) = g''(x_n-) = 0 \quad (3.30)$$

sein muss.

Ein Gleichungssystem für die Steigungen

Zur Lösung des eingangs gestellten Interpolationsproblems ist jetzt also für jedes Teilintervall $[x_i, x_{i+1}]$ ein kubisches Polynom $g_i(x)$ zu finden, mit folgenden Eigenschaften:

$$\begin{array}{llll} g_i(x_i) = f_i & g_i(x_{i+1}) = f_{i+1} & 0 \leq i \leq n & 2n + 2 \text{ Bedingungen} \\ g'_{i-1}(x_i) = g'_i(x_i) & g''_{i-1}(x_i) = g''_i(x_i) & 1 \leq i < n & 2n \text{ Bedingungen} \\ g''_0(x_0) = 0 & g''_n(x_n) = 0 & & 2 \text{ Bedingungen} \end{array}$$

Dies sind $4n + 4$ lineare Bedingungen für $n + 1$ Polynome, die je 4 Koeffizienten haben. Es sollte sich also ein lineares Gleichungssystem finden lassen, welches diese Koeffizienten findet.

Aus Abschnitt 3.3 ist bekannt, dass die kubischen Polynome $g_i(x)$ durch die bereits bekannten Funktionswerte f_i und die noch zu findenen Steigungen in den Stützstellen bestimmt sind. Wir schreiben daher $s_i = g'_i(x_i)$ für die Steigungen und machen es uns zum Ziel, ein Gleichungssystem für die s_i zu finden.

In Abschnitt 3.3.3 haben wir Hermite-Interpolationspolynome für zwei Stützstellen zusammengestellt. Wir haben dort die Polynome H_i und H_i^1 konstruiert, aus denen sich mit der Substitution

$x \rightarrow (x - x_0)/m$ die Hermite-Interpolationspolynome für das Intervall $[x_0, x_0 + m]$ bilden liess. Wir bezeichnen die Länge des Intervalls $[x_i, x_{i+1}]$ mit $m_i = x_{i+1} - x_i$.

Die gesuchte Funktion im Intervall ist daher

$$g_i(x) = f_i H_0((x - x_i)/m_i) + f_{i+1} H_1((x - x_i)/m_i) + s_i m_i H_0^1((x - x_i)/m_i) + s_{i+1} m_i H_1^1((x - x_i)/m_i). \quad (3.31)$$

Diese Funktion hat die richtigen Funktionswerte und Ableitungen an den Intervallenden.

Die Steigungen s_i in (3.31) ist noch nicht bekannt, aber die Bedingung an die zweiten Ableitungen wurde noch nicht ausgenutzt. Die zweiten Ableitungen erfüllen

$$\begin{aligned} i = 0 \quad g''_0(x_0) &= -\frac{6f_0}{m_0^2} + \frac{6f_1}{m_0^2} - \frac{4s_0}{m_0} + \frac{2s_1}{m_0} \\ i = 1 \quad g''_0(x_1) &= \frac{6f_0}{m_0^2} - \frac{6f_1}{m_0^2} + \frac{2s_0}{m_0} - \frac{4s_1}{m_0} \\ &= g''_1(x_1) = -\frac{6f_1}{m_1^2} + \frac{6f_2}{m_1^2} - \frac{4s_1}{m_1} + \frac{2s_2}{m_1} \\ i = 2 \quad g''_1(x_2) &= \frac{6f_1}{m_1^2} - \frac{6f_2}{m_1^2} + \frac{2s_1}{m_1} - \frac{4s_2}{m_1} \\ &= g''_2(x_2) = -\frac{6f_2}{m_2^2} + \frac{6f_3}{m_2^2} - \frac{4s_2}{m_2} + \frac{2s_3}{m_2} \\ &\vdots \\ i = n \quad g''_n(x_n) &= \frac{6f_{n-1}}{m_n^2} - \frac{6f_n}{m_n^2} + \frac{2s_{n-1}}{m_n} - \frac{4s_n}{m_n}. \end{aligned}$$

In allen Gleichungen kommt der Faktor 2 vor, den wir herausdividieren können. Schaffen wir die Terme in f_i auf die rechte Seite und sammeln die Terme mit s_i auf der linken Seite, erhalten wir das Gleichungssystem

$$\begin{aligned} \frac{2}{m_0}s_0 + \frac{1}{m_0}s_1 &= 3\frac{f_1 - f_0}{m_0^2} \\ \frac{1}{m_0}s_0 + \left(\frac{2}{m_0} + \frac{2}{m_1}\right)s_1 + \frac{1}{m_1}s_2 &= 3\frac{f_2 - f_1}{m_1^2} \\ \frac{1}{m_1}s_1 + \left(\frac{2}{m_1} + \frac{2}{m_2}\right)s_2 + \frac{1}{m_2}s_3 &= 3\frac{f_3 - f_2}{m_2^2} \\ &\vdots \\ \frac{1}{m_{n-2}}s_{n-1} + \frac{2}{m_{n-1}}s_n &= 3\frac{f_n - f_{n-1}}{m_{n-1}^2} \end{aligned} \quad (3.32)$$

Die Koeffizientenmatrix und die rechte Seite dieses Gleichungssystems sind

$$A = \begin{pmatrix} \frac{2}{m_0} & \frac{1}{m_0} & & \\ \frac{1}{m_0} & \frac{2}{m_0} + \frac{2}{m_1} & \frac{2}{m_1} & \\ & \frac{1}{m_1} & \frac{2}{m_1} + \frac{2}{m_2} & \frac{1}{m_2} \\ & & \frac{1}{m_2} & \ddots & \ddots \\ & & & \ddots & \ddots & \frac{1}{m_{n-2}} \\ & & & & \ddots & \frac{1}{m_{n-2}} \\ & & & & & \frac{1}{m_{n-1}} \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 3\frac{f_1 - f_0}{m_0^2} \\ 3\frac{f_2 - f_1}{m_1^2} \\ 3\frac{f_3 - f_2}{m_2^2} \\ \vdots \\ 3\frac{f_{n-1} - f_{n-2}}{m_{n-2}^2} \\ 3\frac{f_n - f_{n-1}}{m_{n-1}^2} \end{pmatrix}.$$

Die Gleichungen werden besonders einfach, wenn alle Abstände gleich sind, zum Beispiel $m = m_0 = \dots = m_{n-1}$. Dann kann man die Gleichungen mit m multiplizieren und bekommt für die Koeffizientenmatrix und die rechte Seite

$$A = \begin{pmatrix} 2 & 1 & & \\ 1 & 2 & 1 & \\ & 1 & 2 & 1 \\ & & 1 & \ddots & \ddots \\ & & & \ddots & \ddots & 1 \\ & & & & 1 & 2 \end{pmatrix} \quad \text{und} \quad b = \frac{3}{m} \begin{pmatrix} f_1 - f_0 \\ f_2 - f_1 \\ f_3 - f_2 \\ \vdots \\ f_{n-1} - f_{n-2} \\ f_n - f_{n-1} \end{pmatrix}$$

Nach Lösung dieses Gleichungssystems kann man in jedem Teilintervall $[x_i, x_{i+1}]$ mit Hilfe des kubischen Hermite-Interpolationspolynoms eine Spline-Interpolationsfunktion konstruieren.

3.5.4 Bézier-Kurven und Splines in der Ebene

Spline-Interpolation kann auch verwendet werden, um Kurven in der Ebene oder im Raum zu approximieren. In der Computergraphik ist dabei besonders wichtig, dass sich Kurvenpunkte mit einfachen Operationen aus einer kleinen Zahl von Parametern berechnen lassen, damit sie zum Beispiel von einem Graphikprozessor berechnet werden können. Dieser Abschnitt soll daher den Zusammenhang zwischen Bézier-Kurven und Splines aufzeigen.

Kurven in der Ebene

Eine Kurve in der Ebene ist eine Abbildung

$$\gamma: \mathbb{R} \rightarrow \mathbb{R}^2 : t \mapsto \gamma(t) = (x(t), y(t)),$$

genannt die Parameterdarstellung der Kurve. Man kann sich den Parameter t als die Zeit vorstellen und die Funktionen $x(t)$ bzw. $y(t)$ als die Koordinaten eines sich auf der Kurve bewegenden Punktes zur Zeit t . Der Tangentialvektor

$$\dot{\gamma}(t) = \frac{d\gamma(t)}{dt} = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix}$$

kann entsprechend auch als der Geschwindigkeitsvektor zur Zeit t interpretiert werden.

Beispiel. Ein Kreis in der Ebene kann beschrieben werden mit der Parameterisierung

$$\gamma(t) = (\cos t, \sin t) \quad \text{mit Geschwindigkeitsvektor} \quad \dot{\gamma}(t) = \begin{pmatrix} -\sin t \\ \cos t \end{pmatrix}.$$

Der Betrag der Geschwindigkeit ist $|\dot{\gamma}(t)|^2 = \sin^2 t + \cos^2 t = 1$, also konstant. \circ

Beispiel. Jeder Graph einer Funktion $f(x)$ kann als Kurve mit der Parameterisierung

$$\gamma: \mathbb{R} \rightarrow \mathbb{R}^2 : t \mapsto (t, f(t))$$

aufgefasst werden. Der Tangentialvektor ist

$$\dot{\gamma}(t) = \begin{pmatrix} 1 \\ f'(t) \end{pmatrix}.$$

Insbesondere ist die Geschwindigkeit $|\dot{\gamma}(t)|^2 = 1 + f'(x)^2 > 1$ im Allgemeinen nicht konstant. \circ

Das Beispiel zeigt, dass Graphen von Funktionen zwar als Kurven aufgefasst werden können, als Bahnbeschreibung zum Beispiel für einen Roboter taugen sie dagegen kaum. Andererseits haben die Spline-Interpolationsfunktionen die schöne Eigenschaft, dass die mittlere zweite Ableitung minimiert wird. Sie sind daher “die am wenigsten gekrümmten” Kurven, die durch die Stützstellen gehen. Eine solche Minimaleigenschaft für die Bahnkurve eines Roboters könnte eine Bahn beschreiben, die sich mit maximaler Geschwindigkeit durchfahren lässt. Die Zentripetalkraft, die die Räder in den Kurven aufbringen müssen, kann nicht grösser sein als die Haftriebung. Je grösser die Bahnkrümmung, desto grösser auch die Zentripetalkraft und desto langsamer muss der Roboter durch die Kurve fahren, um nicht ins Rutschen zu geraten.

Wir betrachten also eine Kurve, die durch die vorgegebene Punkte

$$P_0 = (x_0, y_0), P_1 = (x_1, y_1), P_2 = (x_2, y_2), \dots, P_n = (x_n, y_n)$$

gehen soll. Wir fordern, dass der Punkt P_i zum Zeitpunkt t_i durchlaufen wird. Wir entfernen uns hier etwas von der Anwendung einer Robotersteuerung, da würde man nur die Punkte vorgeben und dann eine Bahn suchen, mit der sich die Zeitpunkte t_i so wählen lassen, dass die Gesamtzeit minimal wird. Die Funktionen $x(t)$ und $y(t)$ erfüllen jetzt also

$$x(t_i) = x_i \quad \text{und} \quad y(t_i) = y_i.$$

Das Problem, eine ebene Kurve durch die Punkte P_i zu parametrisieren ist damit zerlegt worden in zwei unabhängige Interpolationsprobleme für die Funktionen $x(t)$ und $y(t)$.

Jedes in diesem Kapitel besprochene Interpolationsverfahren kann dafür eine mögliche Lösung liefern, wobei sich die Spline-Interpolation wegen der genannten Minimaleigenschaft besonders aufdrängt. Dabei müssen zuerst die Steigungen und den Knotenstellen t_i ermittelt werden, aus denen sich dann mittels Hermite-Interpolation die Kurvenstücke zwischen den Punkten P_i als kubische Kurven berechnen lassen. Die Steigungen in den Knotenstellen t_i sind die Ableitungen $\dot{x}(t_i)$ und $\dot{y}(t_i)$, d. h. es müssen zwischen P_i und P_{i+1} eine kubische Kurve in der Ebene berechnen, die Tangentialvektor $(\dot{x}(t_i), \dot{y}(t_i))^t$ bzw. $(\dot{x}(t_{i+1}), \dot{y}(t_{i+1}))^t$ haben. Eine besonders elegante Lösung für dieses Problem sind die Bézier-Kurven.

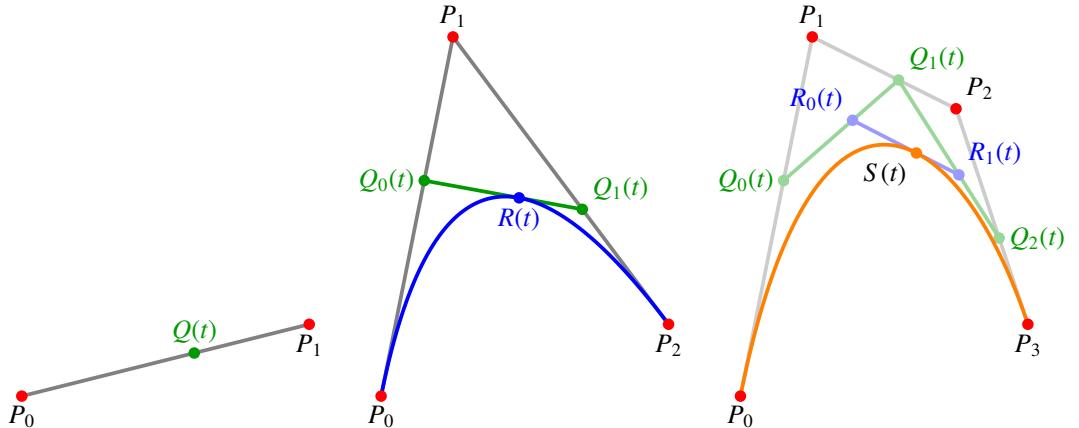


Abbildung 3.15: Bézier-Kurven bis zur Ordnung 3

Bézier-Kurven

Die einfachste Verbindung zwischen zwei Punkten \$P_0\$ und \$P_1\$ mit Ortsvektoren \$p_0\$ und \$p_1\$ ist eine Strecke, die man mit

$$\gamma(t) = (1-t)p_0 + tp_1$$

parametrisieren kann. Der Tangentialvektor ist bereits bestimmt, er ist

$$\dot{\gamma}(t) = -p_0 + p_1 = p_1 - p_0.$$

Die zwei Punkte legen den Tangentialvektor bereits fest, man kann ihn nicht mehr frei wählen.

Kann man eine einfach zu berechnende Kurve finden, die den Punkt \$P_0\$ mit einem vorgegebenen Geschwindigkeitsvektor verlässt? Paul de Casteljau hat vorgeschlagen, drei Punkte \$P_0\$, \$P_1\$ und \$P_2\$ zu verwenden und zunächst wie vorhin die Strecken

$$\begin{aligned} q_0(t) &= (1-t)p_0 + tp_1 \\ q_1(t) &= (1-t)p_1 + tp_2 \end{aligned}$$

zu bilden. Zur Zeit \$t = 0\$ verlässt das erste Segment den Punkt \$P_0\$ mit der Geschwindigkeit \$p_1 - p_0\$. Zur Zeit \$t = 1\$ kommt das zweite Segment im Punkt \$P_2\$ an mit der Geschwindigkeit \$p_2 - p_1\$. Man muss also zwischen \$t = 0\$ und \$t = 1\$ "vom ersten Segment auf das zweite wechseln". Dazu verbindet man die Punkte \$q_0(t)\$ und \$q_1(t)\$ mit einer Strecke und wählt den Streckenparameter wieder als \$t\$. Man erhält so eine Kurve

$$\begin{aligned} r(t) &= (1-t)q_0(t) + tq_1(t) \\ &= (1-t)((1-t)p_0 + tp_1) + t((1-t)p_1 + tp_2) \\ &= (1-t)^2 p_0 + 2t(1-t)p_1 + t^2 p_2. \end{aligned}$$

Der Geschwindigkeitsvektor zu den Zeiten \$t = 0\$ und \$t = 1\$ ist

$$\dot{r}(t) = -2(1-t)p_0 + (2(1-t) - 2t)p_1 + 2tp_2$$

$$= \begin{cases} -2p_0 + 2p_1 = 2(p_1 - p_0) & t = 0 \\ -2p_1 + 2p_2 = 2(p_2 - p_1) & t = 1 \end{cases}$$

Die gefundene Kurve verlässt also den Punkt P_0 genau in Richtung auf P_1 und kommt im Punkt P_2 aus der Richtung von P_1 an. Wir haben also eine quadratische Kurve gefunden, die einen Teil der Forderungen erfüllt.

Man nennt diese Kurve, dargestellt in Abbildung 3.15 Mitte, eine quadratische *Bézier-Kurve* mit *Kontrollpunkten* P_0 , P_1 und P_2 . Der Punkt P_1 kontrolliert die Start- und Ankunftsrichtung.

Kubische Bézier-Kurven

Die beiden Richtungen in Anfangs- und Endpunkten müssen unabhängig voneinander vorgegeben werden können, wir versuchen daher, die gleiche Konstruktion mit vier Kontrollpunkten durchzuführen. Gegeben seien jetzt also die Punkte P_0, \dots, P_3 . Dann können wir drei Strecken

$$q_0(t) = (1-t)p_0 + tp_1$$

$$q_1(t) = (1-t)p_1 + tp_2$$

$$q_2(t) = (1-t)p_2 + tp_3$$

konstruieren. Die erste hat die “richtige” Startrichtung, die letzte die “richtige” Ankunftsrichtung. Kombinieren wir $q_0(t)$ und $q_1(t)$, erhalten wir eine quadratische Kurve, die vom Punkt P_0 mit der richtigen Geschwindigkeit weggeht, die Kombination von $q_1(t)$ mit $q_2(t)$ liefert eine quadratische Bézier-Kurve, welche im Punkt P_3 mit der richtigen Geschwindigkeit ankommt. Wir können also erneut kombinieren:

$$\left. \begin{array}{l} r_0(t) = (1-t)q_0(t) + tq_1(t) \\ \quad = (1-t)^2 p_0 + 2t(1-t)p_1 + t^2 p_2 \\ r_1(t) = (1-t)q_1(t) + tq_2(t) \\ \quad = (1-t)^2 p_1 + 2t(1-t)p_2 + t^2 p_3 \end{array} \right\} \Rightarrow s(t) = (1-t)r_0(t) + tr_1(t) \quad (3.33)$$

(siehe auch Abbildung 3.15 rechts). Wir berechnen

$$\begin{aligned} s(t) &= (1-t)((1-t)^2 p_0 + 2t(1-t)p_1 + t^2 p_2) \\ &= (1-t)^3 p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t)p_2 + t^3 p_3. \end{aligned}$$

Der Tagentialvektor an den Stellen $t = 0$ und $t = 1$ ist

$$\begin{aligned} \dot{s}(t) &= -3(1-t)^2 p_0 + (3(1-t)^2 - 6t(1-t))p_1 + (6t(1-t) - 3t^2)p_2 + 3t^2 p_3 \\ \Rightarrow \dot{s}(0) &= -3p_0 + 3p_1 = 3(p_1 - p_0) \\ \dot{s}(1) &= -3p_2 + 3p_3 = 3(p_3 - p_2), \end{aligned}$$

die Kurve verlässt also wie erwartet den Punkt P_0 genau in Richtung auf P_1 und kommt genau aus der Richtung von P_2 in P_3 an.

Bézier-Kurven und Hermite-Interpolation

Man kann auch die Spline-Funktionen wieder aus der zweidimensionalen Kurve $s(t)$ rekonstruieren. Gegeben sind dafür zwei Werte y_0 und y_1 und die Steigungen m_0 und m_1 in den Punkten $x = 0$ und

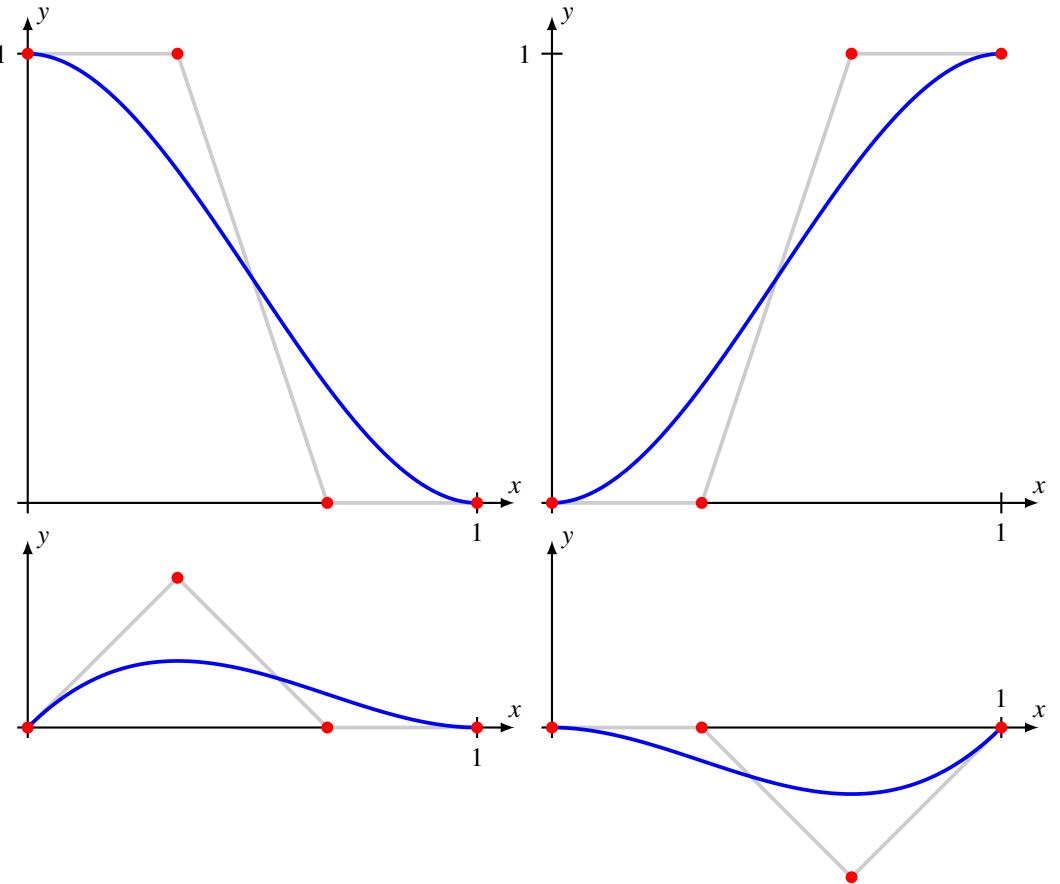


Abbildung 3.16: Die Polynome $H_k(x)$ und $H_k^l(x)$ für $k = 0, 1$ und $l = 0, 1$ berechnet als Bézier-Kurven der Ordnung 3.

$x = 1$. Der Graph des kubischen Polynoms $p(x)$ mit $p(0) = y_0$, $p(1) = y_1$, $p'(0) = m_0$ und $p'(1) = m_1$ soll jetzt als Bézier-Kurve geschrieben werden. Dazu müssen die Kontrollpunkte gefunden werden. Es ist klar, dass $P_0 = (0, y_0)$ und $P_3 = (1, y_1)$. Für die inneren Kontrollpunkte muss ein geeigneter x -Wert gewählt werden, also

$$P_1 = (x_1, y_0 + x_1 m_0) \quad \text{und} \quad P_2 = (x_2, y_1 - (1 - x_2)m_1)$$

Wählt man $x_1 = \frac{1}{3}$ und $x_2 = \frac{2}{3}$, dann wird

$$\begin{aligned} x(t) &= (1-t)^3 \cdot 0 + 3t(1-t)^2 \cdot \frac{1}{3} + 3t^2(1-t) \cdot \frac{2}{3} + t^3 \cdot 1 \\ &= t - 2t^2 + t^3 + 2t^2 - 2t^3 + t^3 = t. \end{aligned}$$

Mit dieser Wahl ist also $x = t$, so dass das gesuchte Polynom $p(x) = y(x)$ wird:

$$p(x) = y(x) = (1-x)^3 y_0 + 3x(1-x)^2 \left(y_0 + \frac{1}{3}m_0 \right) + 3x^2(1-x) \left(y_1 - \frac{1}{3}m_1 \right) + x^3 y_1$$

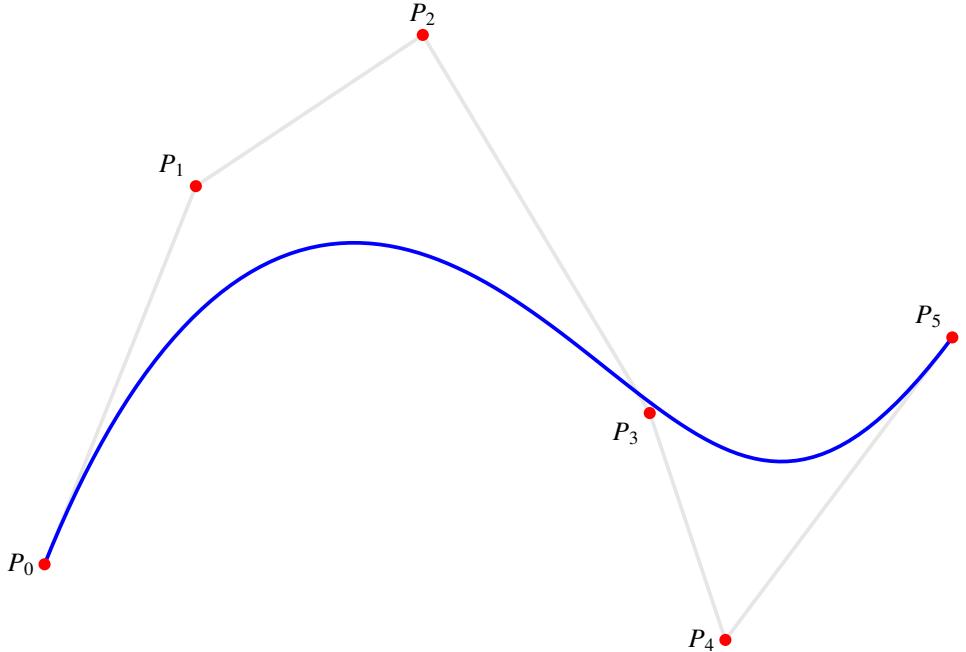


Abbildung 3.17: Bézier-Kurve vom Grad 5 mit 6 Kontrollpunkten.

$$\begin{aligned}
 &= ((1-x)^3 + 3x(1-x)^2)y_0 + x(1-x)^2m_0 - x^2(1-x)m_1 + (3x^2(1-x) + x^3)y_1 \\
 &= (1+2x)(1-x)^2y_0 + x(1-x)^2m_0 + x^2(x-1)m_1 + x^2(3-2x)y_1.
 \end{aligned}$$

Die Koeffizienten von y_0 , y_1 , m_0 und m_1 sind die Polynome

$$\begin{aligned}
 y_0 : \quad H_0(x) &= 2x^3 - 3x^2 + 1 & m_0 : \quad H_0^1(x) &= x - 2x^2 + x^3 \\
 y_1 : \quad H_1(x) &= 3x^2 - 2x^3 & m_1 : \quad H_1^1(x) &= x^3 - x^2
 \end{aligned} \tag{3.34}$$

Die Polynome (3.34) stimmen mit den Polynomen überein, die in (3.17) gefunden worden sind.

Bézier-Kurven höherer Ordnung

Der Prozess, der die kubischen Bézier-Kurven produziert hat, kann natürlich iteriert werden, um Kurven beliebiger Ordnung zu erzeugen. Ausgehend von $n+1$ Kontrollpunkten P_0, \dots, P_n , konstruiert man eine rekursiv eine Folge von Punkten wie folgt. Zunächst setzt man $P_{0k}(t) = P_k$ mit zugehörigen Ortsvektoren $p_{0k}(t)$. Dann definiert man die *konvexe Kombination* der beiden Kurven als

$$p_{ik}(t) = (1-t)p_{i-1,k}(t) + p_{i-1,k+1}(t) \quad 1 \leq i \leq n, \quad 0 \leq k \leq n-i.$$

Der Vektor $p_{n0}(t)$ beschreibt eine Kurve vom Grad $n-1$. In Abbildung 3.17 ist eine Bézier-Kurve der Ordnung 5 mit 6 Kontrollpunkten dargestellt.

Der rekursive Berechnungsprozess von $p_{n0}(t)$ ist in Abbildung 3.18 schematisch dargestellt. Jeder schräge Pfeil steht für zwei Multiplikationen, für jeden Summenknoten sind also vier Multiplikationen und zwei Additionen auszuführen. Daraus kann man ablesen, dass die Berechnung von $s(t)$

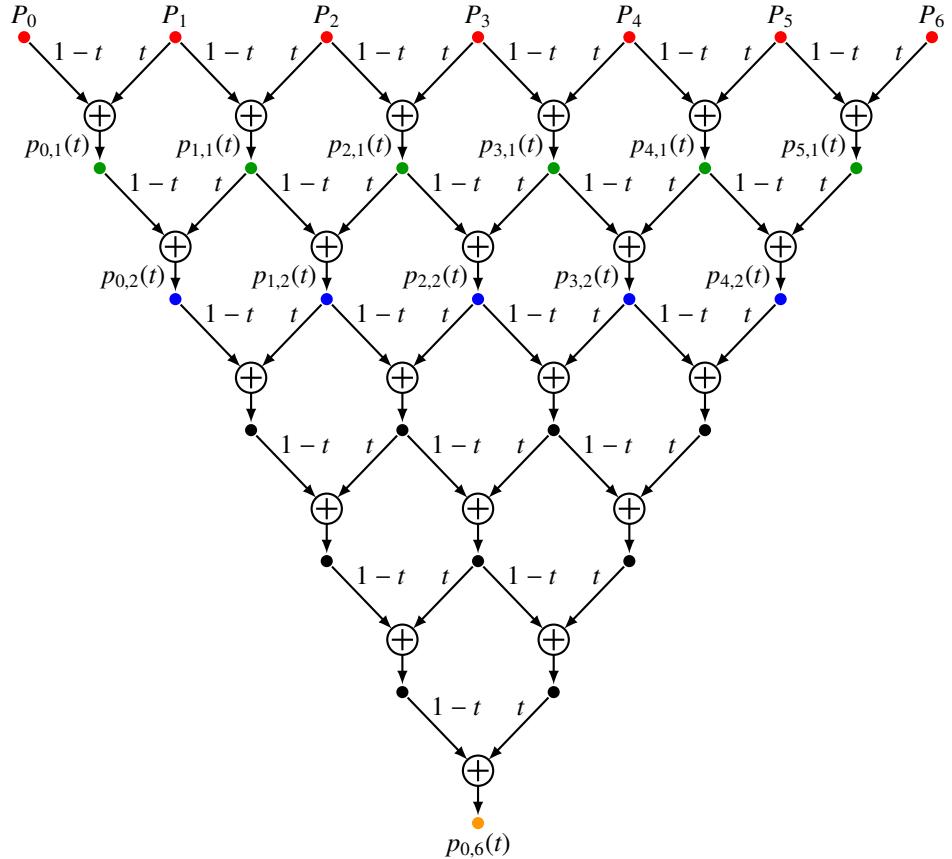


Abbildung 3.18: Berechnungsschema für die Koordinaten eines Punktes einer Bézier-Kurve mit 7 Kontrollpunkten. Jeder schräge Pfeil entspricht zwei Multiplikationen. Die Anzahl Multiplikationen ist viermal so gross wie die Zahl der Summationen.

genau

$$4 \cdot (1 + 2 + \dots + n) = 4 \sum_{k=1}^n k = 2n(n+1) \quad (3.35)$$

Multiplikationen und $n(n+1)$ Additionen benötigt.

Satz 3.19. Die Kurve $p_{n0}(t)$ hat die Eigenschaft

$$\begin{aligned} p_{n0}(0) &= n(p_1 - p_0) \\ \dot{p}_{n0}(1) &= n(p_n - p_{n-1}). \end{aligned}$$

In expliziter Form gilt

$$p_{n0}(t) = \sum_{k=0}^n \binom{n}{k} t^k (1-t)^{n-k} p_k = \sum_{k=0}^n B_{k,n}(t) p_k$$

wobei $B_{k,n}(t) = \binom{n}{k} t^k (1-t)^{n-k}$ die sogenannten Bernstein-Polynome sind.

Man braucht immer die Werte aller Bernsteinpolynome. Dazu berechnet man in $2(n - 1)$ Multiplikationen zuerst alle Potenzen von t und $(1 - t)$. Mit $2(n - 1)$ weiteren Multiplikationen bekommt man dann die Werte der Bernsteinpolynome. Insgesamt sind also $4(n - 1)$ Multiplikationen für die Bernstein-Polynome nötig. Die Berechnung der Bézier-Kurve benötigt nochmals $2(n + 1)$ Multiplikationen und $2n$ Additionen. Insgesamt sind also $6n - 2$ Multiplikationen und $2n$ Addition nötig. Die Berechnung nach dem Schema von Abbildung 3.18 wächst dagegen quadratisch mit n an. Für $n \geq 2$ folgt aus (3.35) dass das rekursive Verfahren mindestens $2n \cdot 3 = 6n$ und damit bereits mehr Multiplikationen benötigen.

Beweis. Die Terme mit p_k im Polynom $p_{0n}(t)$ kann man aus Abbildung 3.18 ablesen. Jeder Weg von p_k zum Polynom $p_{0n}(t)$ setzt sich zusammen aus einer k Links-Pfeilen und $n - k$ Rechts-Pfeilen. Jeder Weg trägt daher einen Summanden $t^k(1 - t)^{n-k}$ bei. Wie beim Pascal-Dreieck kann man einsehen, dass es $\binom{n}{k}$ solche Wege gibt. Damit ist die Formel für die Koeffizienten beweisen. Durch Ableiten könnte man auch die Behauptung über den Tangentialvektor am Anfang und am Ende der Kurve beweisen. \square

Die Eigenschaften des Tangentialvektors bei $t = 0$ und $t = 1$ der Kurve $p_{0n}(t)$ lässt sich aber mit Hilfe der Rekursionsformel auch direkt nachweisen.

Lemma 3.20. *Sind $p(t)$ und $q(t)$ zwei beliebige Kurven derart, dass $\dot{p}(0) = \alpha(q(0) - p(0))$ und $\dot{q}(1)\beta(q(1) - p(1))$, dann ist $\gamma(t) = (1 - t)p(t) + tq(t)$ eine Kurve mit $\dot{\gamma}(0) = (\alpha + 1)(q(0) - p(0))$ und $\dot{\gamma}(1) = (\beta + 1)(q(1) - p(1))$.*

Beweis. Wir berechnen die Ableitung von $\gamma(t)$

$$\begin{aligned}\dot{\gamma}(t) &= -p(t) + (1 - t)\dot{p}(t) + q(t) + t\dot{q}(t) \\ \dot{\gamma}(0) &= -p(0) + q(0) + \dot{p}(0) = (1 + \alpha)(q(0) - p(0)), \\ \dot{\gamma}(1) &= -p(1) + q(1) + \dot{q}(1) = (1 + \beta)(q(1) - p(1)).\end{aligned}$$

\square

Wir können jetzt den Beweis der Eigenschaften des Tangentialvektors mit Hilfe von vollständiger Induktion abschliessen. Die Polynome $p_{k0}(t)$ haben die im Lemma formulierte Eigenschaft, denn es ist $\dot{p}_{k0}(t) = 0$ und daher

$$\begin{aligned}\dot{p}_{k0}(0) &= 0 \cdot p_{k+1,0}(0) - p_{k,0}(0) = 0 \cdot (p_{k+1} - p_k) \\ \dot{p}_{k0}(1) &= 0 \cdot p_{k+1,0}(1) - p_{k,0}(1) = 0 \cdot (p_{k+1} - p_k).\end{aligned}$$

Nehmen wir jetzt an, die Eigenschaft sei für Bézier-Kurven vom Grad l bereits bewiesen. Die Kurve $p_{kl}(t)$ ist eine Bézier-Kurve mit den Kontrollpunkten P_k, \dots, P_{k+l} . Dies bedeutet, dass die Kurven $p_{kl}(t)$ den Tangentialvektor

$$\begin{aligned}\dot{p}_{kl}(0) &= l(p_k - p_{k+1}) \\ \dot{p}_{kl}(1) &= l(p_{k+l} - p_{k+l-1})\end{aligned}$$

haben. Ausserdem ist $p_{kl}(0) = p_k$ und $p_{kl}(1) = p_{k+l}$, so dass auch die Bedingung über die Endpunkte im Lemma erfüllt ist. Es folgt daher mit Hilfe des Lemmas für die konvexe Kombination $p_{k,l+1}(t)$ von $p_{kl}(t)$ und $p_{k+1,l}(t)$

$$\begin{aligned}\dot{p}_{k,l+1}(0) &= (1 + l)(p_{k,l+1}(0) - p_{k,l}(0)) = (l + 1)(p_{k+1} - p_k) \\ \dot{p}_{k,l+1}(1) &= (1 + l)(p_{k,l+1}(1) - p_{k,l}(1)) = (l + 1)(p_{k+l+1} - p_{k+l}).\end{aligned}$$

Damit ist der Induktionsschritt vollzogen und alles bewiesen.

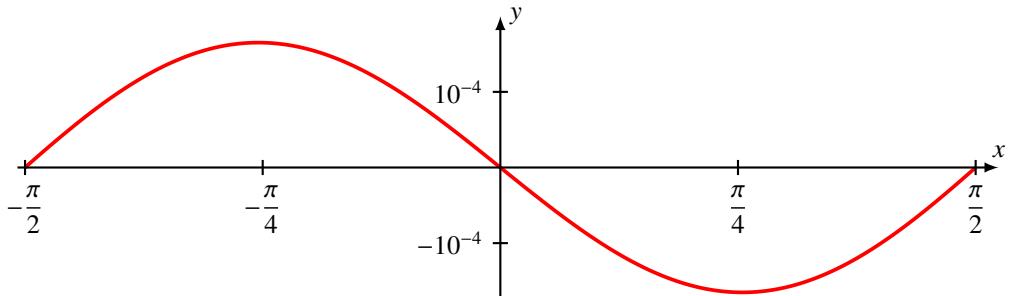


Abbildung 3.19: Fehler des Interpolationspolynoms für $\sin x$ mit Stützstellen $\frac{\pi}{2}k$ mit ganzzahligen k mit $-10 \leq k \leq 10$ (Übung 3.1).

Übungsaufgaben

3.1. Das Polynom $p(x)$ soll die Funktionswerte der Sinusfunktion an den Stellen $k\frac{\pi}{2}$ für ganzzahliges k mit $-10 \leq k \leq 10$ interpolieren. Wie gross ist der Fehler des Interpolationspolynoms für $x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

Lösung. Zunächst halten wir fest, dass 21 Stützstellen verwendet werden, dass also $n = 20$ ist. Nach der Fehlerformel für das Interpolationspolynom gilt

$$|f(x) - p(x)| \leq \frac{|l(x)|}{21!} |f^{(21)}(x)|$$

mit $f(x) = \sin x$. Die Ableitungen von f sind wieder trigonometrische Funktionen, d. h. $|f^{(21)}(x)| \leq 1$. Der Fehler ist daher

$$|f(x) - p(x)| \leq \frac{|l(x)|}{21!}.$$

Um $|l(x)|$ abzuschätzen verwendet man

$$\begin{aligned} |l(x)| &= |x - x_0| \cdot |x - x_1| \cdots |x - x_n| \\ &\leq (11 \frac{\pi}{2} \cdot 10 \frac{\pi}{2} \cdots 2 \frac{\pi}{2})^2 \frac{\pi}{2} \\ &= \left(\frac{\pi}{2}\right)^{21} (11!)^2 \end{aligned}$$

Damit kann man jetzt den Fehler abschätzen:

$$|\sin x - p(x)| \leq \left(\frac{\pi}{2}\right)^{21} \frac{(11!)^2}{21!} = 22 \cdot \left(\frac{\pi}{2}\right)^{21} \left(\frac{22}{11}\right)^{-1} = 0.40972.$$

In der Tat ist der Fehler viel kleiner als diese Schranke vermuten lässt. In Abbildung 3.20 kann man erkennen, dass das Interpolationspolynom die Funktion in der Mitte des Intervalls sehr genau wiedergeben kann. Nur am Rande weicht es wegen des Runge's Phänomens stark ab. In Abbildung 3.19 ist der Fehler $\sin x - p(x)$ für $x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ dargestellt, es zeigt sich, dass der Betrag des Fehlers kleiner als $1.66 \cdot 10^{-4}$ ist. ○

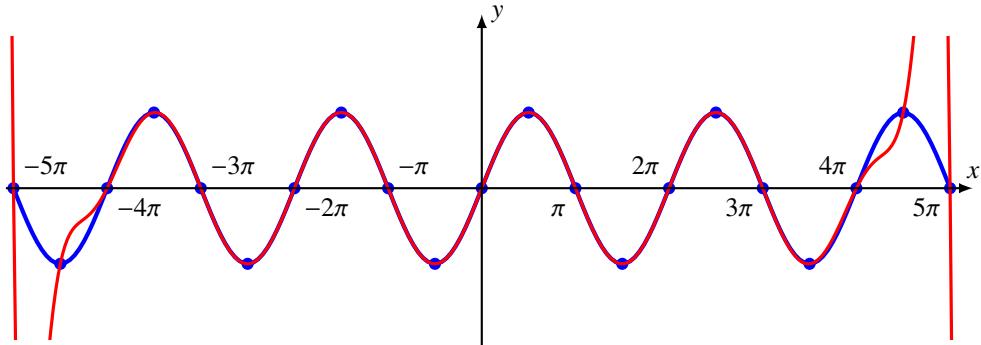
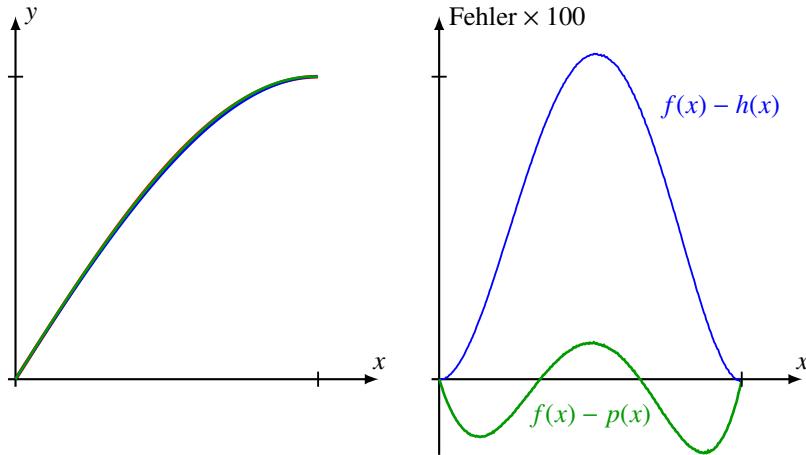


Abbildung 3.20: Graph des Interpolationspolynoms für $f(x) = \sin x$ mit Stützstellen $\frac{\pi}{2}k$ für ganzzahlige k mit $-10 \leq k \leq 10$. Trotz des grossen Abstandes und der sehr speziellen Wahl der Stützstellen folgt das Interpolationspolynom (rot) der Funktion (blau) in der Mitte des Definitionsbereichs sehr genau.

3.2. Wir möchten das Lagrange- und das Hermite-Interpolationspolynom für die Funktion $f(x) = \sin \frac{\pi}{2}x$ im Intervall $[0, 1]$ vergleichen. Die Abbildung zeigt, dass die Polynome sehr nahe beieinander liegen.



- Finden Sie das Hermite-Interpolationspolynom $h(x)$ von $f(x)$ für die Stützstellen 0 und 1.
- Finden Sie das Lagrange-Interpolationspolynom $p(x)$ von $f(x)$ für die Stützstellen $0, \frac{1}{3}, \frac{2}{3}$ und 1.
- Berechnen Sie die Funktionswerte $f(\frac{1}{2})$, $p(\frac{1}{2})$ und $h(\frac{1}{2})$.

Lösung. a) Das Hermite-Interpolationspolynom kann aus den Funktions- und Ableitungswerten an den Intervallenden berechnet werden:

$$\begin{aligned} f(0) &= 0 & f(1) &= 1 \\ f'(0) &= \frac{\pi}{2} & f'(1) &= 0 \end{aligned} \tag{3.36}$$

Daraus kann man das Interpolationspolynom ablesen:

$$\begin{aligned} h(x) &= \frac{\pi}{2} H_0^1(x) + H_1(x) = \frac{\pi}{2}(x^3 - 2x^2 + x) - 2x^3 + 3x^2 \\ &= \left(\frac{\pi}{2} - 2\right)x^3 + (3 - \pi)x^2 + \frac{\pi}{2}x \end{aligned}$$

b) Für das Lagrange-Interpolationspolynom geht man aus vom Polynom

$$l(x) = x(x - \frac{1}{3})(x - \frac{2}{3})(x - 1)$$

und konstruiert daraus die Polynome

$$\begin{aligned} l_0(x) &= \frac{(x - \frac{1}{3})(x - \frac{2}{3})(x - 1)}{-\frac{1}{3}(-\frac{2}{3})(-1)} &= -\frac{9}{2}x^3 + 9x^2 - \frac{11}{2}x + 1 \\ l_1(x) &= \frac{x(x - \frac{2}{3})(x - 1)}{\frac{1}{3}(-\frac{1}{3})(-\frac{2}{3})} &= \frac{27}{2}x^3 - \frac{45}{2}x^2 + 9x \\ l_2(x) &= -\frac{x(x - \frac{1}{3})(x - 1)}{\frac{2}{3}\frac{1}{3}(-\frac{1}{3})} &= -\frac{27}{2}x^3 + 18x^2 - \frac{9}{2}x \\ l_3(x) &= \frac{x(x - \frac{1}{3})(x - \frac{2}{3})}{\frac{2}{3}\frac{1}{3}} &= \frac{9}{2}x^3 - \frac{9}{2}x^2 + x \end{aligned}$$

Das Lagrange-Interpolationspolynom ist dann

$$p(x) = \sum_{k=0}^3 f_k l_k(x) = \sin \frac{\pi}{6} l_1(x) + \sin \frac{\pi}{3} l_2(x) + l_3(x) = \frac{1}{2} l_1(x) + \frac{\sqrt{3}}{2} l_2(x) + l_3(x)$$

c) Die Funktionswerte in $x = \frac{1}{2}$ sind

$$f(\frac{1}{2}) = \frac{\sqrt{2}}{2} = 0.70710678$$

$$p(\frac{1}{2}) = 0.69843726$$

$$h(\frac{1}{2}) = 0.69634954$$

$$f(\frac{1}{2}) - p(\frac{1}{2}) = 0.00866952$$

$$f(\frac{1}{2}) - h(\frac{1}{2}) = 0.01075724$$

Diese Werte und die Abbildung oben zeigen, dass das Lagrange-Interpolationspolynome etwas genauer ist. Interessant am Hermite-Interpolationspolynome ist hingegen, dass der Fehler im ganzen Intervall > 0 ist, was man mit Hilfe der Fehlerformel auch direkt nachweisen kann.

○

3.3. Beschreiben Sie den Viertel des Einheitskreises im ersten Quadranten als eine Bézier-Kurve mit Kontrollpunkten

$$(1, 0), \quad (1, a), \quad (a, 1) \quad \text{und} \quad (0, 1).$$

Wählen Sie a so, dass die Bézier-Kurve durch den Punkt $(1/\sqrt{2}, 1/\sqrt{2})$ verläuft.

a) Bestimmen Sie die maximale Abweichung der Bézier-Kurve vom Einheitskreis.

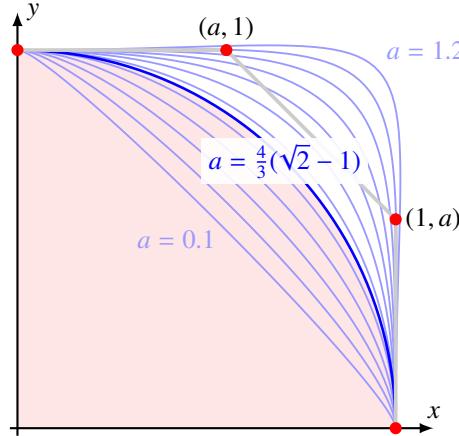


Abbildung 3.21: Approximation eines Kreises (hellrote Fläche) mit Hilfe einer Bézier-Kurve durch die Punkte $(1,0)$, $(1,a)$, $(a,1)$ und $(0,1)$ (rote Punkte). Für $a = \frac{4}{3}(\sqrt{2}-1)$ verläuft die Bézier-Kurve durch den Punkt $(\sqrt{2}/2, \sqrt{2}/2)$ auf dem Einheitskreis.

- b) Vergleichen Sie die Approximation durch die Bézier-Kurve mit der Parametrisierung $t \mapsto (\cos \frac{\pi}{2}t, \sin \frac{\pi}{2}t)$ des Einheitskreises.

Lösung. Die Bézier-Kurve hat die Koordinaten

$$\begin{aligned}x(t) &= (1-t)^3 \cdot 1 + 3(1-t)^2 t \cdot 1 + 3(1-t)t^2 \cdot a + t^3 \cdot 0, \\y(t) &= (1-t)^3 \cdot 0 + 3(1-t)^2 t \cdot a + 3(1-t)t^2 \cdot 1 + t^3 \cdot 1.\end{aligned}$$

Aus Symmetriegründen schneidet die Bézier-Kurve die 45° -Gerade im Parameter-Wert $t = \frac{1}{2}$, also

$$x\left(\frac{1}{2}\right) = \frac{1}{8}(1+3+3a+0) = \frac{3a+4}{8} \quad y\left(\frac{1}{2}\right) = \frac{1}{8}(0+3a+3+1).$$

Daraus ergibt sich

$$3a+4 = 4\sqrt{2} \quad \Rightarrow \quad a = \frac{4}{3}(\sqrt{2}-1) \approx 0.5522847.$$

Damit kann man jetzt die Fehlerbetrachtungen der beiden Teilaufgaben durchführen:

- a) Abbildung 3.22 zeigt den Fehler des Radius $|\vec{b}(t)| - 1$. Die Bézier-Kurve verläuft immer ausserhalb des Kreises. In Abbildung 3.22 rechts ist der Radiusfehler 500-fach überhöht dargestellt.
- b) Der Geschwindigkeitsvektor der Bézier-Kurve zur Zeit $t = 0$ ist nach Satz 3.19 das dreifache des Vektors zwischen den ersten zwei Punkten, also

$$\dot{\vec{b}}(0) = \begin{pmatrix} \dot{x}(0) \\ \dot{y}(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 3a \end{pmatrix} = \begin{pmatrix} 0 \\ 4(\sqrt{2}-1) \end{pmatrix} \approx \begin{pmatrix} 0 \\ 1.65685 \end{pmatrix}.$$

Der Geschwindigkeitsvektor der Kreisparametrisierung $\vec{k}(t)$ ist dagegen

$$\dot{\vec{k}}(0) = \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix} \approx \begin{pmatrix} 0 \\ 1.57079 \end{pmatrix}.$$

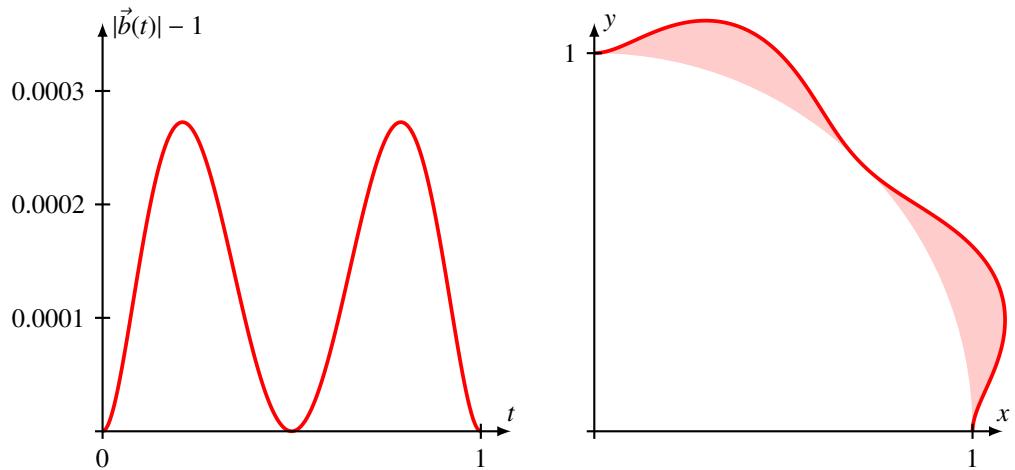


Abbildung 3.22: Radiusfehler $|\vec{b}(t)| - 1$ der Bézier-Approximation $\vec{b}(t)$ des Einheitskreises. Der Fehler des Radius (links) ist sehr gering, rechts ist die Abweichung vom Kreis 500-fach überhöht dargestellt.

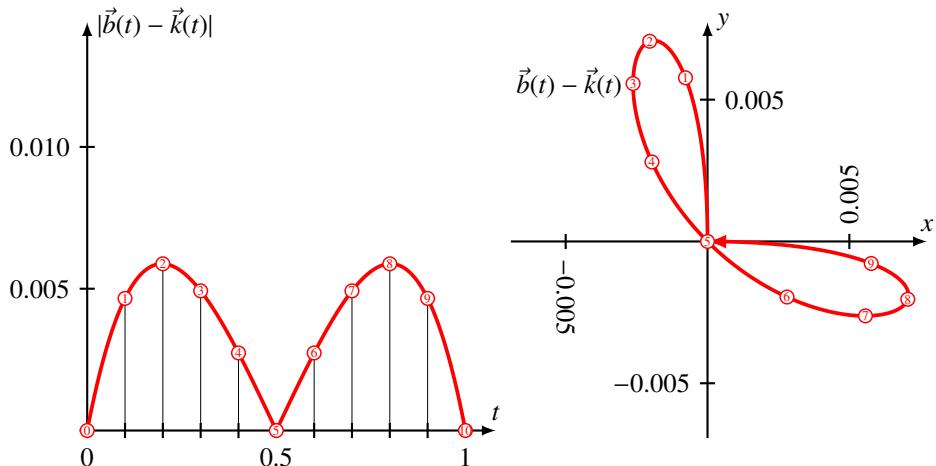


Abbildung 3.23: Fehler der Bézier-Approximation $\vec{b}(t)$ des Einheitskreises, der üblicherweise mit $\vec{k}(t) = (\cos t \frac{\pi}{2}, \sin t \frac{\pi}{2})$ parametrisiert wird. Die Abweichung $\vec{b}(t) - \vec{k}(t)$ ist etwa 30 mal grösser als $|\vec{b}(t)| - 1$, aber immer noch sehr klein. Rechts ist der Differenz-Vektor $\vec{b}(t) - \vec{k}(t)$ dargestellt.

Der Punkt auf der Bézier-Kurve entfernt sich daher zu Beginn mit der Geschwindigkeit

$$\vec{v}(0) = \dot{\vec{b}}(0) - \dot{\vec{k}}(0) \approx \begin{pmatrix} 0 \\ 0.0860579 \end{pmatrix}$$

vom Punkt $\vec{k}(t)$. Die graphische Darstellung in Abbildung 3.23 zeigt, dass ungefähr zur Zeit $t = 0.2$ die grösste Entfernung eintritt. Der Unterschied kann also nicht grösser sein als $0.2 \cdot 0.0860579 \approx 0.017$, was immer noch erstaunlich gut ist. \circ

4

Integration

Die Wahrscheinlichkeitsdichte der Standardnormalverteilung führt auf das Integral

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt$$

für die Verteilungsfunktion. Man kann beweisen, dass es für diesen Integranden keine Stammfunktion in analytischer Form als algebraischer Ausdruck bekannter Funktionen geben kann. Es bleibt also nur die numerische Berechnung. Dieses Kapitel stellt einige Methoden zur Berechnung von Integralen zusammen.

4.1 Riemann-Integral und Trapezregel

Die Stammfunktion

$$F(x) = \int_a^x f(t) dt$$

einer Funktion $f(x)$ ist die Lösung der besonders einfachen Differentialgleichung $y' = f(x)$, man könnte also dieses Kapitel einfach überspringen und für die Berechnung von Integralen auf die Methoden zur Lösung gewöhnlicher Differentialgleichungen verweisen, die in Kapitel 5 besprochen werden. Dies ist aber nicht unbedingt sinnvoll. Das bestimmte Integral einer Funktion zwischen zwei Grenzen ist ein wesentlich einfacheres Konzept als die Lösung einer Differentialgleichung, man darf daher davon ausgehen, dass es auch einfachere Berechnungsverfahren dafür geben dürfte. Die relativ komplizierten Lösungsverfahren für gewöhnliche Differentialgleichungen dürften viel zu viel rechnen für das gestellte Problem.

Wir erwarten daher, dass spezialisierte Verfahren zur Berechnung von Integralen folgende Eigenschaften haben:

1. Einfache Anwendung: Der Code zur Berechnung eines Integrals sollte sehr viel einfacher ausfallen als der Code zur Lösung einer Differentialgleichung.
2. Allgemein anwendbar: Das Verfahren sollte für eine grosse Klasse von Integranden anwendbar sein, auch für solche, für die Lösungsverfahren für gewöhnliche Differentialgleichungen schwierig zu konstruieren sind. Zum Beispiel verlangen die Eindeutigkeitssätze für gewöhnliche Differentialgleichungen typischerweise eine Lipschitz-Eigenschaft. Integrale sollten aber auch von Funktionen berechnet werden können, die eine solche Eigenschaft nicht haben.

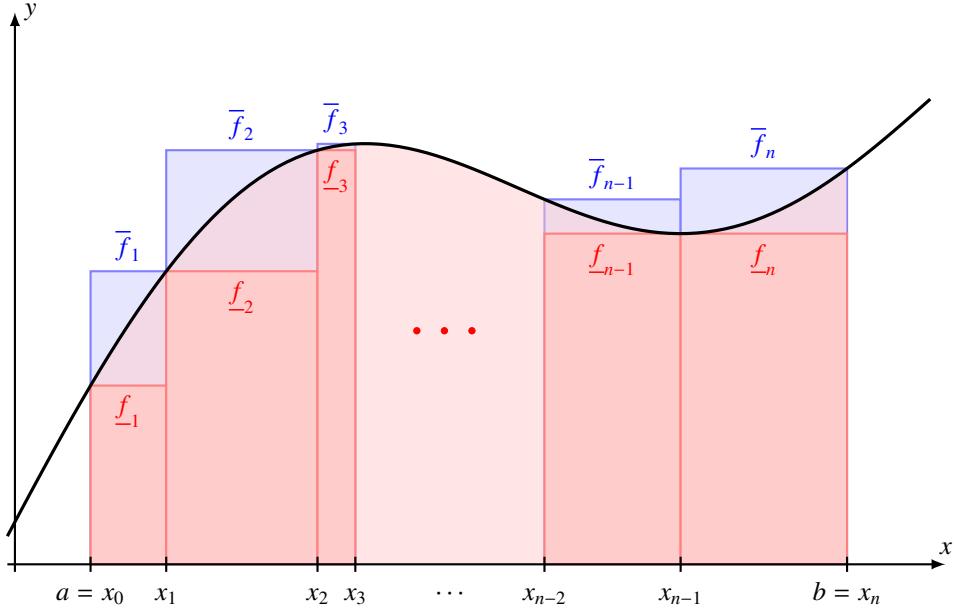


Abbildung 4.1: Definition der oberen und unteren Riemann-Summe und des Riemann-Integrals

3. Schnelle Konvergenz für “gute” Integranden. Glatte Integranden sollten nur an wenigen Stellen ausgewertet werden müssen und bereits gute Resultate ergeben.

Der besser geeignete Ausgangspunkt für die Konstruktion eines Integrationsverfahrens ist daher die ursprüngliche Definition des Riemann-Integrals, welche wir in Abschnitt 4.1.1 rekapitulieren. Daraus ergibt sich dann die Mittelpunktsformel in Abschnitt 4.1.2, deren Fehlerverhalten in Abschnitt 4.1.4 untersucht wird.

4.1.1 Das Riemann-Integral

Im Analysisunterricht wird das Riemann-Integral

$$I = \int_a^b f(x) dx \quad (4.1)$$

einer Funktion $f(x)$ zwischen den Grenzen a und b üblicherweise wie folgt definiert. Zunächst wird das Intervall $[a, b]$ mit Hilfe einer Menge $D = \{x_0, x_1, x_2, \dots, x_{n-1}, x_n\}$ von Zwischenpunkten mit der Eigenschaft

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

unterteilt (siehe Abbildung 4.1). Das *Korn* der Unterteilung ist die Länge des längsten Teilintervalls

$$\delta(D) = \max_{0 \leq i < n} |x_{i+1} - x_i|.$$

Als Approximationen des Integrals (4.1) werden dann die obere und untere Riemann-Summe

$$\bar{I}(D) = \sum_{i=1}^n \bar{f}_i(x_i - x_{i-1}) \quad \text{mit} \quad \bar{f}_i = \max_{\xi \in [x_{i-1}, x_i]} f(\xi)$$

$$\underline{I}(D) = \sum_{i=1}^n \underline{f}_i (x_i - x_{i-1}) \quad \text{mit} \quad \underline{f}_i = \min_{\xi \in [x_{i-1}, x_i]} f(\xi)$$

gebildet. Die untere Riemann-Summe $\underline{I}(D)$ ist in Abbildung 4.1 rot dargestellt, die obere Riemann-Summe $\bar{I}(D)$ blau. Aufgrund der Konstruktion ist klar, dass $\underline{I}(D) \leq I \leq \bar{I}(D)$ sein muss.

Für einigermassen “glatte” Funktionen wird der Unterschied zwischen $\bar{I}(D)$ und $\underline{I}(D)$ kleiner werden, wenn man die Unterteilung verfeinert. Falls die obere und die untere Riemann-Summe bei Verfeinerung gegen den gleichen Grenzwert streben, sagt man, das *Riemann-Integral* existiere und habe den Wert

$$I = \int_a^b = \lim_{\delta(D) \rightarrow 0} \underline{I}(D) = \lim_{\delta(D) \rightarrow 0} \bar{I}(D).$$

Wenn das Integral existiert, kann man offenbar auch jeden beliebigen anderen anderen Wert der Funktion in den Teilintervallen $[x_{i-1}, x_i]$ verwenden. Eine Summe der Form

$$\sum_{i=1}^n f(\xi_i)(x_i - x_{i-1}) \quad \text{mit} \quad \xi_i \in [x_{i-1}, x_i],$$

auch *Riemann-Summe* genannt, ist für jede beliebige Wahl der Unterteilung D und der Auswertepunkte ξ_i ein Approximation des Integrals I von (4.1).

Die Riemann-Summe liefert also bereits eine direkte Berechnungsmöglichkeit für ein beliebiges Integral und sie liefert uns auch ein Möglichkeit, die Größenordnung des zu erwartenden Fehlers abzuschätzen: Die Differenz $\bar{I}(D) - \underline{I}(D)$ ist der grösste mögliche Fehler. Der Berechnungsaufwand lässt sich ebenfalls sehr gut abschätzen. Es muss eine Summe von n Termen gebildet werden, in jeder Summe wird eine Differenz aufeinanderfolgender Teilpunkte gebildet und ein Produkt mit dem Funktionswert. Es wird offensichtlich, dass ausser für sehr einfache Funktionen, für die man das Integral wohl auch analytisch ausrechnen könnte, der Hauptteil der Arbeit in der Berechnung der Funktionswert $f(\xi_i)$ steckt.

Die Riemann-Summe beinhaltet einige Wahlmöglichkeiten, mit der die Berechnung optimiert werden kann. Wir können die Teilpunkte D wählen und zum Beispiel die Unterteilung dort feiner wählen, wo die Funktion sich schnell verändert. Wir können auch die Auswertepunkte ξ_i wählen. Für beides ist jedoch eine detaillierte Kenntnis der Funktion notwendig, welche nur durch die Berechnung zusätzlicher Funktionswerte gewonnen werden kann.

4.1.2 Mittelpunktsregel

Die einfachste Art der Unterteilung des Intervalls ist Teilintervall konstanter Länge zu verwenden. Wir schreiben

$$h = \frac{b-a}{n} \quad \text{und} \quad x_i = a + ih$$

für die Intervalllänge und die Teilpunkte. Werten wir die Funktion im Mittelpunkt eines Teilintervalls aus, also in $\xi = (x_{i-1} + x_i)/2 = x_{i-1} + \frac{1}{2}h = x_i - \frac{1}{2}h$, erhalten wir als Approximation für das Integral (4.1) die *Mittelpunktsregel*

$$M(h) = \sum_{i=1}^n f\left(x_i - \frac{h}{2}\right) \cdot h = h \sum_{i=1}^n f(a + (i - \frac{1}{2})h). \quad (4.2)$$

Die Approximation wird in Abbildung 4.2 veranschaulicht. Für die Berechnung der Summe (4.2) sind genau n Auswertungen der Funktion $f(x)$ notwendig.

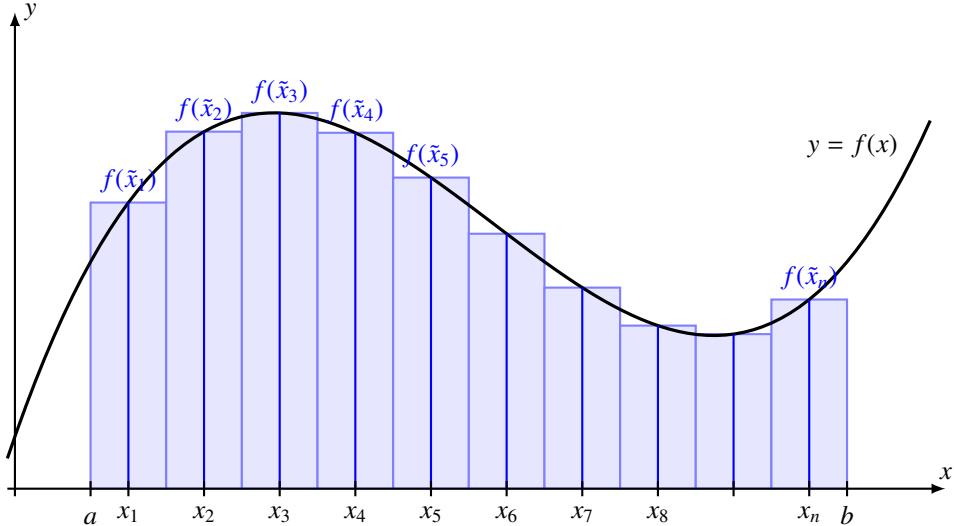


Abbildung 4.2: Approximation eines Integrals mit Hilfe der Mittelpunktsregel

Beispiel. Als Beispiel approximieren wir das Integral

$$I = \int_1^9 x dx = \left[\frac{x^2}{2} \right]_1^9 = \frac{9^2 - 1^2}{2} = \frac{81 - 1}{2} = 40$$

mit Schritten der Schrittweite $h = 2$. Die Zwischenpunkte sind $x_1 = 2, x_2 = 4, x_3 = 6, x_4 = 8$ oder $x_n = 2n$. Die Mittelpunktregel liefert die Approximation

$$M(2) = \sum_{i=1}^n f(2i) \cdot 2 = \sum_{i=1}^n 4i = 4 \sum_{i=1}^n i = 4 \cdot \frac{n(n+1)}{2} = 4 \cdot \frac{4 \cdot 5}{2} = 4 \cdot 2 \cdot 5 = 40.$$

In diesem Beispiel liefert die Mittelpunktformel also den exakten Wert des Integrals. In Abbildung 4.3 kann man den Grund dafür erkennen. Der Graph ist eine Gerade und die “fehlenden” Dreieck in der Summe $M(h)$ links und rechts vom Funktionswert sind gleich gross, so dass sich diese Fehler wegheben. \circlearrowright

Das Beispiel illustriert, dass die Approximation gar nicht so schlecht ist, wie sich auf Grund der deutlich hervortretenden Stufen in Abbildung 4.2 vielleicht vermuten lässt.

4.1.3 Trapezregel

Statt die Fläche unter dem Graphen von $f(x)$ in jedem Teilintervall durch den Wert im Mittelpunkt des Intervalls zu berechnen wie bei der Mittelpunktsregel, könnte sie auch approximiert werden durch ein Trapez mit Eckpunkten $(x_{i-1}, 0), (x_i, 0), (x_{i-1}, f(x_{i-1}))$ und $(x_i, f(x_i))$ (siehe Abbildung 4.4). Die Mittellinie eines solchen Trapezes hat die Länge $m = \frac{1}{2}(f(x_{i-1}) + f(x_i))$, die Höhe ist $x_i - x_{i-1} = h$. Der Flächeninhalt eines einzelnen Trapezes ist daher $\frac{h}{2}(f(x_{i-1}) + f(x_i))$ und es ergibt sich

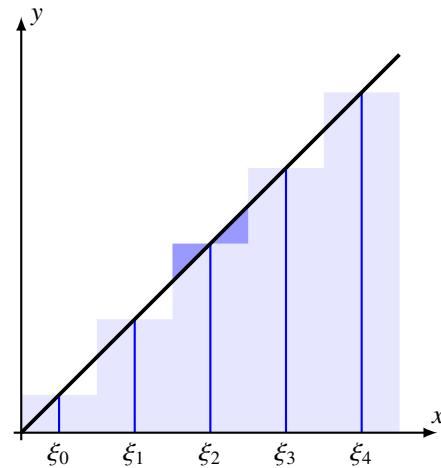


Abbildung 4.3: Auswertung des Integrals $\int_0^1 x \, dx$ mit Hilfe der Mittelpunktformel. In diesem speziellen Fall liefert die Mittelpunktformel den exakten Wert des Integrals, wie die hervorgehobenen Dreiecke illustrieren.

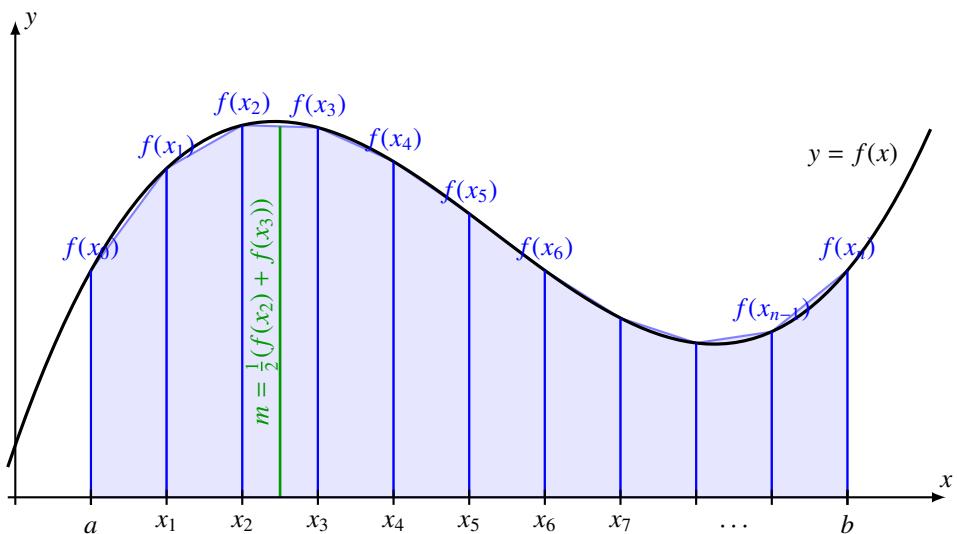


Abbildung 4.4: Approximation eines Integrals mit Hilfe der Trapezregel

die Approximation

$$\begin{aligned} \int_a^b f(x) dx &\approx T(h) = \frac{h}{2} \sum_{i=1}^n (f(x_{i-1}) + f(x_i)) \\ &= (\frac{1}{2}f(x_0) + f(x_1) + f(x_2) + \dots + f(x_{n-2}) + f(x_{n-1}) + \frac{1}{2}f(x_n)) \cdot h \end{aligned} \quad (4.3)$$

für das Integral. In Abbildung 4.4 ist die Summe der Trapezflächen dargestellt. Die Approximation $T(h)$ in (4.3) heisst *Trapezregel*. Die Trapezregel erfordert genau $n + 1$ Funktionsauswertungen.

Verfeinerung

Die Trapezregel $T(h)$ benötigt die Funktionswerte an den Stellen $x_i = a + hi$. Halbiert man die Schrittweite, müssen zusätzlich die Funktionswerte an den Zwischenpunkten

$$\frac{x_{i-1} + x_i}{2} = a + h(i - \frac{1}{2})$$

für $i = 1, \dots, n$ bestimmt werden. Das sind genau die Funktionswerte, die in der Mittelpunktformel summiert wurden. Somit kann man die Trapezsumme $T(\frac{h}{2})$ durch Umordnen der Terme in der Summe durch die Trapezsummen $T(h)$ und die Mittelpunktformel $M(h)$ ausdrücken:

$$\begin{aligned} T(\frac{h}{2}) &= \frac{h}{2} \cdot (\frac{1}{2}f(a) + f(a + 1 \cdot \frac{h}{2}) + f(a + 2 \cdot \frac{h}{2}) + \dots + f(a + (2n - 1) \cdot \frac{h}{2}) + \frac{1}{2}f(a + 2n \cdot \frac{h}{2})) \\ &= \frac{h}{2} \cdot (\text{gerade Terme}) + \frac{h}{2} \cdot (\text{ungerade Terme}) \\ &= \frac{h}{2} \left(\frac{1}{2}f(a) + f(a + 2 \cdot \frac{h}{2}) + f(a + 4 \cdot \frac{h}{2}) + \dots + \frac{1}{2}f(a + (2n) \cdot \frac{h}{2}) \right) \\ &\quad + \frac{h}{2} \left(f(a + 1 \cdot \frac{h}{2}) + f(a + 3 \cdot \frac{h}{2}) + \dots + f(a + (2n - 1) \cdot \frac{h}{2}) \right) \\ &= \frac{1}{2} \cdot h \cdot \left(\frac{1}{2}f(a) + f(a + h) + f(a + 2h) + \dots + \frac{1}{2}f(a + nh) \right) \\ &\quad + \frac{1}{2} \cdot h \cdot \left(f(a + (1 - \frac{1}{2})h) + f(a + (2 - \frac{1}{2})h) + \dots + f(a + (n - \frac{1}{2})h) \right) \\ &= \frac{1}{2}T(h) + \frac{1}{2}M(h). \end{aligned} \quad (4.4)$$

Halbierung der Schrittweite bedeutet daher nur, dass man die Mittelpunktregel $M(h)$ zusätzlich auswerten muss.

Für die Berechnung von $T(\frac{h}{2})$ sind $2n + 1$ Funktionsauswertungen notwendig, davon entfallen $n + 1$ auf die Berechnung von $T(h)$ und n auf die Berechnung von $M(h)$.

Beispiel. Als Beispiel berechnen wir das Integral

$$I = \int_0^1 \sqrt{1 - x^2} dx. \quad (4.5)$$

Es berechnet die Fläche des im ersten Quadranten liegenden Teils des Einheitskreises, hat also den Wert $I = \frac{\pi}{4} \approx 0.78539816$.

Das Octave-Programm von Listing 4.1 kann zur Berechnung verwendet werden. In Zeile 5 wird der Integrand als Funktion definiert. In Zeile 9 wird die Funktion $M(h, a, b, f)$ definiert, die die

```

1 || a = 0;
2 || b = 1;
3 || h = b-a;
4
5 || function y = f(x)
6 ||   y = sqrt(1 - x*x);
7 || endfunction
8
9 || function summe = M(h, a, b, f)
10 ||   x = a + h/2;
11 ||   s = 0;
12 ||   while (x < b)
13 ||     s = s + f(x);
14 ||     x = x + h;
15 ||   end
16 ||   summe = h * s;
17 || endfunction
18
19 || T = h * ((1/2) * f(a) + (1/2) * f(b))
20
21 || for k = (2:20)
22 ||   T = (1/2) * T + (1/2) * M(h, a, b, @f);
23 ||   h = h/2;
24 || endfor

```

Listing 4.1: Programm zur Berechnung der Trapez- und Mittelpunktsummen für das Integral (4.5)

Mittelpunktsumme $M(h)$ der Funktion f berechnet. Es ist nicht nötig, eine Funktion für $T(h)$ zu definieren, da sich der Wert $T(\frac{h}{2})$ aus $T(h)$ und $M(h)$ ergibt. In Zeile 19 wird der erste Wert $T(1)$ berechnet. In der nachfolgenden Schleife zwischen Zeilen 21 und 24 wird die Rekursionsformel (4.4) angewendet. Die sich ergebenden Zahlenwerte sind in Tabelle 4.1 zusammengestellt.

Für den Wert von $T(h)$ in Zeile k der Tabelle sind $2^{k-1} + 1$ Funktionsauswertungen nötig, auf der letzten Zeile sind dies 524289 Funktionsauswertungen. Es fällt auf, dass die Konvergenz ziemlich langsam ist. Der Fehler von $T(h)$ nimmt ungefähr so schnell ab, wie h kleiner wird. Sollte sich diese Systematik bestätigen lassen, könnte man daraus ein Verfahren zur Beschleunigung der Konvergenz ableiten. ○

4.1.4 Fehler von Trapez- und Mittelpunktsregel

Im Beispiel im vorangegangenen Abschnitt hat sich in Tabelle 4.1 eine Gesetzmässigkeit in der Entwicklung der Fehler von $T(h)$ gezeigt. Es schien, als wäre $I - T(h) \sim h$. Ziel dieses Abschnittes ist, dies nachzuweisen. Als Werkzeug dafür benötigen wir die Euler-Maclaurinsche Summenformel.

Partielle Integration

Der Fehler der Trapezregel dürfte umso grösser werden, je stärker der Graph der Funktion $f(x)$ gekrümmmt ist. Wir vermuten daher einen Zusammenhang zwischen dem Fehler und der zweiten

k	$T(h)$	h
1	0.5000000000000000	1.0000000000000000
2	0.6830127018922193	0.5000000000000000
3	0.7489272670256102	0.2500000000000000
4	0.7724547860892934	0.1250000000000000
5	0.7808132594569352	0.0625000000000000
6	0.7837756057192828	0.0312500000000000
7	0.7848242281949210	0.0156250000000000
8	0.7851951980991537	0.0078125000000000
9	0.7853263957393075	0.0039062500000000
10	0.7853727881799138	0.0019531250000000
11	0.7853891916347545	0.0009765625000000
12	0.7853949913528619	0.0004882812500000
13	0.7853970419019385	0.0002441406250000
14	0.7853977668874246	0.0001220703125000
15	0.7853980232097235	0.0000610351562500
16	0.7853981138335553	0.0000305175781250
17	0.7853981458739578	0.0000152587890625
18	0.7853981572019572	0.0000076293945312
19	0.7853981612070116	0.0000038146972656
20	0.7853981626230124	0.0000019073486328
exakt	0.7853981633974483	

Tabelle 4.1: Approximation des Integrals (4.5) mit Hilfe der Trapezregel. Unterstrichen sind die Stellen des Wertes von $T(h)$, die bereits korrekt sind.

Ableitung von $f(x)$. Um diesen Zusammenhang zu finden, betrachten wir das Integral

$$I = \int_0^1 g(t) dt. \quad (4.6)$$

Den Integranden können wir auch als Produkt $1 \cdot g(t)$ der konstanten Funktion 1 mit $g(t)$ schreiben und das Integral partiell ausführen:

$$I = \int_0^1 1 \cdot g(t) dt = \left[p_1(t) \cdot g(t) \right]_0^1 - \int_0^1 p_1(t) \cdot g'(t) dt$$

wobei wir für $p_1(t)$ ein beliebiges Polynom wählen können, dessen Ableitung $p'_1(t) = 1$ ist. Das zweite Integral können wir nochmals partiell integrieren und erhalten

$$= \left[p_1(t) \cdot g(t) \right]_0^1 - \left[p_2(t) \cdot g'(t) \right]_0^1 + \int_0^1 p_2(t) \cdot g''(t) dt$$

wobei wieder $p'_2(t) = p_1(t)$ sein muss. Dieser Prozess lässt sich natürlich wiederholen und ergibt nach k Schritten

$$= \left[p_1(t) \cdot g(t) \right]_0^1 - \left[p_2(t) \cdot g'(t) \right]_0^1 + \left[p_3(t) \cdot g''(t) \right]_0^1 - \left[p_4(t) \cdot g'''(t) \right]_0^1 + \dots$$

$$-(-1)^k \left[p_k(t) \cdot g^{(k-1)}(t) \right]_0^1 + (-1)^k \int_0^1 p_k(t) g^{(k)}(t) dt \quad (4.7)$$

Die Polynome $p_k(t)$ sind bis jetzt noch nicht eindeutig festgelegt, es ist nur die Rekursionsformel

$$p'_k(t) = p_{k-1}(t) \quad \text{für alle } k > 0 \quad \text{und} \quad p_0(t) = 1$$

gegeben. Durch eine geschickte Wahl der Polynome soll jetzt aus der Entwicklung möglichst viel Information über das Integral extrahiert werden.

Wir betrachten zunächst den Fall, dass $g(t)$ selbst ein Polynom ist. Genügend hohe Ableitungen eines Polynoms verschwinden: Ist die Ordnung k der Ableitung grösser als der Grad des Polynoms, dann ist $g^{(k)}(t) = 0$. Dies bedeutet, dass in der Summe (4.7) der Integralterm verschwindet, sobald k grösser als der Grad von g ist. Aber auch für $k = \deg g$ kann man den letzten Term zum verschwinden bringen, wenn man $p_k(t)$ geeignet wählt. In diesem Fall ist $g^{(k)}(t)$ konstant, der Integralterm verschwindet also genau dann, wenn das Integral von $p_k(t)$ verschwindet. Wir legen daher die Polynome $p_k(t)$ durch die Forderung fest, dass

$$\int_0^1 p_k(t) dt = 0 \quad \forall k \quad (4.8)$$

sein soll.

Die Polynome $p_k(t)$

Wir berechnen die ersten paar Polynome der Folge $p_k(t)$. Zunächst ist $p_1(t) = t + C$, wir müssen also C so wählen, dass das Integral

$$\int_0^1 p_1(t) dt = \int_0^1 t + C dt = \left[\frac{1}{2}t^2 + Ct \right]_0^1 = \frac{1}{2} + C$$

verschwindet. Es folgt $C = -\frac{1}{2}$ und $p_1(t) = t - \frac{1}{2}$.

Für $p_2(t)$ folgt zunächst durch Integration $p_2(t) = \frac{1}{2}t^2 - \frac{1}{2}t + C$ und dann aus der Integralbedingung (4.8)

$$\int_0^1 p_2(t) dt = \left[\frac{1}{6}t^3 - \frac{1}{4}t^2 + Ct \right]_0^1 = \frac{1}{6} - \frac{1}{4} + C = -\frac{1}{12} + C,$$

woraus $C = \frac{1}{12}$ folgt und damit $p_2(t) = \frac{1}{2}t^2 - \frac{1}{2}t - \frac{1}{12}$. Auf dies Art kann man schrittweise die folgenden Polynome finden:

$$\begin{aligned} p_1(t) &= t - \frac{1}{2} \\ p_2(t) &= \frac{1}{2}t^2 - \frac{1}{2}t + \frac{1}{12} \\ p_3(t) &= \frac{1}{6}t^3 - \frac{1}{4}t^2 + \frac{1}{12}t \\ p_4(t) &= \frac{1}{24}t^4 - \frac{1}{2}t^3 + \frac{1}{24}t^2 - \frac{1}{720} \\ p_5(t) &= \frac{1}{120}t^5 - \frac{1}{240}t^4 + \frac{1}{360}t^3 - \frac{1}{3600}t \end{aligned}$$

Der führende Koeffizient des Polynoms $p_k(t)$ ist $1/k!$, indem man diesen ausklammert, kann man etwas einfachere Polynome bekommen:

$$\begin{array}{ll} B_0(t) = 1 & \\ p_1(t) = 1 \cdot B_1(t) & B_1(t) = t - \frac{1}{2} \\ p_2(t) = \frac{1}{2!} \cdot B_2(t) & B_2(t) = t^2 - t + \frac{1}{6} \\ p_3(t) = \frac{1}{3!} \cdot B_3(t) & B_3(t) = t^3 - \frac{3}{2}t^2 + \frac{1}{2}t \\ p_4(t) = \frac{1}{4!} \cdot B_4(t) & B_4(t) = t^4 - 2t^3 + t^2 - \frac{1}{30} \\ p_5(t) = \frac{1}{5!} \cdot B_5(t) & B_5(t) = t^5 - \frac{5}{2}t^4 - \frac{5}{3}t^3 - \frac{1}{6}t \end{array}$$

Die Polynome $B_k(t)$ heißen *Bernoulli-Polynome*.

Eigenschaften der Polynome $p_k(t)$

Auch ohne die expliziten Formeln des vorangegangenen Abschnittes lassen sich nützliche Symmetrieeigenschaften der Polynome $p_k(t)$ bezüglich des Punktes $t = \frac{1}{2}$ ableiten.

Wir untersuchen die Symmetrie der Polynome bezüglich des Punktes $t_0 = \frac{1}{2}$. Wir schreiben $\tilde{p}_k(t)$ für die Stammfunktion von $p_{k-1}(t)$ mit Integrationskonstante $C_k = 0$, es ist also

$$p_k(t) = \tilde{p}_k(t) + C_k \quad \text{und} \quad \tilde{p}_k(0) = 0.$$

Die Integrationskonstante C_k wurde im vorangegangenen Abschnitt bereits bestimmt.

Zunächst halten wir fest, dass die Polynome $p_k(t)$ alle entweder gerade oder ungerade sind bezüglich t_0 .

Lemma 4.1. Die Polynome $p_{2k}(t)$ sind gerade bezüglich $\frac{1}{2}$, die Polynome $p_{2k+1}(t)$ sind ungerade bezüglich $\frac{1}{2}$. Insbesondere ist $p_{2k}(0) = p_{2k}(1)$ und $p_{2k+1}(1) = -p_{2k+1}(0)$.

Beweis. Die ersten zwei Polynome $p_0(t) = 1$ und $p_1(t) = t - \frac{1}{2}$ haben tatsächlich die genannten Symmetrieeigenschaften. Um die Aussage zu beweisen muss jetzt also nur untersucht werden, dass $p_k(t)$ die verlangten Eigenschaften hat, wenn alle vorangegangenen Polynome die “richtigen” Symmetrien haben.

Zunächst lernt man in der Analysis, dass die Ableitung einer geraden Funktion ungerade ist und die Ableitung einer ungeraden Funktion gerade. Dies impliziert auch, dass die Ableitung einer bezüglich t_0 geraden Funktion ungerade ist bezüglich t_0 und umgekehrt.

Die Funktion $p_k(t)$ ist jedoch eine Stammfunktion, nicht eine Ableitung. Die Stammfunktion einer bezüglich t_0 ungeraden Funktion $p_{k-1}(t)$ ist gerade bezüglich t_0 , unabhängig von der gewählten Integrationskonstante, da eine Konstante auch gerade ist bezüglich t_0 . Die Stammfunktion einer bezüglich t_0 geraden Funktion $p_{k-1}(t)$ ist jedoch nicht automatisch ungerade bezüglich t_0 , sondern nur dann, wenn sie den Wert 0 hat in t_0 . Es folgt, dass $p_k(t) - p_k(t_0)$ eine bezüglich t_0 ungerade Funktion ist.

Im vorliegenden Fall ist $t_0 = \frac{1}{2}$ und wir haben die zusätzliche Bedingung (4.8), die wir in dem Fall, wo $p_{k-1}(t)$ gerade ist, ausnutzen können. Da $p_k(t) - p_k(\frac{1}{2})$ ungerade ist, gilt

$$0 = \int_0^1 p_k(t) - p_k\left(\frac{1}{2}\right) dt = \int_0^1 p_k(t) dt - p_k\left(\frac{1}{2}\right).$$

Darin verschwindet der erste Summand wegen (4.8), also gilt auch $p_k(\frac{1}{2}) = 0$. Folglich ist $p_k(t)$ ungerade bezüglich $\frac{1}{2}$. \square

Die spezielle Wahl der Polynome $p_k(t)$ führt jetzt dazu, dass die Formel (4.7) für das Integral (4.6) von $g(t)$ über das Intervall $[0, 1]$ weiter vereinfacht werden kann. Die Werte von $p_k(t)$ an den Intervallgrenzen haben gemäss Lemma 4.1 gleiche Werte für gerade k und entgegengesetzte Werte für ungerade k . Es gilt

$$\begin{aligned} I = \int_0^1 g(t) dt &= p_1(1)(g(1) + g(0)) - p_2(1)(g'(1) - g'(0)) + p_3(1)(g''(1) + g''(0)) \\ &\quad - \cdots + (-1)^k \int_0^1 p_k(t) g^{(k)}(t) dt \end{aligned} \quad (4.9)$$

Für $k = 2$ ergibt sich die einfachere Formel

$$\int_0^1 g(t) dt = \underbrace{\frac{1}{2}(g(1) + g(0)) + p_2(1)(g'(0) - g'(1))}_{= T(1)} + \int_0^1 p_2(t) g''(t) dt, \quad (4.10)$$

welche einen Zusammenhang zwischen dem Integral und der einfachsten Form der Trapezsumme herstellt.

Euler-Maclaurin-Summenformel

Unser Interesse gilt den Summen $T(h)$ und $M(h)$, wir möchten gerne den Fehler solcher Summen abschätzen. Die Formel (4.10) vergleicht ein einzelnes Trapez (den Wert $T(1)$ auf der rechten Seite) mit dem gesuchten Integral und gibt eine Formel für den Unterschied. Um den Fehler der Trapezformel zu bestimmen, müssen wir Summen von vielen Trapezen berechnen.

Wir beginnen mit Schrittänge $h = 1$ bestimmen jetzt das Integral

$$I_n = \int_0^n g(t) dt.$$

Wir zerlegen das Integral in viele einzelne Teilintegrale auf Intervallen der Länge 1:

$$\int_0^n g(t) dt = \int_0^1 g(t) dt + \int_1^2 g(t) dt + \int_2^3 g(t) dt + \cdots + \int_{n-1}^n g(t) dt$$

und wenden die Formel (4.9) auf jeden einzelnen Term an

$$\begin{aligned} &= p_1(1) \sum_{i=1}^n (g(i) + g(i-1)) + p_2(1) \sum_{i=1}^n (g'(i) - g'(i-1)) \\ &\quad + p_3(1) \sum_{i=1}^n (g''(i) + g''(i-1)) + p_4(1) \sum_{i=1}^n (g'''(i) - g'''(i-1)) \\ &\quad + \int_0^1 p_k(t) \sum_{i=1}^n g^{(4)}(k+t) dt. \end{aligned}$$

Man beachte, dass Ableitungen gerader Ordnung als Summen in die Summen eingehen, die Ableitungen ungerader Ordnung jedoch als Differenzen.

In den Summen über Differenzen heben sich die inneren Terme jeweils weg:

$$\begin{aligned} \sum_{i=1}^n (g'(i) - g'(i-1)) &= g'(1) - g'(0) + g'(2) - g'(1) + g'(3) - g'(2) + \cdots + g'(n) - g'(n-1) \\ &= g'(n) - g'(0) \\ \sum_{i=1}^n (g'''(i) - g'''(i-1)) &= g'''(1) - g'''(0) + g'''(2) - g'''(1) + \cdots + g'''(n) - g'''(n-1) \\ &= g'''(n) - g'''(0). \end{aligned}$$

Damit kann man die Formel für das Integral weiter vereinfachen und erhält

$$\begin{aligned} \int_0^n g(t) dt &= p_1(1) \sum_{i=1}^t (g(i) + g(i-1)) + p_2(1)(g'(n) - g'(0)) \\ &\quad + p_3(1) \sum_{i=1}^n (g''(i) + g''(i-1)) + p_4(1)(g'''(n) - g'''(0)) + \int_0^1 p_4(t) \sum_{i=1}^n g^{(4)}(i+t) dt. \end{aligned} \tag{4.11}$$

Dies ist ein Spezialfall der *Euler-Maclaurinschen Summenformel*.

Fehler der Trapezformel

Die Euler-Maclaurinsche Summenformel (4.11) kann noch etwas vereinfacht werden. Wir wissen, dass $p_3(t)$ eine bezüglich $t_0 = \frac{1}{2}$ ungerade Funktion ist, also ist $p_3(1) = -p_3(0) = 0$, wie man aus der früher hergeleiteten expliziten Formel für $p_3(t)$ berechnen kann. Damit vereinfacht sich die Formel (4.11) zu

$$\begin{aligned} \int_0^n g(t) dt &= \sum_{i=1}^n \frac{1}{2} (g(i-1) + g(i)) + p_2(1)(g'(n) - g'(0)) + p_4(1)(g'''(n) - g'''(0)) \\ &\quad + \int_0^1 p_4(t) \sum_{i=1}^n g^{(4)}(i+t) dt. \end{aligned} \tag{4.12}$$

Der erste Term auf der rechten Seite sieht aus wie eine Trapezsumme, nur ist es die falsche Funktion und es sind die falschen Teilpunkte.

Durch eine geeignete Variablentransformation können wir das Intervall $[a, b]$ auf das Intervall $[0, n]$ transformieren. Dazu schreiben wir

$$x = a + \frac{b-a}{n} \cdot t = a + ht \quad \text{mit} \quad h = \frac{b-a}{n}.$$

Die Funktion

$$g(t) = f(x) = f(a + ht)$$

kann dann zur Berechnung des Integrals herangezogen werden. Zunächst folgt aus der Variablentransformation

$$\int_0^n g(t) dt = \int_0^n f(\underbrace{a+ht}_x) dt \quad \text{mit} \quad t = \frac{x-a}{h}$$

$$= \frac{1}{h} \int_a^b f(x) dx$$

$$\Rightarrow \int_a^b f(x) dx = h \int_0^n g(t) dt.$$

In der Euler-Maclaurinschen Summenformel brauchen wir die Ableitungen von $g(t)$, die wir natürlich wieder durch Ableitungen von $f(x)$ ausdrücken möchten:

$$g'(t) = hf'(a + ht) = hf'(x)$$

$$g''(t) = h^2 f''(x)$$

$$\vdots \quad \vdots$$

$$g^{(k)}(t) = h^k f^{(k)}(x)$$

Setzen wir dies alles in die Summenformel (4.12) ein, erhalten wir

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{i=1}^n \frac{h}{2} (f(x_{i-1}) + f(x_i)) + \frac{h^2}{12} (f'(b) - f'(a)) - \frac{h^4}{720} (f'''(b) - f'''(a)) \\ &\quad + h \int_0^1 p_4(t) \sum_{i=1}^n g^{(4)}(i+t) dt \\ &= T(h) + \frac{h^2}{12} (f'(b) - f'(a)) - \frac{h^4}{720} (f'''(b) - f'''(a)) \\ &\quad + h \sum_{i=1}^n \int_0^1 p_4(t) g^{(4)}(i+t) dt. \end{aligned}$$

Die Integrale im letzten Term können wir wieder auf die Variable x transformieren. Dazu verwenden wir für den Summanden i die Variablentransformation

$$x = x_{i-1} + ht \quad \Rightarrow \quad t = \frac{1}{h}(x - x_{i-1}), \quad dt = \frac{1}{h} dx$$

und erhalten

$$\begin{aligned} \int_0^1 p_4(t) g^{(4)}(i+t) dt &= \int_{x_{i-1}}^{x_i} p_4\left(\frac{1}{h}(x - x_{i-1})\right) h^4 f^{(4)}(x) \frac{1}{h} dx \\ &= h^3 \int_{x_{i-1}}^{x_i} p_4\left(\frac{1}{h}(x - x_{i-1})\right) f^{(4)}(x) dx \end{aligned}$$

Die Funktion p_4 im Integranden kann mit Hilfe der früher abgeleiteten expliziten Formel für $p_4(t)$ abgeschätzt werden. Sei K eine Konstante so, dass $|p_4(t)| \leq K$ für $0 \leq t \leq 1$. Dann ist können wir das Integral auf der rechten Seite abschätzen durch

$$\left| \int_0^1 p_4(t) g^{(4)}(i+t) dt \right| \leq h^3 \int_{x_{i-1}}^{x_i} K |f^{(4)}(x)| dx.$$

Summieren wir über i erhalten wir

$$\left| \sum_{i=1}^n \int_0^1 p_4(t) g^{(4)}(i+t) dt \right| \leq h^3 \sum_{i=1}^n \int_{x_{i-1}}^{x_i} K |f^{(4)}(x)| dx = h^3 K \int_a^b |f^{(4)}(x)| dx = h^3 K_1$$

mit einer neuen Konstanten K_1 .

Nach dieser langen Rechnung können wir jetzt den Fehler für die Trapezformel vollständig hinschreiben

$$\begin{aligned} \int_a^b f(x) dx &= T(h) + h^2 \cdot \frac{f'(b) - f'(a)}{12} - h^4 \frac{f'''(b) - f'''(a)}{720} + h^4 K_1 \\ &= T(h) + Ch^2 + O(h^4). \end{aligned} \quad (4.13)$$

Die dominante Komponente des Fehlers ist also der Term Ch^2 , der verbleibende Fehler ist von der Größenordnung $O(h^4)$, also im allgemeinen viel kleiner.

Man beachte, dass diese Abschätzungen nur funktionieren, wenn die dritten Ableitungen von $f(x)$ an den Grenzen und die vierten Ableitungen von $f(x)$ im Inneren des Intervalls $[a, b]$ nicht übermäßig gross werden.

Wie muss h gewählt werden?

Wie gross h genau gewählt werden muss hängt natürlich von der Konstanten C ab. Aus

$$C = \frac{f'(b) - f'(a)}{12}$$

kann man gegebenenfalls ein Abschätzung gewinnen. Zum Beispiel weiss man, dass die Ableitung der Funktion $\sin x$ betragsmäßig niemals grösser als 1 werden kann. Für das Integral der Sinusfunktion kann man also davon ausgehen, dass höchstens $C = 1/6$ ist. Einen Fehler der Grösse ε erhält man also, wenn man h so wählt, dass $h^2/6 < \varepsilon$ ist, oder $h < \sqrt{6\varepsilon}$. Für 10 Nachkommastellen bedeutet dies $h < \sqrt{6} \cdot 10^{-10}$. Um das Integral über das Intervall $[0, \pi]$ mit dieser Schrittweite zu berechnen, sind über 128000 Funktionsauswertungen notwendig.

Man kann die Frage auch umgekehrt stellen: Welche Genauigkeit kann man sinnvollerweise von der Trapezregel erwarten? Dabei kann man ein in vernünftiger Zeit durchführbare Rechnung von etwa 10^3 bis 10^4 Funktionsauswertungen zu Grunde legen. Dies führt im genannte Beispiel des Integrals der Sinusfunktion auf eine Schrittweite zwischen $h = 0.003$ und $h = 0.0003$. Für diese h wird der Fehler zwischen 0.00001 und 0.0000001 gross. Viel mehr als etwa sechs Nachkommastellen lassen sich mit der Trapezregel alleine also nicht gewinnen. Dies wird auch durch die Rechnungen im Beispiel im nächsten Abschnitt bestätigt.

Das Beispiel auf Seite 122 zeigt auch, dass die Konvergenz noch schlechter sein kann. Da die Ableitung des Integranden in diesem Beispiel an der Stelle $x = 1$ divergiert, wird der Koeffizient C in der Fehlerformel sehr gross.

4.2 Romberg-Algorithmus

Die bisher betrachteten Methoden zur Berechnung von Integralen zeichnen sich dadurch aus, dass die Fehler bei einer gegebenen Schrittweite sehr genau bekannt sind. Es sollte daher möglich sein, die Ideen von Abschnitt 1.4.2 anzuwenden und die Konvergenz des Verfahrens zu beschleunigen, ohne dass weitere Funktionswerte berechnet werden müssen.

Elimination des Fehlers

Wir kehren zurück zur Berechnung des Integrals

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

mit Hilfe der Trapezregel und bezeichnen wieder mit $T(h)$ die Approximation mit der Trapezregel mit Schrittweite h . In (4.13) haben wir den Fehler von $T(h)$ bestimmt. Wir nehmen an, dass der Fehler vierter Ordnung im Vergleich zum Term Ch^2 vernachlässigbar ist. Wenn wir die Schrittweite immer wieder halbieren, erhalten wir immer bessere Approximationen

$$\begin{aligned} I &= T(h) + Ch^2 \\ &= T\left(\frac{h}{2}\right) + \frac{Ch^2}{4}. \end{aligned}$$

Nehmen wir für den Moment an, dass dies exakte Gleichungen sind, dann können wir die Unbekannte C eliminieren und nach der Unbekannten I auflösen. Dazu multiplizieren wir die zweite Gleichung mit 4 und subtrahieren die erste Gleichung. Wir erhalten

$$3I = 4T\left(\frac{h}{2}\right) - T(h) \quad \Rightarrow \quad I = \frac{4T(h/2) - T(h)}{3}.$$

Natürlich ist dies auch noch nicht der exakte Wert des Integrals, wir haben ja die Terme vierter Ordnung vernachlässigt. Wir können aber daraus schliessen, dass

$$T^*(h/2) = \frac{4T(h/2) - T(h)}{3}$$

eine Approximation für das Integral ist, deren Fehler nur noch von vierter Ordnung $O(h^4)$ sein wird.

Da wir für die Approximationen $T^*(h)$ auch wieder eine Approximation für die Fehler haben, können wir das gleiche Prinzip nochmals anwenden und auch den Fehler vierter Ordnung eliminieren. Multiplizieren wir die zweite Gleichung in

$$\begin{aligned} I &= T^*(h) + Ch^4 \\ &= T^*(h/2) + C \frac{h^4}{16} \end{aligned}$$

mit 16 und subtrahieren die erste, so erhalten wir

$$15I = 16T^*(h/2) - T(h) \quad \Rightarrow \quad T^{**}(h/2) = I = \frac{16T^*(h/2) - T^*(h)}{15}.$$

Dieses Verfahren der fortlaufenden Verbesserung der Konvergenz durch Elimination des Fehlers ist bekannt als *Romberg-Algorithmus*.

Ein Beispiel

Wir berechnen das Integral

$$I = \int_0^\pi \sin x dx = 2$$

k	$T(\pi 2^{-k})$	$T^*(\pi 2^{-k})$	$T^{**}(\pi 2^{-k})$
0	0.0000000000000002		
1	1.5707963267948966	2.0943951023931953	
2	1.8961188979370398	2.0045597549844207	1.9985707318238357
3	1.9742316019455508	2.0002691699483877	1.9999831309459855
4	1.9935703437723393	2.0000165910479355	1.9999997524545718
5	1.9983933609701441	2.0000010333694123	1.9999999961908441
6	1.9995983886400366	2.0000000645300009	1.9999999999407068
7	1.9998996001842024	2.0000000040322576	1.999999999990750
8	1.9999749002350549	2.0000000002520060	1.999999999999891
9	1.9999937250705797	2.0000000000157545	2.0000000000000044
10	1.9999984312683776	2.000000000009770	1.9999999999999920
11	1.9999996078171345	2.000000000000533	1.9999999999999918

Tabelle 4.2: Berechnung des Integrals $\int_0^\pi \sin x dx$ mit Hilfe der Trapezregel (Spalte $T(\pi 2^{-k})$) und Beschleunigung der Konvergenz mit Hilfe des Romberg-Algorithmus. Für die Resultate auf der Zeile $k > 0$ sind $2^k + 1$ Funktionsauswertungen notwendig.

mit Hilfe der Trapezregel und wenden das Beschleunigungsschema von Romberg auf die erhaltenen Werte an.

Die gefundenen Resultate sind in Tabelle 4.2 dargestellt. Man kann erkennen, dass die Maschinengenauigkeit von 15 Nachkommastellen mit der Trapezregel auch bei $k = 11$ nicht erreicht wird. Für die erste Romberg-Beschleunigung T^* wird die Genauigkeit bei $k = 11$ erreicht, sie kann aber durch Erhöhung von k nicht mehr verbessert werden, da die aufsummierten Rundungsfehler wichtiger werden. In beiden Fällen muss der Integrand 2049 mal ausgewertet werden.

Die zweite Romberg-Beschleunigung erreicht Maschinengenauigkeit bereits bei $k = 9$, als mit nur 513 Funktionsauswertungen. Die Romberg-Beschleunigung ist in diesem Beispiel also ausserordentlich erfolgreich.

Allgemeine Form des Romberg-Verfahrens

In noch allgemeinerer Form kann man die Formeln des Romberg-Algorithmus wie folgt schreiben. Die Werte $T_{k,0} = T(2^{-k}h)$ mit $k = 0, 1, \dots$ sind die Werte der Trapezsumme mit $h = 2^{-k}(b - a)$. Die Beschleunigungen sind

$$\begin{aligned} T_{k,1} &= \frac{4T_{k,0} - T_{k-1,0}}{4 - 1} \\ T_{k,2} &= \frac{4^2 T_{k,1} - T_{k-1,1}}{4^2 - 1} \\ &\vdots && \vdots \\ T_{k,l} &= \frac{4^l T_{k,l-1} - T_{k-1,l-1}}{4^l - 1} \end{aligned}$$

Dieses Verfahren funktioniert und beschleunigt die Konvergenz, sofern die Ableitungen von $f(x)$ bis zur Ordnung 2/ ausreichend glatt und beschränkt sind.

Wieviele Stellen Genauigkeit lassen sich erreichen?

Das Romberg-Verfahren basiert darauf, dass man die Trapezsummen für sukzessiv halbiert Schritt-länge berechnet. In jeder Iteration wird die Schrittweite also halbiert. Die Fehler von $T_{k,l}$ sind von der Ordnung h^{2l} , also

$$T_{k,l} = I + O(2^{-2(l+1)}).$$

Auf jeder Zeile wird der Fehler also um den Faktor $2^{-2l} = \frac{1}{4^l}$ kleiner, dies entspricht einem Gewinn von zwei Binärstellen oder $2(l+1)\log_{10}2 = 0.60206(l+1)$ Dezimalstellen. Man kann dies auch in der Tabelle 4.2 verfolgen. In der ersten Spalte mit $l=0$ braucht es etwa 10 Zeilen, um sechs Dezimalstellen zu gewinnen, in der zweiten Spalte mit $l=1$ sind sechs Dezimalstellen bereits nach fünf Zeilen erreicht, 12 Dezimalstellen nach 10 Zeilen. In der dritten Spalte mit $l=2$ erreicht man 12 Stellen bereits ungefähr nach fünf Zeilen.

4.3 Weitere Integrationsverfahren

Die bisher vorgestellten Verfahren haben vor allem den Mangel, dass die Zahl der Funktionsauswertungen sehr gross wird, wenn eine hohe Genauigkeit angestrebt wird. Das Verfahren der *Gauss-Quadratur*, im Kapitel 10 vorgestellt, findet hochgenaue Integralwerte mit einer sehr geringen Anzahl von Funktionsauswertungen.

Die Wahrscheinlichkeitsrechnung hat einen besonderen Bedarf nach Integralauswertungen, wie zum Beispiel die Berechnung der Verteilungsfunktion der Normalverteilung. In diesen Anwendungen gibt es jedoch häufig auch die Möglichkeit, die gesuchte Wahrscheinlichkeit durch Simulation eines Wahrscheinlichkeitsexperimentes zu erhalten. Dies führt zu den sogenannten *Monte-Carlo-Methoden*.

Übungsaufgaben

4.1. Berechnen Sie eine Approximation des Integrals der Funktion $f(x)$ über das Intervall $[a, b]$ mit folgender Idee. Bestimmen Sie erst ein Interpolationspolynom zweiten Grades, welches mit f in den Punkten a, b und dem Mittelpunkt m des Intervalls $[a, b]$ übereinstimmt. Dann ist

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

eine Approximation. Schreiben Sie I als Ausdruck in $f(a), f(b)$ und $f(m)$. Diese Formel ist bekannt als die *Simpsonsche Regel* oder auch die *Keplersche Fassregel*.

Lösung. Zur Bestimmung des Interpolationspolynoms braucht man das $l(x) = (x-a)(x-m)(x-b)$ und die Polynome vom Grad 2

$$\begin{aligned} l_0(x) &= \frac{1}{(a-m)(a-b)}(x-m)(x-b), \\ l_1(x) &= \frac{1}{(m-a)(m-b)}(x-a)(x-b), \\ l_2(x) &= \frac{1}{(b-a)(b-m)}(x-a)(x-m). \end{aligned} \tag{4.14}$$

Das Interpolationspolynom ist dann

$$p(x) = f(a)l_0(x) + f(m)l_1(x) + f(b)l_2(x).$$

Für das Integral muss man jetzt die Integrale der einzelnen Polynome $l_k(x)$ bestimmen. Dies ist etwas mühsam, aber durch Variablentransformation auf die Werte $a = -1$, $b = 1$ und $m = 0$ wird es einfacher:

$$\begin{aligned} \int_{-1}^1 \frac{x(x-1)}{(-1)(-1-1)} dx &= \frac{1}{2} \left[\frac{1}{3}x^3 - \frac{1}{2}x^2 \right]_{-1}^1 = \frac{1}{3}, \\ \int_{-1}^1 \frac{(x+1)(x-1)}{(0-(-1))(0-1)} dx &= - \int_{-1}^1 x^2 - 1 dx = - \left[\frac{1}{3}x^3 - x \right]_{-1}^1 = -\frac{2}{3} + 2 = \frac{4}{3}, \\ \int_{-1}^1 \frac{(x+1)x}{1-(-1)(1-0)} dx &= \frac{1}{3}. \end{aligned}$$

Zusammen mit den Normierungsnennern in (4.14) erhalten wir somit

$$\begin{aligned} \int_a^b l_0(x) dx &= \frac{b-a}{2} \cdot \frac{1}{3} = \frac{b-a}{6} \\ \int_a^b l_1(x) dx &= \frac{b-a}{2} \cdot \frac{4}{3} = \frac{4(b-a)}{6} \\ \int_a^b l_2(x) dx &= \frac{b-a}{2} \cdot \frac{1}{3} = \frac{b-a}{6}. \end{aligned}$$

Die verschiedenen Faktoren im Nenner können wir alle durch die Intervalllänge $b-a$ ausdrücken. Es ist zum Beispiel $m-a = \frac{1}{2}(b-a)$ und $m-b = -\frac{1}{2}(m-a)$. Damit erhalten wir jetzt für das Integral von $p(x)$

$$\int_a^b p(x) dx = \frac{b-a}{6}(f(a) + 4f(m) + f(b)). \quad \circlearrowright$$

4.2. Berechnen Sie das Integral

$$I = \int_0^1 \sin x dx$$

der Funktion $f(x) = \sin x$ auf vier verschiedene Weisen:

- a) Exakt.
- b) Mit Hilfe der Mittelpunktsformel mit genau zwei Teilintervallen.
- c) Mit Hilfe der Trapezformel mit genau zwei Teilintervallen.
- d) Mit Hilfe der Simpsonschen Formel

$$\int_a^b f(x) dx \approx I_S = \frac{b-a}{6}(f(a) + 4f(m) + f(b)) \quad \text{mit } m = (a+b)/2$$

(Aufgabe 4.1).

Lösung. a) Die exakte Berechnung des Integrals liefert

$$I = \int_0^1 \sin x \, dx = \left[-\cos x \right]_0^1 = 1 - \cos 1 \approx 0.45969769413186028260.$$

b) Mit der Mittelpunktsformel ist die Näherung

$$I_M = \frac{1}{2} f(\frac{1}{4}) + f(\frac{3}{4}) = 0.46452135963892854816.$$

c) Mit der Trapezformel ist die Näherung

$$I_T = \frac{1}{4} (f(0) + 2f(\frac{1}{2}) + f(1)) = \frac{1}{4} (0 + 2 \sin \frac{1}{2} + \sin 1) = 0.45008051550407562679.$$

d) Mit der Simpsonschen Regel findet man

$$I_S = \frac{1}{6} (f(0) + 4f(\frac{1}{2}) + f(1)) = \frac{2}{3} \sin \frac{1}{2} + \frac{1}{6} \sin 1 = 0.45986218987078475127.$$

Die Mittelpunktsformel und die Trapezregel verwenden Interpolation höchstens vom Grad 1 zur Approximation des Integrals. Die Simpsonsche Formel verwendet ein quadratisches Approximationspolynom, von dem man eine etwas höhere Genauigkeit erwarten kann. Wir vergleichen die Resultate in der folgenden Tabelle

Methode		Wert
exakt	I	0.45969769413186028260
Mittelpunktsformel	I_M	0.46452135963892854816
Trapezformel	I_T	0.45008051550407562679
Simpsonsche Regel	I_S	0.45986218987078475127

Die Simpsonsche Regel scheint also tatsächlich etwas besser zu sein.

Man kann mit Hilfe der Simpsonschen Regel ebenfalls eine Integrationsmethode ähnliche wie die Trapezregel finden. Dieser zusätzliche Aufwand lohnt sich aber meistens nicht, da sich mit der Trapezregel und anschliessender Romberg-Beschleunigung die gleiche Genauigkeit leichter erreichen lässt. ○

5

Gewöhnliche Differentialgleichungen

Gewöhnliche Differentialgleichungen gehören zu den wichtigsten Problemen, die vor allem mit numerischen Methoden gelöst werden. Nur selten lassen sich Probleme der Praxis in geschlossener Form lösen. In diesem Kapitel wird nach einer Rekapitulation der Problemstellung in Abschnitt 5.1 zunächst das Grundprinzip der numerischen Lösungsverfahren vorgestellt. In Abschnitt 5.4 wird das Runge-Kutta-Verfahren entwickelt, welches einen guten Mittelweg zwischen Genauigkeit und Komplexität darstellt. Später im Kapitel kommen Mehrschrittverfahren und das Randwertproblem zur Rede.

5.1 Problemstellung

Eine *gewöhnliche Differentialgleichung* für eine reellwertige Funktion $y(x)$ stellt einen Zusammenhang her zwischen der Funktion und ihren Ableitungen. Wir schreiben die Ableitungen als y' , y'' , y''' und $y^{(n)}$ für die n -te Ableitung. Wir lassen oft das Argument der Funktion weg. Beispiele von Differentialgleichungen sind

$$\begin{array}{ll} y' = -Ny & \text{Ordnung: 1} \\ y'' = -\omega^2 y & \text{Ordnung: 2} \\ x^2 y'' + xy' + (x^2 - n^2)y = 0 & \text{Ordnung: 2} \end{array}$$

Die Abhängigkeit kann in expliziter Form als

$$y^{(n)} = f(x, y, y', \dots, y^{(n-1)}) \quad (5.1)$$

oder in impliziter Form

$$F(x, y, y', \dots, y^{(n)}) = 0$$

gegeben sein. Die *Ordnung* einer Differentialgleichung ist die höchste vorkommende Ableitung.

Differentialgleichungen erster Ordnung lassen sich mit Hilfe eines Richtungsfeldes visualisieren, wie in Abbildung 5.1 dargestellt. In jedem Punkt (x, y) der x - y -Ebene wird die Steigung $y' = f(x, y)$ eingezeichnet. Eine Lösung der Differentialgleichung hat in diesem Bild als Graph eine Kurve in der x - y -Ebene, die an jeder Stelle das Richtungsfeld hat.

Insbesondere in Anwendungen in der Physik ist die Zeit die unabhängige Variable. Die abhängige Variable ist dann zum Beispiel die Ortskoordinate $x(t)$ und wir bezeichnen ihre Ableitungen mit

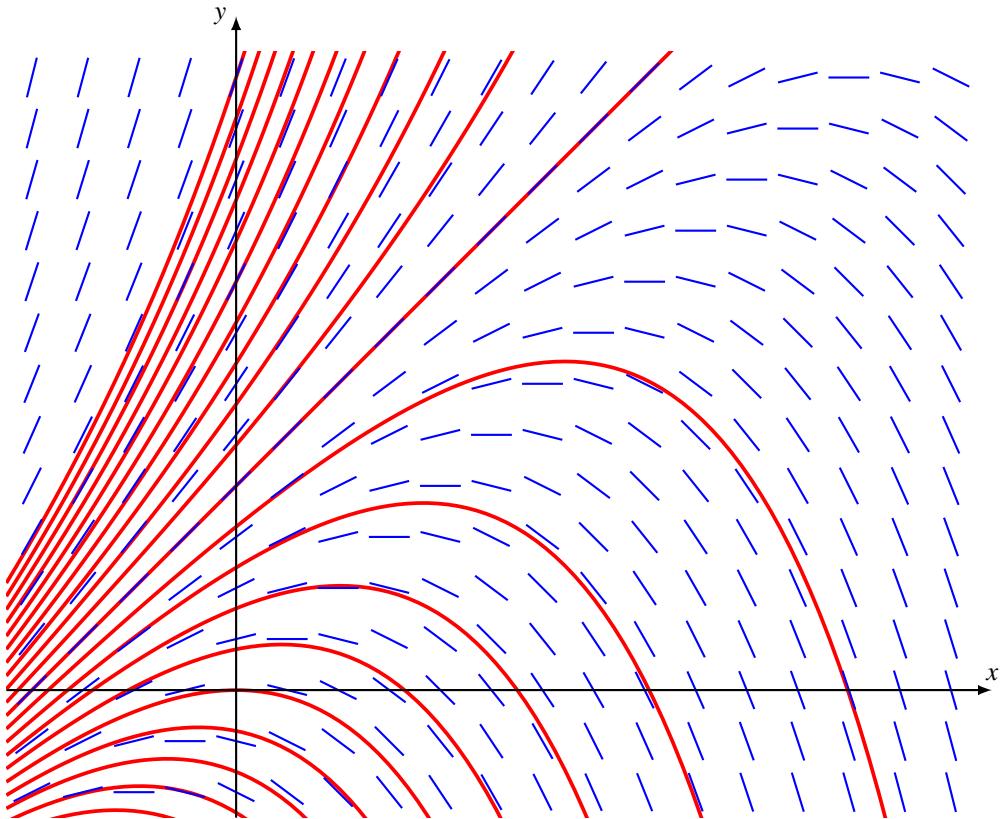


Abbildung 5.1: Richtungsfeld der Differentialgleichung $y' = y - x$ mit einzelnen Lösungskurven.

$\dot{x}(t)$ für die Geschwindigkeit, $\ddot{x}(t)$ für die Beschleunigung. Dieses Beispiel suggeriert auch, dass die abhängige Variable ein Vektor sein kann, den man als den Ortsvektor eines Teilchens interpretieren kann. Auch die Funktion $f(t, x, \dots, x^{(n-1)})$ muss dann vektorwertig sein und ebenso alle Argumente von f ausser dem Ersten.

5.1.1 Reduktion der Ordnung

Eine Differentialgleichung n -ter Ordnung für eine skalare Funktion kann in eine Vektor-Differentialgleichung erster Ordnung für eine n -dimensionale vektorwertige Funktion umgewandelt werden. Ist $y(x)$ die gesuchte Funktion in der Differentialgleichung (5.1), dann kann man den Vektor

$$u(x) = \begin{pmatrix} y(x) \\ y'(x) \\ \vdots \\ y^{(n-1)}(x) \end{pmatrix} \in \mathbb{R}^n$$

bilden. Er erfüllt die Differentialgleichung

$$\frac{d}{dx} \begin{pmatrix} y \\ y' \\ \vdots \\ y^{(n-1)} \end{pmatrix} = \begin{pmatrix} y' \\ y'' \\ \vdots \\ y^{(n)} \end{pmatrix} = \begin{pmatrix} y' \\ y'' \\ \vdots \\ f(x, y, y', \dots, y^{(n-1)}) \end{pmatrix}. \quad (5.2)$$

Der Vektor auf der rechten Seite hängt nur von x , der Funktion y und ihren Ableitungen bis zur $n - 1$ -ten Ordnung ab, also von u , man kann (5.2) daher als

$$\frac{d}{dx} u = \tilde{f}(x, u) \quad (5.3)$$

schreiben. Im Folgenden werden wir fast ausschliesslich Differentialgleichungen erster Ordnung der Form $y' = f(x, y)$ betrachten und dabei stillschweigend zulassen, dass y ein Vektor ist.

5.1.2 Anfangswertprobleme

Die Differentialgleichung $y' = f(x, y)$ alleine kann eine Lösungsfunktion $y(x)$ nicht festlegen, sie codiert nur, wie sich die Lösung verändern wird. Es ist also zusätzlich die Angabe eines Punktes der Lösungskurve notwendig. Man nennt das Problem, eine Funktion $y(x)$ zu finden, welche

$$y' = f(x, y) \quad \text{und} \quad y(0) = y_0$$

erfüllt, ein *Anfangswertproblem*. Ein Anfangswertproblem verlangt für die gewöhnliche Differentialgleichung n -ter Ordnung verlangt also die Angabe der Werte von $y(0), y'(0), \dots, y^{(n-1)}(0)$

Existenz und Eindeutigkeit von Lösungen

Die Existenz und Eindeutigkeit einer Lösung ist aus den Beispielen und graphischen Darstellungen intuitiv verständlich, für einen exakten Beweis sind jedoch zusätzliche Voraussetzungen nötig.

Definition 5.1. Eine Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ heisst global Lipschitz-stetig, wenn es eine Zahl L gibt

$$|f(x_2) - f(x_1)| \leq L |x_2 - x_1| \quad (5.4)$$

für alle Vektoren $x_1, x_2 \in \mathbb{R}^n$. Eine Funktion heisst lokal Lipschitz-stetig im Punkt x_0 , wenn die Bedingung (5.4) für x_i in einer Umgebung von x_0 erfüllt ist.

Eine Funktion ist insbesondere dann lokal Lipschitz-stetig, wenn sie stetig differenzierbar ist. In diesem Fall ist die Ableitung $f'(x)$ in einer Umgebung von x_0 beschränkt, also $|f'(x)| < M$, und der Mittelwertsatz der Differentialrechnung sagt, dass

$$|f(x_2) - f(x_1)| \leq M|x_2 - x_1|$$

ist, f ist also lokal Lipschitz-stetig.

Satz 5.2 (Picard-Lindelöf). Ist die Funktion $f(x, y)$ lokal Lipschitz-stetig bezüglich der Variablen y für $x \in [x_0, b]$ und $|y - y_0| < R$. Dann hat das Anfangswertproblem

$$y'(x) = f(x, y) \quad \text{und} \quad y(x_0) = y_0$$

ein eindeutige Lösung, die in einem Intervall $[x_0, x_0 + \varepsilon)$ definiert ist.

In diesem Buch werden die Funktionen f der Differentialgleichungen meistens stetig differenzierbar sein, so dass der Satz 5.2 in unseren Anwendungen die lokale Existenz und Eindeutigkeit einer Lösung garantiert.

5.1.3 Randwertprobleme

Wenn man einen Ball wirft, wird seine Bewegung durch die Vektordifferentialgleichung zweiter Ordnung

$$\frac{d^2}{dt^2} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{g}{m} \end{pmatrix}$$

beschrieben. Die Bahn ist ausserdem bestimmt durch die Anfangsbedingungen, d. h. den Anfangspunkt und die Anfangsgeschwindigkeit der Bahn. Praktischer Ballwurf verlangt aber, dass ein Ziel getroffen wird. Die Aufgabenstellung ist daher eine Bahnkurve $\gamma(t)$ zu finden, welche sowohl durch den Anfangspunkt als auch den Zielpunkt verläuft.

Die Lösung einer Differentialgleichungen erster Ordnung für eine unbekannte reellwertige Funktion $y(x)$ ist vollständig durch einen einzigen Anfangswert bestimmt. Eine Differentialgleichung zweiter Ordnung für eine unbekannte reellwertige Funktion $y(x)$ verlangt dagegen zwei Anfangswerte, nämlich für $y(0)$ und $y'(0)$. In Analogie zum Problem des Ballwurfs könnte die Lösungsfunktion auch festgelegt werden durch den Wert für $x = 0$ und $x = 1$. Gesucht ist also eine Funktion $y(x)$ auf dem Intervall $[0, 1]$, die

$$y'' = f(x, y, y') \quad \text{mit} \quad y(0) = y_0, \quad y(1) = y_1 \quad (5.5)$$

erfüllt. Die Lösungsfunktion muss also bestimmte Werte am Rand des Definitionsbereichs annehmen, man spricht von einem *Randwertproblem*.

Beispiel. Wir lösen die Differentialgleichung $y'' = -y$ mit den Randwerten $y(0) = 1$ und $y(1) = 2$. Die homogene Differentialgleichung hat die Funktionen

$$y(x) = A \cos x + B \sin x$$

als allgemeine Lösung. Die Konstanten A und B müssen so gewählt werden, dass die Randwerte korrekt sind. Setzt man $x = 0$ und $x = 1$ ein, erhält man die linearen Gleichungen

$$\begin{aligned} a &= y(0) = A \cos 0 + B \sin 0 & \Rightarrow & & A &= 0 \\ b &= y(1) = A \cos 1 + B \sin 1 & \Rightarrow & & B &= \frac{b - a \cos 1}{\sin 1}. \end{aligned}$$

Die Lösung des Randwertproblems ist daher die Funktion

$$y(x) = a \cos x + \frac{b - a \cos 1}{\sin 1} \sin x,$$

wie man auch durch Einsetzen von $x = 0$ und $x = 1$ verifizieren kann. ○

5.1.4 Höhere Ableitungen

Die Differentialgleichung $y' = f(x, y)$ erlaubt nicht nur die erste Ableitung einer Funktion zu bestimmen. Durch Ableitung nach x können wir auch die höheren Ableitungen bestimmen, die eine Lösung der Differentialgleichung haben muss, die durch den Punkt (x, y) verläuft.

$$\begin{aligned} y'(x) &= f(x, y), \\ y''(x) &= \frac{dy'(x)}{dx} = \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y} y'(x) = \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y} f(x, y), \end{aligned} \quad (5.6)$$

$$\begin{aligned}
y'''(x) &= \frac{dy''(x)}{dx} \\
&= \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y \partial x} y'(x) + \left(\frac{\partial^2 f(x, y)}{\partial x \partial y} + \frac{\partial^2 f(x, y)}{\partial y^2} y'(x) \right) f(x, y) \\
&\quad + \frac{\partial f(x, y)}{\partial y} \left(\frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y} f(x, y) \right), \\
&= \frac{\partial^2 f(x, y)}{\partial x^2} + 2 \frac{\partial^2 f(x, y)}{\partial x \partial y} f(x, y) + \frac{\partial^2 f(x, y)}{\partial y^2} f(x, y)^2 + \left(\frac{\partial f(x, y)}{\partial y} \right)^2 f(x, y) + \frac{\partial f(x, y)}{\partial y} \frac{\partial f(x, y)}{\partial x}.
\end{aligned} \tag{5.7}$$

Die Terme werden offensichtlich schnell kompliziert. Die Rechnung zeigt aber auch, dass die Lösung einer Differentialgleichung mindestens so oft differenzierbar ist wie die Funktion $f(x, y)$, die die Differentialgleichung definiert. Die Möglichkeit, höhere Ableitungen der Lösung zu berechnen, ermöglicht zusammen mit dem Taylor-Polynom ein recht genaues Lösungsverfahren für gewöhnliche Differentialgleichungen, welches im Kapitel 12 beschrieben wird.

5.1.5 Ableitung nach der Anfangsbedingung

Verändert man die Anfangsbedingung, ändert sich auch die Lösung, die Komponente y_i ist also eine Funktion von x und von allen Anfangswerten y_{j0} :

$$y_i(x, y_{10}, \dots, y_{n0}).$$

Wenn man untersuchen will, wie empfindlich die y_i auf Änderungen der Anfangswerte reagieren, dann sucht man die Ableitungen der y_i nach den Anfangswerten, also die sogenannte Jacobi-Matrix

$$J(x) = \begin{pmatrix} \frac{\partial y_1}{\partial y_{10}} & \cdots & \frac{\partial y_1}{\partial y_{n0}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial y_{10}} & \cdots & \frac{\partial y_n}{\partial y_{n0}} \end{pmatrix}$$

Wir schreiben $J(x)$ abkürzend auch

$$J(x) = \frac{\partial y}{\partial y_0}.$$

Differentialgleichung für $J(x)$

Für $x = 0$ ist $y(x) = y(0) = y_0$, die Ableitung der y -Werte nach den Anfangswerten ist daher die Einheitsmatrix:

$$J(0) = E,$$

und es liegt nahe, dass auch $J(x)$ eine Differentialgleichung erfüllen muss.

Wie ändert sich $J(x)$ zwischen x und $x + \Delta x$? Die Werte von y ändern sich gemäss der Differentialgleichung

$$\frac{dy}{dx} = f(x, y), \tag{5.8}$$

aber y hängt von den Anfangsbedingungen ab. Leiten wir (5.8) nach y_0 ab, finden wir

$$\frac{\partial}{\partial y_0} \frac{dy}{dx} = \frac{\partial}{\partial y_0} f(x, y). \tag{5.9}$$

Die rechte Seite wir mit der Kettenregel zu

$$\frac{\partial}{\partial y_{j_0}} f_i(x, y) = \sum_{k=1}^n \frac{\partial f_i(x, y)}{\partial y_k} \underbrace{\frac{\partial y_k}{\partial y_{j_0}}}_{J_{ij}(x)}.$$

Schreiben wir die Ableitungen von f nach y in die Matrix

$$\frac{\partial f(x, y)}{\partial y} = \begin{pmatrix} \frac{\partial f_1(x, y)}{\partial y_1} & \dots & \frac{\partial f_1(x, y)}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(x, y)}{\partial y_1} & \dots & \frac{\partial f_n(x, y)}{\partial y_n} \end{pmatrix} = F(x, y), \quad (5.10)$$

kann die Gleichung (5.9) mit dem Matrizenprodukt als

$$\frac{\partial}{\partial y_0} \frac{dy}{dx} = F(x, y) J(x)$$

sehr viel kompakter geschrieben werden. Die Ableitungen auf der linken Seite vertauschen,

$$\frac{\partial}{\partial y_0} \frac{d}{dx} = \frac{d}{dx} \frac{\partial y}{\partial y_0}.$$

Setzen wir dies in die Gleichung (5.10), erhalten wir den folgenden Satz.

Satz 5.3 (Differentialgleichung für die Jacobi-Matrix $J(x)$). Ist $y(x, y_0)$ die Lösung der Differentialgleichung

$$y' = f(x, y) \quad \text{mit Anfangsbedingung} \quad y(0) = y_0,$$

dann erfüllt die Jacobi-Matrix

$$\frac{\partial y}{\partial y_0} = J(x)$$

die Differentialgleichung

$$\frac{d}{dx} \frac{\partial y}{\partial x} = \frac{d}{dx} J(x) = F(x, y) J(x) \quad \text{mit Anfangsbedingung} \quad J(x) = E, \quad (5.11)$$

wobei F die Ableitung von f nach y ist,

$$F(x, y) = \frac{\partial f(x, y)}{\partial y}.$$

Vor allem bei der numerischen Lösung mit dem Computer lässt sich die Jacobi-Matrix also gleich mit bestimmen, indem man die gefundene Lösung y als Input für die Gleichung (5.11) verwendet.

Die Möglichkeit, die Jacobi-Matrix zu berechnen, wird sich im Abschnitt 5.7.2 bei der numerischen Lösung von Randwertproblemen als besonders nützlich erweisen.

Beispiel

Wir betrachten wieder die Schwingungsdifferentialgleichung $y'' = -y$, oder vielmehr die Form erster Ordnung

$$\frac{d}{dx} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix}.$$

Die Ableitungsmatrix $F(x, y)$ ist

$$F(x, y) = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Die Jacobi-Matrix erfüllt also die Differentialgleichung

$$\frac{d}{dx} J(x) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} J(x).$$

Da wir die Differentialgleichung bereits vollständig gelöst und die Lösung

$$y(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix} = \begin{pmatrix} y_{10} \cos x + y_{20} \sin x \\ -y_{10} \sin x + y_{20} \cos x \end{pmatrix} = \begin{pmatrix} \cos x & \sin x \\ -\sin x & \cos x \end{pmatrix} \begin{pmatrix} y_{10} \\ y_{20} \end{pmatrix} \quad (5.12)$$

gefunden haben, können wir die Jacobi-Matrix auch aus der Lösung berechnen, und verifizieren, ob sie die Differentialgleichung erfüllt. Die Jacobi-Matrix von (5.12) ist

$$J(x) = \frac{\partial y}{\partial y_0} = \begin{pmatrix} \cos x & \sin x \\ -\sin x & \cos x \end{pmatrix}.$$

Die Ableitung nach x davon ist

$$\frac{d}{dx} J(x) = \begin{pmatrix} -\sin x & \cos x \\ -\cos x & -\sin x \end{pmatrix}.$$

Setzen wir dies in die Differentialgleichung (5.11) ein finden wir

$$F(x, y)J(x) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \cos x & \sin x \\ -\sin x & \cos x \end{pmatrix} = \begin{pmatrix} -\sin x & \cos x \\ -\cos x & -\sin x \end{pmatrix} = \frac{d}{dx} J(x),$$

die Differentialgleichung ist also tatsächlich erfüllt.

5.1.6 Numerische Lösung der Differentialgleichung für $y(x)$ und $J(x)$

Für die numerische Berechnung von $y(x)$ und $J(x)$ müssen die beiden Differentialgleichungen

$$y' = f(x, y) \quad \text{und} \quad J' = F(x, y)J$$

in eine einzige zusammengefasst werden. Wir schreiben dazu die Vektoren $y(x)$ und die Matrix $J(x)$ in einen einzigen Vektor

$$Y(x) = \begin{pmatrix} y(x) \\ J(x) \end{pmatrix},$$

wobei wir irgend eine Methode verwenden, eine Matrix als Vektor anzurufen. Welche Methode dazu verwendet wird ist egal, solange es immer die Gleiche ist. Zum Beispiel können die Matrixelemente zeilenweise in den Spaltenvektor Y abgefüllt werden. In dieser Notation wird die zusammengefasste Differentialgleichung

$$\frac{d}{dx} \begin{pmatrix} \textcolor{red}{y} \\ \textcolor{red}{J} \end{pmatrix} = \begin{pmatrix} f(x, \textcolor{red}{y}) \\ F(x, \textcolor{red}{y}) \textcolor{red}{J} \end{pmatrix} \quad \text{mit der Anfangsbedingung} \quad Y(0) = \begin{pmatrix} y_0 \\ E \end{pmatrix}, \quad (5.13)$$

wobei wir die unbekannten Funktionen rot hervorgehoben haben, um die Abhängigkeiten deutlicher zu machen.

Wir schreiben dies noch aus für den Fall $n = 2$. In diesem Fall sind insgesamt sechs Funktionen zu bestimmen, die zwei Komponenten von y , $y_1(x)$ und $y_2(x)$ und die vier Komponenten von $J(x)$. Um daraus eine einzige Differentialgleichung zu machen, packen wir die sechs Funktionen in einen Vektor $Y(x)$

$$Y(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ J_{11}(x) \\ J_{12}(x) \\ J_{21}(x) \\ J_{22}(x) \end{pmatrix}.$$

Für die Ableitung der ersten zwei Komponenten verwenden wir die Differentialgleichung (5.8), für die J -Komponenten aber die Differentialgleichung (5.11). So erhalten wir die kombinierte Differentialgleichung

$$\frac{d}{dx} Y(x) = \frac{d}{dx} \begin{pmatrix} y_1(x) \\ y_2(x) \\ J_{11}(x) \\ J_{12}(x) \\ J_{21}(x) \\ J_{22}(x) \end{pmatrix} = \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \\ \frac{\partial f_1(x, y)}{\partial x_1} J_{11}(x) + \frac{\partial f_1(x, y)}{\partial x_2} J_{21}(x) \\ \frac{\partial f_1(x, y)}{\partial x_1} J_{12}(x) + \frac{\partial f_1(x, y)}{\partial x_2} J_{22}(x) \\ \frac{\partial f_2(x, y)}{\partial x_1} J_{11}(x) + \frac{\partial f_2(x, y)}{\partial x_2} J_{21}(x) \\ \frac{\partial f_2(x, y)}{\partial x_1} J_{12}(x) + \frac{\partial f_2(x, y)}{\partial x_2} J_{22}(x) \end{pmatrix} \quad (5.14)$$

mit Anfangswerten

$$Y(x_0) = \begin{pmatrix} y_1(x_0) \\ y_2(x_0) \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \quad (5.15)$$

5.1.7 Abhängigkeit von Parametern

Im Allgemeinen wird eine Differentialgleichung von Parametern abhängen, so dass die Lösung nicht nur von der Anfangsbedingung, sondern auch von den Werten dieser Parameter abhängen wird. Um dies explizit zu machen, fassen wir die Parameter in einen Vektor c zusammen und machen

die Abhängigkeit der Differentialgleichung von c durch ein zusätzliches Argument der Funktion f explizit:

$$\frac{dy}{dx} = f(x, y, c).$$

Die Lösung $y(x)$ hängt dann zusätzlich von der Wahl der Parameter ab, wir können das durch die Schreibweise $y(x, c)$ sichtbar machen.

Oft sucht man dann geeignete Parameter, für die die Lösung der Differentialgleichung bestimmte Eigenschaften hat, zum Beispiel durch einen bestimmten Punkt verläuft, d. h. wir suchen Werte c derart, dass $y(x_1, c) = y_1$. Ein besonders effizientes Verfahren zur Bestimmung solcher Parameterwerte ist das Newton-Verfahren, welches aber die Ableitungen

$$\frac{\partial y(x, c)}{\partial c} = \begin{pmatrix} \frac{\partial y_1(x, c)}{\partial c_1} & \cdots & \frac{\partial y_1(x, c)}{\partial c_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n(x, c)}{\partial c_1} & \cdots & \frac{\partial y_n(x, c)}{\partial c_n} \end{pmatrix} = C(x)$$

von $y(x, c)$ nach c benötigt.

Um die Abhängigkeit von c zu berechnen, leiten wir die Differentialgleichung nach c ab und erhalten

$$\frac{\partial}{\partial c} \frac{d}{dx} y(x, c) = \frac{\partial f(x, y, c)}{\partial y} \frac{\partial y}{\partial c} + \frac{\partial f(x, y, c)}{\partial c}.$$

Wieder können auf der linken Seite die beiden Ableitungen vertauscht werden, so dass eine Differentialgleichung für $C(x)$ entsteht:

$$\begin{aligned} \frac{d}{dx} \frac{\partial y(x, c)}{\partial c} &= \frac{\partial f(x, y, c)}{\partial y} \frac{\partial y(x, c)}{\partial c} + \frac{\partial f(x, y, c)}{\partial c} \\ \Rightarrow \quad \frac{d}{dx} C(x) &= \frac{\partial f(x, y, c)}{\partial y} C(x) + \frac{\partial f(x, y, c)}{\partial c}. \end{aligned}$$

Dies ist eine lineare inhomogene Differentialgleichung erster Ordnung für die Matrix $C(x)$. Da die Parameter keinen Einfluss auf die Anfangswerte haben, hat $C(x)$ den Anfangswert $C(x_0) = 0$.

5.2 Grundprinzip numerischer Lösungsverfahren

Wir versuchen als einführendes Beispiel, die Differentialgleichung

$$y' = -ay, \quad y(0) = y_0 \tag{5.16}$$

numerisch zu lösen. Die Lösung ist natürlich bekannt, es ist

$$y(x) = y_0 e^{-ax}. \tag{5.17}$$

Dazu unterteilen wir die x -Achse in diskrete Abschnitte der Länge h , genannt die *Schrittweite*, und bezeichnen die Teilpunkte mit $x_k = kh$. Das Ziel ist jetzt, $y(x_k)$ näherungsweise zu berechnen. Wir schreiben y_k für die Näherungswerte von $y(x_k)$.

Die Ableitung liefert eine lineare Approximation für $y(x)$, nämlich

$$y(x + \Delta x) \approx y(x) + y'(x) \cdot \Delta x$$

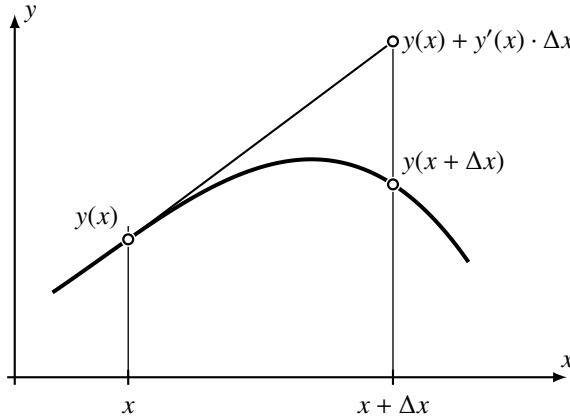


Abbildung 5.2: Lineare Approximation von $y(x + \Delta x)$ durch Information, die am Punkt x verfügbar ist.

(Abbildung 5.2). Für die Punkte x_k bedeutet das

$$y(x_{k+1}) \approx y(x_k) + y'(x_k) \cdot h.$$

Die Differentialgleichung liefert Werte für $y'(x_k)$ aus x_k und $y(x_k)$, damit können wir aus dieser Approximation ein allgemeines Näherungsverfahren für die Lösung einer Differentialgleichung konstruieren.

Satz 5.4 (Euler-Verfahren). *Die Differentialgleichung*

$$y' = f(x, y), \quad y(0) = y_0 \tag{5.18}$$

und die Schrittweite h definieren eine Folge

$$y_k = y_{k-1} + h \cdot f(x_{k-1}, y_{k-1}), \quad k > 0,$$

mit $x_k = kh$, die eine Näherung für die Funktionswerte $y(x_k)$ der Lösung $y(x)$ der Differentialgleichung (5.18) ist.

Dieses Verfahren ist nicht besonders gut, wie wir im Folgenden zeigen wollen. Die Diskussion soll vor allem zeigen, worauf bei der Weiterentwicklung des Verfahrens geachtet werden muss.

Im vorliegenden Beispiel liefert die Differentialgleichung (5.16) den Wert $y'(x_k) = -\alpha y(x_k)$ für die Ableitung, woraus wir die Rekursionsformel

$$y_{k+1} = y_k - \alpha y_k h.$$

gewinnen. Die Rekursionsgleichung kann in diesem Fall exakt gelöst werden, und wir finden

$$y(x_{k+1}) = y(x_k) - \alpha y(x_k) h = (1 - \alpha h)y(x_k) = \dots = (1 - \alpha h)^{k+1} y_0 \tag{5.19}$$

für die Näherung y_k der Funktionswerte $y(x_k)$.

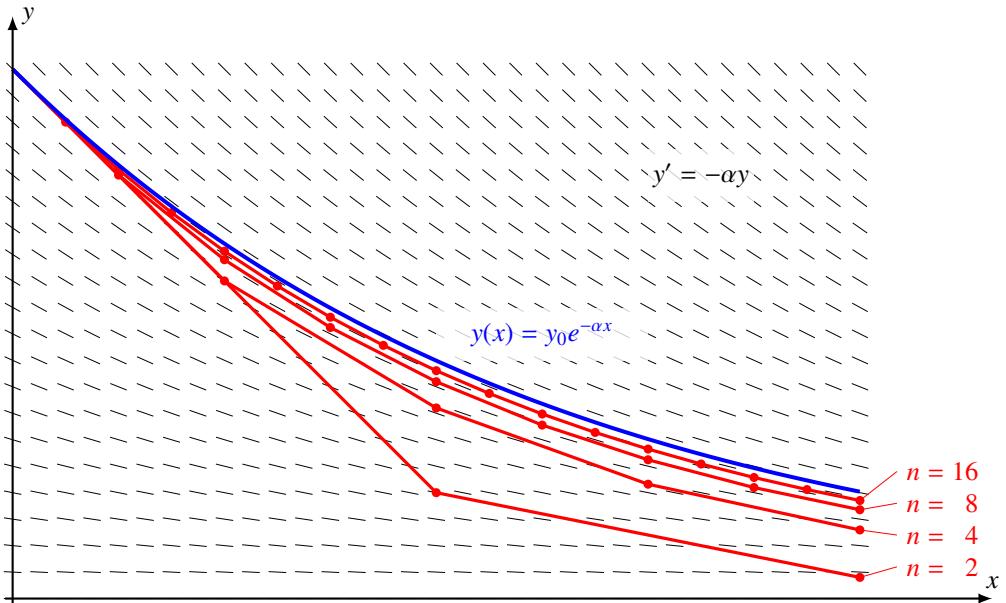


Abbildung 5.3: Approximationen der Lösung der Differentialgleichung $y' = -\alpha y$ mit verschiedener Anzahl Schritte (rot) nähern sich für wachsendes n der exakten Lösung (blau).

Wir möchten $y(x)$ für einen ganz bestimmten x -Wert berechnen. Dazu unterteilen wir das Intervall $[0, x]$ in n Teilschritte der Breite x/n und wenden die Formel (5.19) an:

$$y(x) = y(x_n) = (1 - \alpha h)^n y_0 = \left(1 + \frac{-\alpha x}{n}\right)^n y_0.$$

Für eine grosse Zahl von Teilschritten erhalten wir so tatsächlich die korrekte Lösung:

$$\lim_{n \rightarrow \infty} y_0 \left(1 + \frac{-\alpha x}{n}\right)^n = y_0 e^{-\alpha x}.$$

Abbildung 5.3 zeigt, wie die durch (5.19) gegebenen Approximationen mit zunehmendem n der exakten Lösung $y(x) = e^{-\alpha x}$ näher kommen.

Wir können auch den Fehler des numerischen Verfahrens berechnen. Bei der Schrittweite h ist der Fehler von y_k die Differenz

$$y(x_k) - y_k = y_0 e^{-\alpha h k} - y_0 (1 - \alpha h)^k = y_0 ((e^{-\alpha h})^k - (1 - \alpha h)^k) = y_0 e^{-\alpha h k} \left(1 - \left(\frac{1 - \alpha h}{e^{-\alpha h}}\right)^k\right).$$

Man beachte, dass der Zähler $1 - \alpha h$ die Approximation y_1 ist, als eine Approximation von $e^{-\alpha h}$, dem Nenner. Schreiben wir

$$q = \frac{1 - \alpha h}{e^{-\alpha h}},$$

für den Quotienten zwischen der Approximation und dem korrekten Wert, dann ist sicher immer $q < 1$. Den Fehler können wir jetzt schreiben

$$y(x_k) - y_k = y_0 e^{-\alpha h k} (1 - q^k) = y(x_k) (1 - q^k).$$

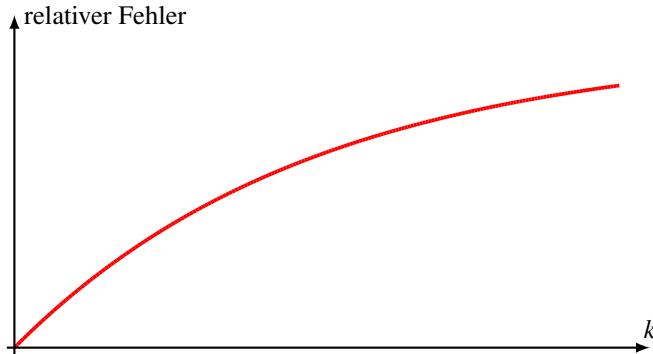


Abbildung 5.4: Relativer Fehler des Euler-Verfahrens für die Differentialgleichung (5.16) in Abhängigkeit von der Anzahl k der Schritte.

Der relative Fehler des Verfahrens ist also

$$\frac{y(x_k) - y_k}{y(x_k)} = (1 - q^k).$$

Ganz unabhängig von der Schrittweite h wird der relative Fehler des Verfahrens immer gegen 1 streben, der Fehler wird also von der gleichen Größenordnung wie die berechneten Resultate.

Die Abbildung 5.4 zeigt, dass zu Beginn des Verfahrens der relative Fehler ungefähr linear mit der Anzahl der Schritte zunimmt. Um eine angemessene Genauigkeit über einen grösseren Bereich zu erreichen, muss das Euler-Verfahren also sehr viel kleinere Schritte und eine entsprechend grössere Anzahl von Schritten ausführen, die viel Rechenzeit benötigen.

Ein praktisch nützliches Verfahren muss also anstreben, mit einer sehr viel kleineren Anzahl von Schritten eine deutlich grössere Genauigkeit der Approximation zu erreichen.

5.3 Fehler-Entwicklung numerischer Lösungen

Wir betrachten wieder die Differentialgleichung (5.18) und versuchen, den Fehler eines Näherungsverfahrens zu bestimmen, welches Schritte der Grösse h durchführt, um den Wert $y(x)$ zu approximieren.

Das Euler-Verfahren verwendet Schritte der Form

$$y_{k+1} = y_k + h f(x_k, y_k).$$

In jedem einzelnen Schritt der Länge Δx entsteht ein Fehler, dessen Grösse wir aus der Taylor-Entwicklung

$$y(x + \Delta x) = y(x) + y'(x) \cdot \Delta x + R(x)\Delta x^2$$

abschätzen können. Die Funktion $R(x)$ ist beschränkt und beschreibt den verbleibenden Fehler. Um $y(x)$ zu approximieren, müssen $n = x/h$ Schritte der Schrittweite h durchgeführt werden, von denen jeder einen Fehler von der Größenordnung $R(x)h^2$ hat. Der Gesamtfehler ist daher von der Größenordnung

$$y(x) - y_n = O\left(R(x)h^2 \frac{x}{h}\right) = O(h),$$

er ist also von erster Ordnung in h . Um eine zusätzliche Stelle Genauigkeit zu erhalten, muss man also zehnmal so viele Schritte von zehnmal kleinerer Länge durchführen. Die höhere Zahl von Einzelschritt führt jedoch auch zusätzliche Rundungsfehler ein.

Könnte man den Fehler des Einzelschrittes wesentlich verkleinern, würde auch die Abhängigkeit des Fehlers des Verfahrens von der Schrittweite vorteilhafter. Wäre der Fehler des Einzelschrittes $O(h^k)$ statt $O(h^2)$, dann wäre der Gesamtfehler des Verfahrens nur noch $O(h^{k-1})$. Für $k = 3$ bedeutet dies, dass eine Halbierung der Schrittweite zwar doppelt so viele Schritte braucht, aber auch, dass in jedem Schritt nur ein Achtel des Fehlers auftritt. Der Gesamtfehler ist also nur ein Viertel. Mit zehnmal mehr Arbeit kann man also nicht nur eine Stelle an Genauigkeit gewinnen, sondern gleich deren zwei.

Man nennt ein Verfahren, bei dem der Gesamt-Fehler von der Größenordnung $O(h^k)$ ist, von einem Verfahren k -ter Ordnung. Das Euler-Verfahren ist also ein Verfahren erster Ordnung oder ein lineares Verfahren. In der Praxis werden Verfahren bis zu vierter und fünfter Ordnung verwendet, so dass eine zehnmal kleinere Schrittweite zu gleich vier Stellen Genauigkeitsgewinn führen. Das Ziel der kommenden Abschnitte muss daher sein, einfach berechenbare Approximationen der Funktion $y(x)$ mit möglichst geringen Einzelschrittfehlern zu finden.

5.4 Einschrittverfahren

Die relativ geringe Genauigkeit des Euler-Schrittes beruht darauf, dass die zu Beginn des Schrittes berechnete Ableitung $f(x_k, y_k)$ nur für das linke Ende des Intervalls $[x_k, x_k + h]$ zutrifft, weiter rechts im Intervall wird die Abweichung immer grösser. Eine mögliche Lösung des Problems könnte darin bestehen, statt nur einer linearen Näherung zusätzliche Glieder der Taylor-Reihe

$$y(x + \Delta x) = y(x) + y'(x) \cdot \Delta x + \frac{1}{2} y''(x) \cdot \Delta x^2 + \frac{1}{6} y'''(x) \cdot \Delta x^3 + o(\Delta x^3) \quad (5.20)$$

zu verwenden. In (5.20) werden höhere Ableitungen von $y(x)$ benötigt, während die Differentialgleichung nur die erste Ableitung liefert. Die höheren Ableitungen wurden aber bereits im Abschnitt 5.1.4 berechnet.

Wir untersuchen, wie sich das Verfahren für die Beispiel-Gleichung (5.16) anwenden lässt. Dort gilt

$$\begin{aligned} y'(x) &= f(x, y) = -\alpha y \\ \Rightarrow \quad \frac{\partial f}{\partial x} &= 0 \quad \frac{\partial f}{\partial y} = -\alpha. \end{aligned}$$

Alle zweiten Ableitungen verschwinden. Die Gleichungen werden damit einfach:

$$\begin{aligned} y''(x) &= -\alpha f(x, y) = \alpha^2 y \\ y'''(x) &= \alpha^2 f(x, y) = -\alpha^3 y. \end{aligned}$$

Statt der linearen Approximation sollte daher die kubische Approximation

$$y_{k+1} = y_k - ah y_k + \frac{1}{2} \alpha^2 h^2 y_k - \frac{1}{6} \alpha^3 h^3 y_k = y_k \underbrace{\left(1 - ah + \frac{1}{2} \alpha^2 h^2 - \frac{1}{6} \alpha^3 h^3\right)}_{\approx e^{-ah}} \quad (5.21)$$

verwendet werden. Dass man hier mit einer grösseren Genauigkeit rechnen darf, ist schon daran erkennbar, dass der Klammerausdruck auf der rechten Seite eine viel bessere Approximation von

i	x	$e^{-\alpha x}$	Euler	kubisch
1	0.1	0.95122942	0.95000000	0.95122917
2	0.2	0.90483742	0.90250000	0.90483693
3	0.3	0.86070798	0.85737500	0.86070728
4	0.4	0.81873075	0.81450625	0.81872987
5	0.5	0.77880078	0.77378094	0.77879973
6	0.6	0.74081822	0.73509189	0.74081702
7	0.7	0.70468809	0.69833730	0.70468675
8	0.8	0.67032005	0.66342043	0.67031859
9	0.9	0.63762815	0.63024941	0.63762660
10	1.0	0.60653066	0.59873694	0.60652902

Tabelle 5.1: Näherungswerte für die Lösung $e^{-\alpha x}$ der Beispieldifferentialgleichung (5.16) nach dem Euler-Verfahren und nach dem kubischen Verfahren (5.21) mit einer Schrittweite von 0.1. Unterstrichen ist jeweils die Stellen, die nach Rundung auf die angegebene Anzahl stellen mit dem exakten Wert übereinstimmt.

$e^{-\alpha x}$ ist also der Faktor $(1-\alpha h)$ im Euler-Verfahren. Genauer erwarten wir, dass wir hier ein kubisches Verfahren konstruiert haben.

In Tabelle 5.1 werden die Resultate des kubischen Verfahrens denen des Euler-Verfahrens gegenübergestellt. Im ersten Schritt ist der Fehler des Euler-Verfahrens kleiner als 10^{-2} , was einer Einheit in der zweiten Nachkommastelle entspricht. Der Fehler des kubischen Verfahrens ist kleiner als 10^{-6} , eine Einheit in der sechsten Nachkommastelle, ungefähr die von einem kubischen Verfahren zu erwartende Verbesserung. Nach zehn Rechenschritten liefert das Euler-Verfahren dank Rundung gerade noch eine korrekte Stelle, während das kubische Verfahren immer noch gerundet fünf korrekte Stellen gibt.

Es wurde bereits darauf hingewiesen, dass die Terme für die Ableitungen sehr kompliziert werden. Noch viel gravierender ist allerdings, dass auch die partiellen Ableitungen von f nach x und y bekannt sein müssen. Es ist zwar im Prinzip möglich, diese zu berechnen, der Rechenaufwand dafür kann aber so erheblich sein, dass er den Genauigkeitsgewinn leicht wieder zunichte machen kann. Praktisch nützliche Verfahren müssen daher danach streben, die höheren Ableitungen von $y(x)$ ausschliesslich aus Funktionswerten von $f(x, y)$ zu berechnen.

Wir möchten aber weiterhin nur y_{k+1} ausschliesslich aus x_k und y_k berechnen, also in einem einzelnen Schritt der Form

$$y_{k+1} = y_k + h F(x_k, y_k, h).$$

Die Funktion $F(x, y, h)$ heisst die *Inkrementfunktion* des Verfahrens. Für das Euler-Verfahren ist $F(x, y, h) = f(x, y)$. Es soll also eine Inkrementfunktion gefunden werden, bei der $y(x + \Delta x)$ durch $y(x) + \Delta x \cdot F(x, y, \Delta x)$ bis auf Terme höherer Ordnung approximiert werden kann.

5.4.1 Quadratische Verfahren

Ein quadratisches Verfahren verwendet eine Inkrementfunktion $F(x, y, h)$, welche

$$y(x + h) = y(x) + hF(x, y, h) + O(h^3)$$

erfüllt. Aus den einleitenden Bemerkungen von 5.4 folgt, dass dieses Ziel möglicherweise dadurch erreicht werden kann, dass man Werte von f für verschiedene x geeignet miteinander kombiniert.

Ein denkbarer Ansatz dafür ist

$$F(x, y, h) = af(x, y) + bf(x + ah, y + \beta hf(x, y)),$$

oder anders ausgedrückt: Man führt zuerst etwas Ähnliches wie einen Euler-Schritt durch, um zum Punkt $(x + ah, y + \beta hf(x, y))$ zu gelangen. Dort berechnet man den Wert von f und bildet dann einen geeigneten Mittelwert davon mit $f(x, y)$. Durch geeignete Wahl von a , b , α und β sollte es möglich sein, dass die Inkrementfunktion einen Fehler höchstens dritter Ordnung hat, womit wir dann ein Integrationsverfahren zweiter Ordnung gewonnen hätten.

Wir müssen jetzt die Parameter a , b , α und β bestimmen. Da wir mit dem Übereinstimmen der ersten zwei Ableitungen nur zwei Bedingungen haben, können wir nicht erwarten, dass wir eine eindeutige Lösung finden werden. Vielmehr werden einzelne Parameter frei wählbar sein, es wird eine ganze Familie von quadratischen Lösungsverfahren entstehen, parametrisiert durch eine der Variablen a , b , α und β .

Wir berechnen nun $F(x, y, h)$ bis zur zweiten Ordnung, damit wird $y(x+h)$ bis zur dritten Ordnung ausdrücken können:

$$\begin{aligned} f(x + ah, y + \beta hf(x, y)) &= f(x, y) + ah \frac{\partial f(x, y)}{\partial x} + \beta h \frac{\partial f(x, y)}{\partial y} + O(h^2) \\ F(x, y, h) &= af(x, y) + bf(x + ah, y + \beta hf(x, y)) \\ &= (a + b)f(x, y) + \left(\alpha b \frac{\partial f(x, y)}{\partial x} + \beta b \frac{\partial f(x, y)}{\partial y} f(x, y) \right) h + O(h^2). \end{aligned} \quad (5.22)$$

Damit dies bis zur zweiten Ordnung mit dem Inkrement zwischen x und $x + h$ übereinstimmt, muss (5.22) mit der Taylor-Reihe von $y(x)$ übereinstimmen, also mit

$$\frac{y(x + h) - y(x)}{h} = y'(x) + \frac{1}{2} y''(x)h + O(h^2) = f(x, y) + \frac{1}{2} \frac{\partial f(x, y)}{\partial x} + \frac{1}{2} \frac{\partial f(x, y)}{\partial y} f(x, y) + O(h^2), \quad (5.23)$$

wobei wir für $y''(x)$ die Gleichung (5.6) verwendet haben. Durch Koeffizientenvergleich finden wir die Bedingungen

$$a + b = 1, \quad \alpha b = \frac{1}{2}, \quad \beta b = \frac{1}{2}.$$

Einzig b kommt in allen drei Gleichungen vor und bestimmt den Wert der jeweiligen anderen Variablen:

$$a = 1 - b, \quad \alpha = \beta = \frac{1}{2b}.$$

Jeder Wert von b zwischen 0 und 1 liefert ein Verfahren mit quadratischer Genauigkeit.

Der Parameterwert $b = 1$ führt auf $\alpha = \beta = 1$ und $a = 0$, die Rekursionsformel ist in diesem Falle

$$y_{k+1} = y_k + hf\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hf(x_k, y_k)\right). \quad (5.24)$$

Das Verfahren führt also erst einen halben Euler-Schritt zum Punkt $(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hf(x_k, y_k))$ durch, berechnet dort mit Hilfe von f die Steigung, die dann für einen Euler-Schritt der Länge h verwendet wird (Abbildung 5.5 links). Daher heisst dieses Verfahren auch das *verbesserte Euler-Verfahren*.

Verwendet man $b = \frac{1}{2}$, folgt zunächst $a = \frac{1}{2}$ und $\alpha = \beta = 1$. Daraus erhält man die Rekursionsformel

$$y_{k+1} = y_k + \frac{h}{2} \left(f(x_k, y_k) + f(x_k + h, y_k + hf(x_k, y_k)) \right). \quad (5.25)$$

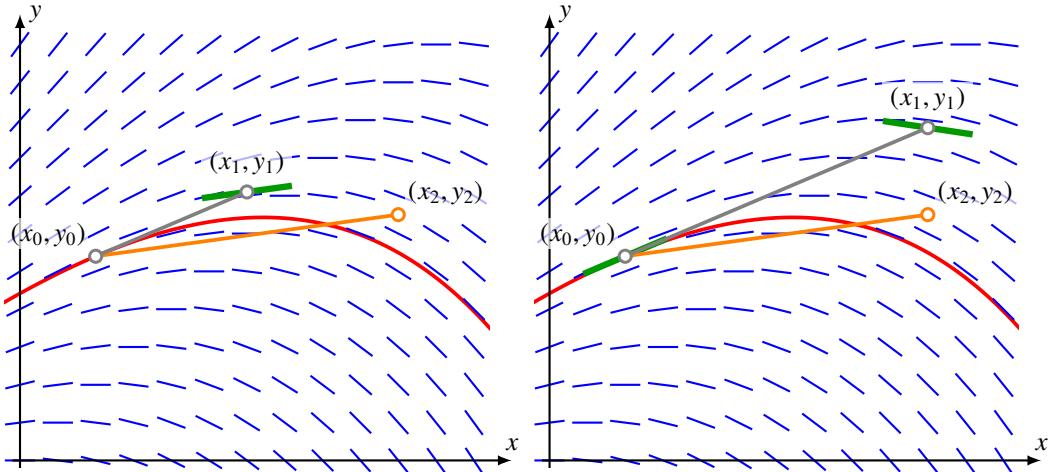


Abbildung 5.5: Einschrittverfahren zweiter Ordnung. Das verbesserte Euler-Verfahren (links) führt zunächst einen Euler-Schritt halber Länge mit der Steigung $f(x_0, y_0)$ (grau) aus und verwendet die bei (x_1, y_1) gefundene Steigung $f(x_1, y_1)$ (grün) für einen ganzen Euler-Schritt (orange), der zu (x_2, y_2) führt. Das vereinfachte Runge-Kutta-Verfahren (rechts) führt einen ganzen Euler-Schritt (grau) aus, der zu (x_1, y_1) führt. Dann wird der Mittelwert der Steigungen $f(x_0, y_0)$ und $f(x_1, y_1)$ (grün) für einen ganzen Euler-Schritt verwendet, der zu (x_2, y_2) führt (orange).

In diesem Verfahren (Abbildung 5.5 rechts) führt man also zuerst einen Euler-Schritt der Länge h durch, mit dem man zum Punkt $(x_k + h, y_k + hf(x_k, y_k))$ gelangt. Dort berechnet mit Hilfe von f die Steigung. Das arithmetische Mittel dieser Steigung mit der im Euler-Verfahren verwendeten Steigung $f(x_k, y_k)$ im Punkt x_k wird dann als Steigung für einen Euler-Schritt verwendet. Statt eines einzigen Steigungswertes werden hier also zwei Steigungswerte von den Enden des Intervalls $[x_k, x_{k+1}]$ gemittelt. Wegen der Ähnlichkeit dieses Vorgehens mit dem später zu besprechenden Runge-Kutta-Verfahren heisst diese Verfahren auch das *vereinfachte Runge-Kutta-Verfahren*.

5.4.2 Runge-Kutta-Verfahren

Das *Runge-Kutta-Verfahren* erweitert die Inkrementfunktion derart, dass der Einzelschritt bis zur fünften Ordnung mit der Taylor-Reihe von $y(x)$ übereinstimmt. So entsteht ein Verfahren vierter Ordnung, es stellt einen guten Kompromiss zwischen Genauigkeit und Rechenaufwand dar.

Da vier Ableitungen korrekt dargestellt werden müssen, ist zu erwarten, dass vier verschiedene Werte von f an verschiedenen Punkten (x, y) ausgewertet und geeignet miteinander kombiniert werden müssen. Genauer: Man bestimmt zuerst die Werte

$$\begin{aligned} k_1 &= f(x_k, y_k) \\ k_2 &= f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2}k_1\right) \\ k_3 &= f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2}k_2\right) \\ k_4 &= f(x_k + h, y_k + hk_3) \end{aligned}$$

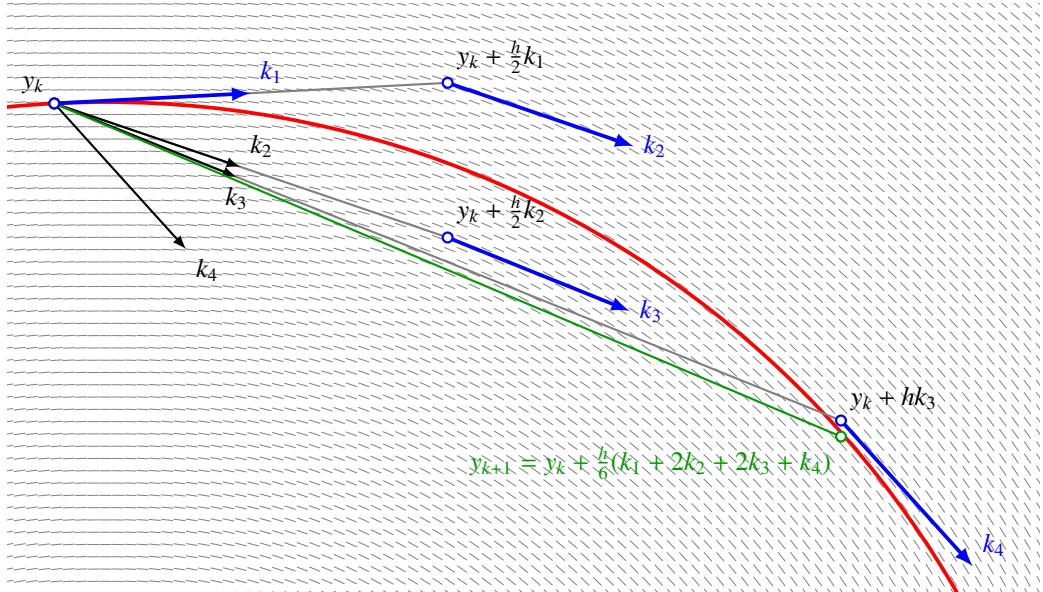


Abbildung 5.6: Zusammenspiel der Richtungen k_1 bis k_4 bei einem Einzelschritt des Runge-Kutta-Verfahrens vierter Ordnung. Der nächste Punkt y_{k+1} wird gemäss Formel (5.26) berechnet.

und setzt diese dann zusammen, um den nächsten Wert y_{k+1} zu berechnen:

$$y_{k+1} = y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (5.26)$$

(Abbildung 5.6). Man kann die Formeln wie folgt interpretieren.

1. Zuerst wird ein halber Euler-Schritt mit der Steigung $k_1 = f(x_k, y_k)$ durchgeführt und am Zielpunkt die Steigung k_2 ermittelt.
2. Mit dieser Steigung k_2 wird dann erneut ein halber Schritt von (x_k, y_k) aus durchgeführt und am Zielpunkt erneut die Steigung k_3 ermittelt.
3. Mit k_3 führt man einen ganzen Schritt aus, an dessen Zielpunkt man die Steigung k_4 findet.
4. Diese vier Steigungen werden jetzt gewichtet gemittelt, wobei k_2 und k_3 doppeltes Gewicht erhalten, und mit dieser Steigung wird ein ganzer Schritt vorgenommen.

Die Formeln für die k_i sowie (5.26) können ganz ähnlich wie das verbesserte Euler-Verfahren bzw. das vereinfachte Runge-Kutta-Verfahren begründet werden. Der Aufwand dafür ist aber beträchtlich, so dass wir auf die detaillierte Darstellung dieser Herleitung verzichten wollen.

Die Tabelle 5.2 demonstriert die überragende Genauigkeit des Runge-Kutta-Verfahrens. Trotz der relativ grossen Schrittweite von $h = 0.1$ erreicht das Verfahren nach zehn Schritten eine Genauigkeit von sieben signifikanten Stellen. Da in jedem Schritt die Funktion f viermal ausgewertet werden muss, ist der Rechenaufwand mit dem Runge-Kutta-Verfahren viermal grösser als im Euler-Verfahren, letzteres kann aber mit nur einer signifikanten Stelle kaum als brauchbar bezeichnet werden. Passt man in jedem Verfahren die Schrittweite so an, dass für die Berechnung der Näherung für

i	x	$y(x) = e^{-\alpha x}$	Euler	verbessert	vereinfacht	Runge-Kutta
0	0.0	1.00000000	1.000	1.00000000	1.00000000	1.00000000000
1	0.1	0.95122942	0.950	0.95125000	0.95125000	0.9512294271
2	0.2	0.90483742	0.902	0.90487656	0.90487656	0.9048374229
3	0.3	0.86070798	0.857	0.86076383	0.86076383	0.8607079834
4	0.4	0.81873075	0.814	0.81880159	0.81880159	0.8187307620
5	0.5	0.77880078	0.773	0.77888502	0.77888502	0.7788007936
6	0.6	0.74081822	0.735	0.74091437	0.74091437	0.7408182327
7	0.7	0.70468809	0.698	0.70479480	0.70479480	0.7046881031
8	0.8	0.67032005	0.663	0.67043605	0.67043605	0.6703200606
9	0.9	0.63762815	0.630	0.63775229	0.63775229	0.6376281672
10	1.0	0.60653066	0.598	0.60666187	0.60666187	0.6065306762

Tabelle 5.2: Vergleich der Genauigkeit der verbesserten numerischen Verfahren. Unterstrichen jeweils die nach Rundung korrekten Stellen der Lösung.

Verfahren	h	Schritte	y_n	Fehler
Euler-Verfahren	0.025	40	0.60462232	0.00190834
verbessertes Euler-Verfahren	0.05	20	0.60656285	-0.00003219
vereinfachtes Runge-Kutta-Verfahren	0.05	20	0.60656285	-0.00003219
Runge-Kutta-Verfahren	0.1	10	0.60653067	-0.00000001

Tabelle 5.3: Vergleich der verschiedenen Verfahren bei gleichbleibendem Rechenaufwand. Die Schrittweite wurde jeweils so angepasst, dass in allen Verfahren bis zum Wert $x = 1$ die gleiche Anzahl von Auswertungen der Funktion f notwendig wurde.

$y(1)$ immer gleich viele Auswertungen der Funktion $f(x, y)$ nötig sind, ergeben sich die Resultate in Tabelle 5.3. Bei gleichem Rechenaufwand ist das Runge-Kutta-Verfahren um viele Größenordnungen präziser. Es gibt daher eigentlich keinen praktischen Grund, überhaupt je etwas anderes als das Runge-Kutta-Verfahren zu verwenden.

5.5 Mehrschrittverfahren

In den Einschrittverfahren wurde wiederholt die Funktion f ausgewertet, um die Inkrementfunktion für einen einzigen Schritt zu bestimmen. Das Ziel dabei war, $y(x + h)$ in Übereinstimmung mit der Taylor-Reihe bis zu möglichst hoher Ordnung zu bestimmen. Im Runge-Kutta-Verfahren wurden dabei halbe Euler-Schritte durchgeführt, man hat also eigentlich die Auflösung nochmals halbiert, um die Inkrementfunktion zu ermitteln. Diese Zwischenwerte geben dem Verfahren die Information über die höheren Ableitungen der Funktionen.

Sobald einige Werte der Lösung berechnet sind, lässt sich die Krümmung der Lösungskurve auch aus diesen Werten ablesen. Es sollte daher auch möglich sein, aus mehreren bereits ermittelten Werten $y_n, y_{n+1}, \dots, y_{n+s-1}$ den nächsten Wert y_{n+s} mit der verlangten Genauigkeit zu berechnen. Der Vorteil eines solchen Vorgehens ist, dass für jeden Schritt nur eine einzige Auswertung der Funktion f nötig ist, nicht mehrere wie bei den besprochenen Einschrittverfahren.

Mehrschrittverfahren sind immer dann die einzige Möglichkeit, wenn Daten nur zu festen Zeitpunkten entlang der Zeitachse zur Verfügung stehen, so dass es die Option, einen “halben Schritt”

durchzuführen also gar nicht gibt. Diese Situation tritt zum Beispiel dann ein, wenn man für die Differentialgleichung wesentliche Daten messen muss und der Sensor dies nur in diskreten Zeitabständen tun kann. Oder wenn Messungen nur zu vorgegebenen Zeitpunkten zur Verfügung stehen, wie zum Beispiel die Positionsmessungen der Planeten Saturn und Uranus, aus deren Abweichungen Urbain Le Verrier und John Crouch Adams die Position des noch unbekannten Planeten Neptun bestimmt und damit dessen Entdeckung ermöglicht haben. Das unten vorgestellte Verfahren stammt unter anderen von Adams.

5.5.1 Ein Verfahren zweiter Ordnung

Als Beispiel versuchen wir daher ein Verfahren aufzubauen, welches y_{n+2} aus den bereits berechneten Werten y_n und y_{n+1} berechnet. Wir nehmen dabei an, dass y_n und y_{n+1} exakt sind. Der neue Datenpunkt soll mit Hilfe eines Ausdrucks der Form

$$y_{n+2} = y_{n+1} + h(af(x_{n+1}, y_{n+1}) + bf(x_n, y_n)) \quad (5.27)$$

gefunden werden. Die Näherung kann wieder mit Hilfe der Ableitungen alleine durch Werte bei x_{n+1} ausgedrückt werden:

$$\begin{aligned} y_{n+2} &= y_{n+1} + h(af(x_{n+1}, y_{n+1}) + bf(x_{n+1} - h, y_n)) \\ &= y_{n+1} + ha f(x_{n+1}, y_{n+1}) + hb f(x_{n+1} - h, y_{n+1} - hf(x_{n+1}, y_{n+1}) + O(h^2)) \\ &= y_{n+1} + ha f(x_{n+1}, y_{n+1}) + hb \left(f(x_{n+1}, y_{n+1}) - h \frac{\partial f(x_{n+1}, y_{n+1})}{\partial x} \right. \\ &\quad \left. - h \frac{\partial f(x_{n+1}, y_{n+1})}{\partial y} f(x_{n+1}, y_{n+1}) + \frac{\partial f(x_{n+1}, y_{n+1})}{\partial y} O(h^2) \right) \\ &= y_{n+1} + (a + b)hf(x_{n+1}, y_{n+1}) - bh^2 \left(\frac{\partial f(x_{n+1}, y_{n+1})}{\partial x} + \frac{\partial f(x_{n+1}, y_{n+1})}{\partial y} f(x_{n+1}, y_{n+1}) + O(h^3) \right). \end{aligned}$$

Sie muss bis zur zweiten Ordnung mit der Taylor-Reihe übereinstimmen:

$$\begin{aligned} y(x_{n+2}) &= y_{n+1} + hy'(x_{n+1}) + \frac{1}{2}h^2y''(x_{n+1}) + O(h^3) \\ &= y_{n+1} + hf(x_{n+1}, y_{n+1}) + \frac{1}{2}h^2 \left(\frac{\partial f(x_{n+1}, y_{n+1})}{\partial x} + \frac{\partial f(x_{n+1}, y_{n+1})}{\partial y} f(x_{n+1}, y_{n+1}) \right). \end{aligned}$$

Vergleicht man Koeffizienten, findet man

$$a + b = 1 \quad -b = \frac{1}{2} \quad \Rightarrow \quad a = \frac{3}{2}.$$

Aus der Formel (5.27) wird somit die Iterationsformel

$$y_{n+2} = y_{n+1} + h \left(\frac{3}{2}f(x_{n+1}, y_{n+1}) - \frac{1}{2}f(x_n, y_n) \right). \quad (5.28)$$

Diese Rekursionsformel definiert ein quadratisches Verfahren, das *Adams-Basforth-Verfahren* mit $s = 2$.

5.5.2 Vergleich mit der Taylor-Reihe

Man kann das Mehrschrittverfahren (5.28) auch als eine Verbesserung des Euler-Verfahrens sehen. Dazu schreiben wir (5.28) als

$$y_{n+2} = y_{n+1} + hf(x_{n+1}, y_{n+1}) + \frac{h}{2} (f(x_{n+1}, y_{n+1}) - f(x_n, y_n)).$$

Die ersten zwei Terme sind vom Euler-Verfahren her bekannt, der dritte Term ist eine Approximation für die zweite Ableitung

$$\begin{aligned} &\approx y(x_{n+1}) + hy'(x_{n+1}) + \frac{h^2}{2!} \cdot \frac{y'(x_{n+1}) - y'(x_n)}{h} \\ &\approx y(x_{n+1}) + hy'(x_{n+1}) + \frac{h^2}{2!} y''(x_{n+1}). \end{aligned}$$

Der dritte Term ist also eine Näherung für den Term der Ordnung 2 der Taylor-Reihe.

5.5.3 Höhere Ordnung

Das Verfahren kann ähnlich wie das Runge-Kutta-Verfahren auf höhere Ordnung erweitert werden. Man findet nach einiger Rechnung

$$\begin{aligned} s = 1: \quad &y_{n+1} = y_n + hf(x_n, y_n), \\ s = 2: \quad &y_{n+2} = y_{n+1} + h\left(\frac{3}{2}f(x_{n+1}, y_{n+1}) - \frac{1}{2}f(x_n, y_n)\right), \\ s = 3: \quad &y_{n+3} = y_{n+2} + h\left(\frac{23}{12}f(x_{n+2}, y_{n+2}) - \frac{4}{3}f(x_{n+1}, y_{n+1}) + \frac{5}{12}f(x_n, y_n)\right), \\ s = 4: \quad &y_{n+4} = y_{n+3} + h\left(\frac{55}{24}f(x_{n+3}, y_{n+3}) - \frac{59}{24}f(x_{n+2}, y_{n+2}) + \frac{37}{24}f(x_{n+1}, y_{n+1}) - \frac{3}{8}f(x_n, y_n)\right). \end{aligned}$$

Ordnung 1 ist natürlich das Euler-Verfahren. Es ist somit möglich, ausgehend von dieser Idee Verfahren beliebig hoher Ordnung zu produzieren.

In der Tabelle 5.4 wird das Adams-Bashforth-Verfahren verglichen mit dem lineare Euler-Verfahren und dem Verfahren vierter Ordnung von Runge-Kutta. Die Verbesserung der Genauigkeit des Adams-Bashforth-Verfahrens gegenüber dem Euler-Verfahren ist konsistent damit, dass das Adams-Bashforth-Verfahren ein quadratisches Verfahren ist.

Nachteilig an den Mehrschrittverfahren ist die Notwendigkeit, genügend viele Werte y_n, \dots, y_{n+s-1} mit ausreichend hoher Genauigkeit zu bestimmen, bevor das Mehrschrittverfahren seine Schritte der Ordnung s beginnen kann. Solange diese Werte nicht zur Verfügung stehen, kann ein Mehrschrittverfahren nur Schritte niedrigerer Ordnung als s durchführen.

Bei einem Einschrittverfahren kann in jedem Schritt die Schrittweite h verändert werden, zum Beispiel für Bereiche von x -Werten, in denen die Steigung von $y(x)$ sehr rasch ändert.

5.5.4 Fehleranalyse für das Adams-Bashford-Verfahren zweiter Ordnung

Für die Beispiel-Differentialgleichung (5.16) können wir das Adams-Bashforth-Verfahren zweiter Ordnung ($s = 2$) vollständig analysieren. Die Rekursionsformel wird zu

$$y_{n+2} = y_{n+1} + h\left(\frac{3}{2}(-\alpha y_{n+1}) - \frac{1}{2}(-\alpha y_n)\right) = \left(1 - \frac{3}{2}\alpha h\right)y_{n+1} + \frac{\alpha h}{2}y_n. \quad (5.29)$$

i	x	$y(x) = e^{-\alpha x}$	Euler	Adams-Bashforth	Runge-Kutta
0	0.0	1.00000000	1.00000000	1.00000000	1.000000000000
1	0.1	0.95122942	0.95000000	0.95128178	0.9512294271
2	0.2	0.90483742	0.90250000	0.90493564	0.9048374229
3	0.3	0.86070798	0.85737500	0.86084752	0.8607079834
4	0.4	0.81873075	0.81450625	0.81890734	0.8187307620
5	0.5	0.77880078	0.77378094	0.77901048	0.7788007936
6	0.6	0.74081822	0.73509189	0.74105738	0.7408182327
7	0.7	0.70468809	0.69833730	0.70495334	0.7046881031
8	0.8	0.67032005	0.66342043	0.67060827	0.6703200606
9	0.9	0.63762815	0.63024941	0.63793648	0.6376281672
10	1.0	0.60653066	0.59873694	0.60685645	0.6065306762

Tabelle 5.4: Vergleich der Genauigkeit der Verfahren von Euler, Adams-Bashforth und Runge-Kutta. Als Startwerte für das Adams-Bashforth-Verfahren wurden die Werte $y(-h) = e^{-\alpha h}$ und $y(0) = 1$ verwendet, um keine zusätzlichen Fehler aus der Durchführung des ersten Schrittes hinzuzufügen.

Dies ist eine Differenzengleichung mit konstanten Koeffizienten, man kann sie mit Hilfe eines Potenzansatzes lösen. Wir nehmen also an, dass $y_n = \lambda^n$, und setzen dies in die Rekursionsformel (5.29) ein. Ausserdem kürzen wir $\alpha h/2 = \delta$ ab. Wir erhalten

$$\lambda^{n+2} - (1 - 3\delta)\lambda^{n+1} - \delta\lambda^n = 0.$$

Nach Division durch λ^n erhalten wir die quadratische Gleichung

$$\lambda^2 - (1 - 3\delta)\lambda - \delta = 0$$

für λ mit den Lösungen

$$\lambda_{\pm} = \frac{1}{2}(1 - 3\delta) \pm \frac{1}{2}\sqrt{(1 - 3\delta)^2 + 4\delta}.$$

Da δ klein ist, wird λ_- ebenfalls klein sein, während λ_+ näher bei 1 sein wird. Der dominante Einfluss auf die Lösung röhrt also von λ_+ her. Um diesen Unterschied genauer zu verstehen, verwenden wir eine lineare Approximation der Wurzel auf der rechten Seite von λ_{\pm} :

$$\begin{aligned} \sqrt{1+x} &= 1 + \frac{x}{2} - \frac{x^2}{4} + \frac{3x^3}{8} - \dots \\ \sqrt{x} &= \sqrt{x_0 + x - x_0} = \sqrt{x_0} \sqrt{1 + \frac{x - x_0}{x_0}} = \sqrt{x_0} \left(1 + \frac{1}{2} \frac{x - x_0}{x_0} - \frac{1}{4} \frac{(x - x_0)^2}{x_0^2} + \dots\right) \\ &= \sqrt{x_0} + \frac{1}{2} \frac{x - x_0}{\sqrt{x_0}} - \frac{1}{4} \frac{(x - x_0)^2}{\sqrt{x_0^3}} + \dots \end{aligned}$$

Wir verwenden diese Approximation mit $x_0 = (1 - 3\delta)^2$ und $x - x_0 = -4\delta$

$$\begin{aligned} \sqrt{(1 - 3\delta)^2 + 4\delta} &= (1 - 3\delta) \left(1 + \frac{1}{2} \frac{4\delta}{(1 - 3\delta)^2} - \frac{1}{4} \frac{16\delta^2}{(1 - 3\delta)^4} + \dots\right) \\ &= (1 - 3\delta) + \frac{1}{2} \frac{4\delta}{1 - 3\delta} - \frac{1}{4} \frac{16\delta^2}{(1 - 3\delta)^3} + \dots \end{aligned}$$

$$\begin{aligned}
&= 1 - 3\delta + 2\delta(1 + 3\delta) - 4\delta^2 + O(\delta^3) \\
&= 1 - 3\delta + 2\delta + 2\delta^2 + O(\delta^3) \\
&= 1 - 3\delta + 2\delta + \frac{1}{2}(2\delta)^2 + O(\delta^3).
\end{aligned}$$

Damit können wir jetzt λ_+ bis zur zweiten Ordnung berechnen:

$$\begin{aligned}
\lambda_+ &= \frac{1}{2} \left((1 - 3\delta) + (1 - 3\delta) + 2\delta + \frac{1}{2}(2\delta)^2 \right) + O(\delta^3) \\
&= 1 - 2\delta + \frac{1}{2}(2\delta)^2 + O(\delta^3) \\
&= e^{-2\delta} + O(\delta^3).
\end{aligned}$$

Die exakte Lösung erfüllt $y_{n+1} = e^{-2\delta}y_n$, der Faktor λ_+ stimmt bis auf Terme mindestens dritter Ordnung mit $e^{-2\delta}$ überein. Damit ist erneut bestätigt, dass wir es mit einem quadratischen Verfahren zu tun haben.

Wir können auch λ_- berechnen und erhalten

$$\lambda_- = -\delta - 2\delta^2 + O(\delta^3).$$

Da δ klein ist, ist eine Komponente der Lösung bereits nach drei Schritten kleiner als $O(\delta^3)$ und spielt daher im Vergleich zu den von λ_+ herrührenden Lösungen in dritter Ordnung keine Rolle.

5.6 Software

Die im letzten Abschnitt entwickelten numerischen Verfahren zur Lösung einer Differentialgleichung kommen ausschliesslich mit Auswertungen der Funktion f aus, die Ableitungen der Funktion f müssen nicht bekannt sein. Es sollte also ein Leichtes sein, eine Softwarebibliothek zur Verfügung zu stellen, mit der eine beliebige gewöhnliche Differentialgleichung gelöst werden kann. Als Input braucht es nur die Funktion f und die Anfangsbedingungen.

Als Beispiel wollen wir in diesem Abschnitt die Differentialgleichung

$$y'' + y = \sin \frac{x}{10}, \quad y(0) = y'(0) = 0$$

in verschiedenen Programmierumgebungen lösen. Als erstes bringen wir die Differentialgleichung wieder in die Standardform einer Vektordifferentialgleichung erste Ordnung:

$$Y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad \Rightarrow \quad \frac{d}{dt} Y = \frac{d}{dx} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -y_1 + \sin \frac{x}{10} \end{pmatrix} = f(x, Y). \quad (5.30)$$

Ein numerisches Verfahren braucht also als Input eine Anfangsbedingung sowie die Funktion f . Außerdem muss es Möglichkeiten bereitstellen, wie man den Gang der Rechnung beeinflussen kann, z. B. um die x -Werte anzugeben, für die die $Y(x)$ ausgegeben werden sollen, oder um Genauigkeitsziele vorzugeben.

5.6.1 Octave

In Octave steht eine einzige Funktion `lsode` zur Verfügung, welche auf zuverlässige Art Differentialgleichungen löst. Der Anwender muss eine Implementation der Funktion f zur Verfügung stellen, allerdings werden die Argumente in der umgekehrten Reihenfolge zu der erwartet, die wir in diesem Skript bisher verwendet haben. Für die Beispieldifferentialgleichung (5.30) kann man sie zum Beispiel so definieren:

```

1 || global omega = 0.1;
2 ||
3 || function yprime = f(y, x)
4 ||     global omega;
5 ||     yprime = [ y(2); -y(1) + sin(omega * x) ];
6 || endfunction

```

Beim Aufruf der Funktion `lsode` muss man den *Namen* der Funktion, die Anfangsbedingung, sowie einen Vektoren mit x -Werten, für die man die Lösung ausgegeben haben möchte, als Argumente übergeben. Der erste Wert im x -Vektor muss der x -Wert für die Anfangsbedingung sein, in unserem Fall also 0. Um die Werte von $y(x)$ für x -Werte zu erhalten, die ganzzahlige Vielfache $x_k = k\Delta x$ von Δx sind, $1 \leq k \leq 10$, muss man also die Befehle

```

1 || y0 = zeros(2,1);
2 || x = (0:10) * deltax;
3 || lsode("f", y0, x)

```

ausführen. Als Rückgabewert erhält man eine Matrix, die in jeder Zeile die Werte von $y(x)$ und $y'(x)$ zum entsprechenden Wert von x aus dem x -Argument enthält. Die Resultate sind zusammen mit den Werten der exakten Lösung (5.17) in der Tabelle 5.5 zusammengestellt. Es ist gut erkennbar, wie der Fehler anfänglich langsam ansteigt, dann aber unter Kontrolle bleibt. Die Dokumentation der Funktion `lsode` beschreibt, wie man mit Hilfe von Optionen ihr Verhalten und insbesondere die Grösse der Fehler weiter beeinflussen kann.

Die Schrittänge muss in einem numerischen Verfahren nicht immer gleich gross gewählt werden. Falls die Ableitungen der gesuchte Funktion nur langsam ändern, können bei gleicher Genauigkeit grössere Schritte verwendet werden, um Rechnzeit einzusparen. Die Funktion `lsode` verwendet eine solche Schrittängensteuerung. Mehr Information dazu findet man im Kapitel 13.

5.6.2 GNU Scientific Library

Während Octave dem Benutzer die Wahl eines geeigneten Verfahrens abnimmt und ihm überhaupt wenig Kontrolle über den Gang der Rechnung gibt, kann ein C-Programmierer durch den Einsatz der GNU Scientific Library (GSL) die volle Kontrolle über alle Aspekte der Iteration erhalten. Der Preis ist eine wesentlich höhere Komplexität. Ziel dieses Abschnitts ist, ein einfaches Beispielprogramm zu zeigen, welches als Basis eigener Programme dienen kann. Es verwendet eine Runge-Kutta-Verfahren achter Ordnung.

Die Funktionen zum Lösen von gewöhnlichen Differentialgleichungen der GSL haben alle das Präfix `gsl_odeiv2_`. Zunächst braucht es natürlich wieder eine Implementation der Funktion f . Die GSL übergibt zwei Arrays, im einen findet die Funktion die aktuellen Y -Werte, im anderen soll sie die Werte der Ableitung zurückgeben. Für die Beispieldifferentialgleichung (5.30) sieht der Code wie folgt aus:

```

1 || int f(double x, const double y[], double f[], void *params) {
2 ||     double omega = *(double *)params;
3 ||     f[0] = y[1];

```

x	$y_{\text{numerisch}}(x)$	$y_{\text{exakt}}(x)$	Fehler
0	0.00000000	0.00000000	0.00000000
1	0.00158525	0.00158528	0.00000003
2	0.01090682	0.01090678	0.00000003
3	0.02858723	0.02858716	0.00000007
4	0.04756207	0.04756212	0.00000004
5	0.05957416	0.05957437	0.00000020
6	0.06276426	0.06276444	0.00000018
7	0.06337942	0.06337932	0.00000010
8	0.07002849	0.07002811	0.00000037
9	0.08576626	0.08576594	0.00000032
10	0.10528405	0.10528416	0.00000010
100	0.84661503	0.84661930	0.00000427
1000	-0.55228836	-0.55234514	0.00005678
2000	0.90392063	0.90373523	0.00018540
3000	-0.99018339	-0.99032256	0.00013917
4000	0.75185982	0.75202340	0.00016358
5000	-0.25298074	-0.25252044	0.00046030
6000	-0.30093757	-0.30056348	0.00037408
7000	0.76905079	0.76889872	0.00015207
8000	-1.00327790	-1.00396748	0.00068958
9000	0.88860437	0.88793099	0.00067338
10000	-0.50337856	-0.50335983	0.00001873

Tabelle 5.5: Exakte und numerische Lösung der Beispieldifferentialgleichung berechnet mit der Funktion `lsode` von Octave.

```

4   ||
5   ||      f[1] = -y[0] + sin(omega * x);
6   ||      return GSL_SUCCESS;
}

```

Der Parameter `params` dient dazu, der Funktion zusätzliche Parameter zu übergeben. In unserem Fall ist das nur die Zahl ω . Da `params` ein void-Pointer ist, kann eine beliebige Struktur zur Parameterübergabe verwendet werden.

Die Differentialgleichung wird beschrieben durch eine Struktur vom Typ `gsl_odeiv2_system`, welche ausser Zeigern auf die Funktion und die Parameter-Struktur auch noch die Dimension der Vektoren enthält. Es kann auch noch ein Funktionszeiger für eine Funktion übergeben werden, die die Jacobi-Matrix berechnet, in unserem Beispiel wird dies jedoch nicht benötigt.

Die eigentliche Berechnung wird von einer “driver”-Funktion durchgeführt. Diese sorgt im wesentlichen für die Wahl der Schrittweite, verwaltet Datenstrukturen und ruft die Funktionen auf, die die einzelnen Schritte durchführen. Die Treiber-Funktion führt die einzelnen Schritte (im Sinne der in Abschnitt 5.4 besprochenen Einschritt-Verfahren) mit Hilfe der Schritt-Funktionen durch, von denen die Bibliothek eine ganze Reihe bereitstellt. Die Funktion `gsl_odeiv2_step_rk4` ist das klassische Runge-Kutta-Verfahren vierter Ordnung, welches in Abschnitt 5.4.2 beschrieben wurde. Im Beispielverfahren verwenden wir `gsl_odeiv2_step_rk8pd`, das Runge-Kutta-Prince-Dormand-Verfahren achter Ordnung. Für Aufgaben allgemeiner Art ebenfalls sehr gut geeignet ist das Runge-Kutta-Fehlberg-Verfahren fünfter Ordnung mit dem Namen `gsl_odeiv2_step_rkf45`. Diese Datenstrukturen werden mit dem Code

```

1 || double omega = 0.1; /* Parameter fuer f */
2 || gsl_odeiv2_system system = { f, NULL, 2, &omega }; /* DG-System */
3 || gsl_odeiv2_driver *driver
4 ||     = gsl_odeiv2_driver_alloc_y_new(&system, gsl_odeiv2_step_rk8pd,
5 ||                                         1e-6, 1e-6, 0.0);

```

initialisiert. Durch Austausch des zweiten Arguments der Driver-Allozierungs-Funktion kann man leicht das Verfahren wechseln und so Zeitaufwand und Genauigkeit für verschiedene Lösungsverfahren vergleichen.

Um die Rechnung durchzuführen, muss jetzt die Driver-Funktion so oft angewendet werden, wie man Punkt der Lösungskurve ausgeben will. Dazu dient die Funktion `gsl_odeiv2_driver_apply`. An Argumenten braucht sie den eben initialisierten Driver, den aktuellen x -Wert, den x_{next} -Wert, für den der nächste Punkt ausgegeben werden soll, sowie einen Vektor, in dem der aktuelle Anfangswert für $Y(x)$ übergeben und $Y(x_{\text{next}})$ zurückgegeben wird. x wird als Referenz übergeben, wenn die Funktion zurückkehrt, findet man dort den neuen aktuellen Wert von x , also im Erfolgsfall x_{next} . In unserem Fall brauchen wir $X(x)$ für ganzzahlige x , die folgende Schleife bewerkstelligt dies:

```

1 || double x = 0.0;
2 || double y[2] = { 0.0, 0.0 };
3 || long lastcounter = evalcounter;
4 || for (int i = 1; i <= 10000; i++) {
5 ||     double xnext = i;
6 ||     int status = gsl_odeiv2_driver_apply(driver, &x, xnext, y);
7 ||     if (status != GSL_SUCCESS) {
8 ||         fprintf(stderr, "error: return value = %d\n", status);
9 ||         return EXIT_FAILURE;
10 ||     }
11 ||     /* Output */
12 ||     ...
13 || }

```

Man kann die Funktion f im Programm natürlich auch mit einem Zähler ausstatten und damit herausfinden, wie viele Aufrufe der Funktion für die numerische Lösung benötigt werden. Es stellt sich heraus, dass für das erste Intervall von 0 bis 1 die Funktion f 131 mal aufgerufen wird, hier versucht die Bibliothek die optimale Schrittweite h zu bestimmen. In allen folgenden Intervallen der Länge 1 von n bis $n + 1$ werden nur noch jeweils 13 Aufrufe der Funktion benötigt. Verwendet man stattdessen das Runge-Kutta-Fehlberg-Verfahren, werden pro Intervall 18 Auswertungen der Funktion f benötigt, und die Genauigkeit sinkt auf zwei Stellen nach dem Komma.

5.7 Randwertprobleme

Die bisher beschriebenen Verfahren gehen von einer Anfangsbedingung aus und berechnen die dadurch eindeutig festgelegte Lösungskurve. Randwertproblem, beschrieben in Abschnitt 5.1.3, verknüpfen dagegen Werte von einzelnen Komponenten von Y an den Rändern eines Intervalls.

5.7.1 Einführende Beispiele

Wir betrachten zwei prototypische Randwertprobleme, die das allgemeine Lösungsverfahren motivieren sollen. In der ersten Aufgabe sind wir dank einer expliziten Form der Lösung nach Einsetzen der Randwertbedingung die Parameter durch Lösen von Gleichungen zu bestimmen.

x	$y_{\text{numerisch}}(x)$	$y_{\text{exakt}}(x)$	Fehler
1	0.01584477	0.01584477	-0.00000000
2	0.10882786	0.10882787	-0.00000000
3	0.28425071	0.28425071	-0.00000001
4	0.46979656	0.46979656	-0.00000000
5	0.58112927	0.58112926	0.00000001
6	0.59856974	0.59856972	0.00000002
7	0.58436266	0.58436265	0.00000001
8	0.62466692	0.62466694	-0.00000001
9	0.74961115	0.74961117	-0.00000003
10	0.90492231	0.90492232	-0.00000001
20	0.82626555	0.82626556	-0.00000000
30	0.24234669	0.24234664	0.00000005
40	-0.83971103	-0.83971092	-0.00000011
50	-0.94210772	-0.94210787	0.00000015
60	-0.25144906	-0.25144893	-0.00000013
70	0.58545210	0.58545205	0.00000005
80	1.09974465	1.09974456	0.00000009
90	0.32597838	0.32597860	-0.00000023
100	-0.49836792	-0.49836823	0.00000031
200	1.01037929	1.01037877	0.00000052
300	-0.89702589	-0.89702630	0.00000041
400	0.83859090	0.83859101	-0.00000010
500	-0.21777633	-0.21777543	-0.00000090
600	-0.31235410	-0.31235239	-0.00000171
700	0.72675906	0.72676124	-0.00000219
800	-1.09422992	-1.09422790	-0.00000202
900	0.80223761	0.80223872	-0.00000111
1000	-0.59500324	-0.59500363	0.00000040
2000	-0.97606658	-0.97606187	-0.00000471
3000	-1.03200392	-1.03199479	-0.00000914
4000	-0.79047800	-0.79047372	-0.00000428
5000	-0.37269297	-0.37270218	0.00000921
6000	0.08785158	0.08783273	0.00001886
7000	0.49827647	0.49826463	0.00001184
8000	0.80219750	0.80220742	-0.00000992
9000	0.94568799	0.94571577	-0.00002778
10000	0.86607968	0.86610200	-0.00002232

Tabelle 5.6: Lösungen der Beispieldifferentialgleichung (5.30) mit Hilfe der GNU Scientific Library (GSL).

Aufgabe 5.5. Mit einem nur der Schwerkraft unterworfenen Ball, der im Ursprung des Koordinatensystems geworfen wird, soll ein Ziel im Punkt P getroffen werden. In welcher Richtung und mit welcher Anfangsgeschwindigkeit muss er geworfen werden?

Um das Problem einfach zu halten, modellieren wir diese Aufgabe wie folgt. Der Ball der Masse m bewegt sich in der x - y -Ebene, wobei die Schwerkraft in negativer y -Richtung zeigt. Das Newtonsche Gesetz liefert die Differentialgleichung zweiter Ordnung

$$m \frac{d^2}{dt^2} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ -mg \end{pmatrix}. \quad (5.31)$$

Die Masse m kann herausgekürzt werden. Gesucht ist eine Lösung so, dass die Bahn durch die Punkte $(0, 0)$ und $P = (p, 0)$ geht.

Genau genommen ist dies nicht ein Randwertproblem wie in Abschnitt 5.1.3, denn es wird nicht verlangt, dass der Ball zu einer bestimmten Zeit t beim Punkt P eintrifft. Die Differentialgleichung bedeutet aber, dass die Horizontalgeschwindigkeit des Balls konstant ist (die horizontale Beschleunigung ist immer 0). Ist v_x die Horizontalgeschwindigkeit, dann erreicht der Ball zur Zeit $t_1 = p/v_x$ die x -Koordinate des Ziels. Gesucht ist also die anfängliche Vertikalgeschwindigkeit, die man dem Ball geben muss, dass zur Zeit p/v_x die y -Komponente der Lösung den Wert 0 hat. In dieser Form liegt ein Randwertproblem wie in Abschnitt 5.1.3 vor.

Die Lösungen der Differentialgleichung 5.31 sind aus dem Physik-Unterricht bekannt: es gilt

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} v_x t \\ v_y t - \frac{1}{2} g t^2 \end{pmatrix} \quad (5.32)$$

Damit lässt sich auch das Randwertproblem lösen. Für $t = v_x/p$ muss $y(t) = 0$ sein, also

$$\begin{aligned} y(t) = y\left(\frac{p}{v_x}\right) &= v_y \frac{p}{v_x} - \frac{1}{2} g \left(\frac{p}{v_x}\right)^2 = 0 \\ \Rightarrow v_y &= \frac{v_x}{p} \frac{1}{2} g \frac{p^2}{v_x^2} = \frac{gp}{2v_x}. \end{aligned} \quad (5.33)$$

Offenbar gibt es zu jedem v_x einen passenden Wert von v_y , mit dem das Ziel getroffen wird.

Die Differentialgleichung (5.31) ist nicht in einer Form, die der numerischen Lösung zugänglich ist. Wir schreiben Sie daher als Differentialgleichung erster Ordnung für vierdimensionale Vektoren:

$$\frac{d}{dt} Y = \frac{d}{dt} \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ 0 \\ -g \end{pmatrix}. \quad (5.34)$$

Gesucht ist eine Lösung, die die Randbedingungen

$$Y(0) = \begin{pmatrix} 0 \\ 0 \\ v_x \\ \textcolor{red}{v_y} \end{pmatrix}, \quad Y\left(\frac{p}{v_x}\right) = \begin{pmatrix} p \\ 0 \\ ? \\ ? \end{pmatrix} \quad (5.35)$$

erfüllt. Darin stehen die roten Einträge für Werte, die nicht vorgegeben sind. Aus der Symmetrie des Problems kann man natürlich auch die Endgeschwindigkeit ablesen. Zu bestimmen ist also v_y so, dass die Lösungskurve durch den Punkt $(p, 0)$ geht.

Wird statt der Horizontalkomponenten der Anfangsgeschwindigkeit die gesamte Anfangsgeschwindigkeit v_0 vorgegeben, dann muss der Winkel gefunden werden, unter dem der Ball geworfen werden muss, um das Ziel zu treffen. Bei der Elevation α sind die Komponenten der Anfangsgeschwindigkeit $v_x = v_0 \cos \alpha$ und $v_y = v_0 \sin \alpha$. Setzt man dies in die Bedingung (5.33) ein, findet man

$$\begin{aligned}v_0 \sin \alpha &= \frac{gp}{2v_0 \cos \alpha} \\2 \sin \alpha \cos \alpha &= \frac{gp}{v_0^2} \\\sin 2\alpha &= \frac{gp}{v_0^2} \\\alpha &= \frac{1}{2} \arcsin \frac{gp}{v_0^2}.\end{aligned}$$

Im Nenner rechts steht im Wesentlichen die kinetische Energie, je mehr kinetische Energie der Ball zu Beginn hat, desto kleiner ist der Winkel, man trifft das Ziel mit einer sehr flachen Bahn. Kleine Winkel reichen auch für geringe Schwerkraft (g klein) und kurze Distanzen (p klein). Die maximale Distanz wird erreicht, wenn das Argument des Arcussinus den Wert 1 erreicht, grösser darf p nicht werden, weil es sonst keine Lösung mehr für α gibt. Die Maximaldistanz ist daher

$$p_{\max} = \frac{v_0^2}{g}.$$

Das Beispiel hat gezeigt, dass sich das Randwertproblem lösen lässt, indem man unter den Lösungen des zugehörigen Anfangswertproblems diejenigen Anfangswerte auswählt, die alle Randbedingungen erfüllen. Zusätzlich zum Anfangswertproblem muss man also in der Lage sein, eine Gleichung für die Anfangswerte zu lösen. Im vorangegangenen Beispiel ging dies algebraisch, das folgende Beispiel zeigt, wie dies numerisch möglich ist.

Aufgabe 5.6. Ein Seil ist zwischen zwei Punkten aufgehängt, welche Form nimmt es allein unter der Wirkung seines Eigengewichtes an?

Die Lösungskurve dieses Problems heisst die *Kettenlinie*. Auch in diesem Fall können wir wieder eine Lösungsfunktion $y(x)$ finden, aber die Bestimmung der Parameter ist jetzt nicht mehr in geschlossener Form möglich. Wir behelfen uns mit einer numerischen Lösung.

Lösung. Zunächst brauchen wir eine Differentialgleichung, deren Lösung die gesuchte Kurve beschreibt. Zur Herleitung dient die Abbildung 5.7. Die Masse des Seils zwischen den beiden Punkten x und $x + \Delta x$ wird von den beiden eingezeichneten Kräften getragen. Die horizontalen Komponenten tragen nicht dazu bei, das Seil zu tragen, sie haben daher entlang des ganzen Seils immer die gleiche Grösse F . Die Masse des Seilstücks ist proportional zu seiner Länge, der Proportionalitätsfaktor ist die lineare Massendichte μ . Nach dem dritten Newtonschen Gesetz sind die vertikalen Kraftkomponenten gleich gross wie die Gewichtskraft des Seilstücks:

$$F(y'(x + \Delta x) - y'(x)) = \mu g \sqrt{(y(x + \Delta x) - y(x))^2 + \Delta x^2}$$

oder nach Divison durch $F\Delta x$:

$$\frac{y'(x + \Delta x) - y'(x)}{\Delta x} = \frac{\mu g}{F} \sqrt{1 + \left(\frac{y(x + \Delta x) - y(x)}{\Delta x}\right)^2}.$$

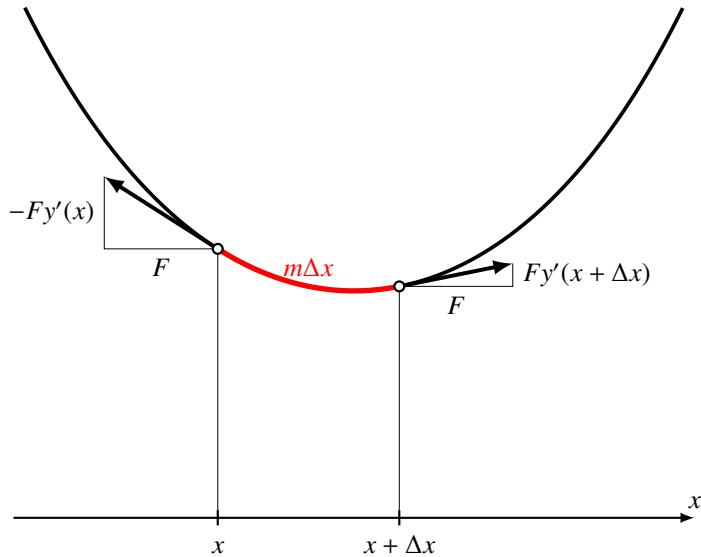


Abbildung 5.7: Herleitung der Differentialgleichung der Kettenlinie

Schreibt man $a = \mu g / F$ und geht zur Grenze $\Delta x \rightarrow 0$ über, erhält man die Differentialgleichung

$$y''(x) = a \sqrt{1 + y'(x)^2}. \quad (5.36)$$

Diese Differentialgleichung hat die Funktion

$$y(x) = \frac{1}{a} \cosh ax + C$$

als Lösung, wie man durch Nachrechnen einsehen kann¹. Die Ableitungen von $y(x)$ sind

$$\begin{aligned} y'(x) &= \sinh ax \\ y''(x) &= a \cosh ax. \end{aligned}$$

Eingesetzt in die Differentialgleichung erhält man

$$a \cosh ax = a \sqrt{1 + \sinh^2 ax},$$

¹Man kann die Gleichung (5.36) natürlich auch direkt lösen. Dazu bestimmt man zuerst die Funktion $z(x) = y'(x)$, welche die Differentialgleichung

$$z'(x) = a \sqrt{1 + z(x)^2}$$

erfüllt. Diese Gleichung lässt sich durch Separation lösen:

$$\frac{dz}{dx} = a \sqrt{1 + z^2} \quad \Rightarrow \quad \frac{1}{a} \int \frac{dz}{\sqrt{1 + z^2}} = \int dx \quad \Rightarrow \quad \frac{1}{a} \operatorname{arsinh} z = x + C_1 \quad \Rightarrow \quad z(x) = \sinh a(x + C_1).$$

Die Funktion $y(x)$ bekommt man jetzt durch Integration

$$y(x) = \int z(x) dx = \frac{1}{a} \cosh a(x + C_1) + C_2,$$

mit $C_1 = -x_0$ und $C_2 = C$ genau die vorgeschlagene Lösung.

was sich zu

$$\cosh^2 ax - \sinh^2 ax = 1$$

umformen lässt, diese Gleichung ist für hyperbolische Funktionen immer erfüllt.

Die Differentialgleichung (5.36) hängt nicht von x ab, also sind auch verschobene Funktionen Lösungen. Die allgemeine Lösung des Problems ist daher die Funktion

$$y(x) = \frac{1}{a} \cosh a(x - x_0) + C. \quad (5.37)$$

Die Bedeutung der Konstante C ist leichter verständlich, wenn man sie als $C = y_0 - 1/a$ schreibt, also

$$y(x) = \frac{1}{a} \cosh a(x - x_0) - \frac{1}{a} + y_0.$$

Setzt man $x = x_0$ ein, findet man $y(x_0) = y_0$, d. h. der Punkt (x_0, y_0) ist der Scheitelpunkt des Graphen von $y(x)$.

Damit kann jetzt das Anfangswertproblem gelöst werden. Wir verlangen, dass die im Punkt $x = x_1$ der Funktionswert $y(x_1) = y_1$ sein soll und die Steigung $y'(x_1) = m$. Setzt man die Lösung (5.37) in die Bedingung für die Steigung ein, erhält man

$$m = \sinh a(x_1 - x_0) \quad \Rightarrow \quad \operatorname{arsinh} m = a(x_1 - x_0) \quad \Rightarrow \quad x_0 = x_1 - \frac{1}{a} \operatorname{arsinh} m.$$

Die Anfangsbedingung für $y(x_1)$ liefert

$$\begin{aligned} y_1 &= y(x_1) = \frac{1}{a} \cosh a(x_1 - x_0) + y_0 - \frac{1}{a} = \frac{1}{a} \cosh \operatorname{arsinh}(m) + y_0 - \frac{1}{a} = \frac{1}{a} \sqrt{1+m^2} + y_0 - \frac{1}{a} \\ \Rightarrow y_0 &= y_1 + \frac{1}{a} - \frac{1}{a} \sqrt{1+m^2}. \end{aligned}$$

Damit ist das Anfangswertproblem vollständig gelöst,

$$y(x) = \frac{1}{a} \cosh(a(x - x_1) + \operatorname{arsinh} m) + y_1 - \frac{1}{a} \sqrt{1+m^2} \quad (5.38)$$

ist die allgemeine Lösung zur Anfangsbedingung $y(x_1) = y_1$, $y'(x_1) = m$.

Mit dieser Lösung kann jetzt auch das Randwertproblem gelöst werden. Es muss ein Wert m gefunden werden, so dass $y(x_2) = y_2$, man muss also die Gleichung

$$y_2 = \frac{1}{a} \cosh(a(x_2 - x_1) + \operatorname{arsinh} m) + y_1 - \frac{1}{a} \sqrt{1+m^2} =: f(m)$$

nach m auflösen. Das ist leider nicht so einfach, weil m in der transzendenten Funktion arsinh auftritt.

Wir können die Gleichung jedoch iterativ mit dem Newton-Verfahren nach Satz 2.5 lösen. Dazu brauchen wir die Ableitung von f , sie ist

$$f'(m) = \frac{\sinh(x_2 - x_1 + \operatorname{arsinh} m) - m}{\sqrt{m^2 + 1}}.$$

Damit können wir die Iterationsformel

$$m_{\text{neu}} = m - \frac{f(m) - y_2}{f'(m)}$$

	m	$f(m)$
1	0.0000000000000000	8.0676619957777653
2	-0.8053266839760436	2.5748454835378896
3	-1.3999183852140562	0.5757860863682827
4	-1.6180720459804230	0.0383682292517731
5	-1.6347219547661627	0.0001812733120350
6	-1.6348013659846743	0.0000000040625958
7	-1.6348013677644737	0.0000000000000004
8	-1.6348013677644739	-0.0000000000000002

Tabelle 5.7: Numerische Lösung des Randwertproblems für die Kettenlinie mit Randbedingungen $x_1 = -1$, $x_2 = 2$, $y(x_1) = y_1$ und $y(x_2) = y_2$.

für den korrekten Wert m finden. Sie kann leicht in Octave implementiert werden, wie Listing 5.1 zeigt.

In Tabelle 5.7 ist der Gang der Berechnung für den Fall $x_1 = -1$, $x_2 = 2$, $y_1 = 1$ und $y_2 = 2$ mit dem Startwert $m = 0$ dargestellt. Wie erwartet ist die Konvergenz quadratisch, in jedem Schritt verdoppelt sich die Anzahl korrekter Stellen. In Abbildung 5.8 sind die Graphen von $y(x)$ zu den im Newton-Algorithmus ermittelten Werten von $m = y'(x_1)$ blau dargestellt, die Lösungskurve ist rot.

○

5.7.2 Schiessverfahren

Wenn man experimentell versucht, ein Ziel zu treffen, dann wird man in wiederholten Versuchen die Richtung anpassen, so dass man dem Ziel immer näher kommt. Genau dies haben wir in der Aufgabe 5.6 gemacht, wo wir iterativ die noch unbekannten Anfangsbedingung $y(x_1) = m$ bestimmt haben, mit der die Lösung durch den rechten Randpunkt $y(x_2) = y_2$ geht.

Auch in der Aufgaben 5.5 konnten wir die Parameter direkt bestimmen. Doch auch dort läuft die Lösung auf die Bestimmung einer geeigneten Anfangsbedingung hinaus. Der y -Wert zur Zeit p/v_x hängt von der Vertikalgeschwindigkeit ab, wir bezeichnen ihn mit $h(v_y)$. Man verändert also v_y , bis die Gleichung $h(v_y) = 0$ erfüllt ist. Um das Randwertproblem zu lösen, muss man also die Gleichung $h(v_y) = 0$ numerisch lösen.

Man kann dies z. B. dadurch machen, dass man nach zwei Werten von v_y sucht, so dass die zum einen gehörige Bahn unter dem Punkt P durchgeht, während der Ball im anderen Fall darüber hinwegfliegt. Durch wiederholte Halbierung des Intervalls kann man dann den korrekten Wert für v_y immer genauer eingrenzen². Der Nachteil dieses Verfahrens ist, dass mit jedem Schritt die Genauigkeit nur um in Bit ansteigt, es sind also sehr viele Iterationen notwendig.

Schnellere Konvergenz kann mit dem Newton-Verfahren für Vektorgleichungen erreicht werden, welches in Satz 2.6 beschrieben wird. Für die Anwendung des Newton-Verfahrens auf das Randwert-Problem ist die Bestimmung der Steigung der Funktion nötig, die die Abweichung der Kurve von der Randbedingung am rechten Rand angibt. Wir müssen also berechnen, wie schnell sich $y(p/v_x)$

²Tatsächlich wird dieses Verfahren in der Artillerie verwendet. Der Schiesskommandant beobachtet die einschlagenden Granaten und kommandiert Änderungen der Anfangs-Elevation an die Geschützbatterien. Dabei sucht er Einschläge, die aus seiner Perspektive vor bzw. hinter dem Ziel liegen, und halbiert dann das Intervall, bis die Einschläge dem Ziel genügend nahe kommen.

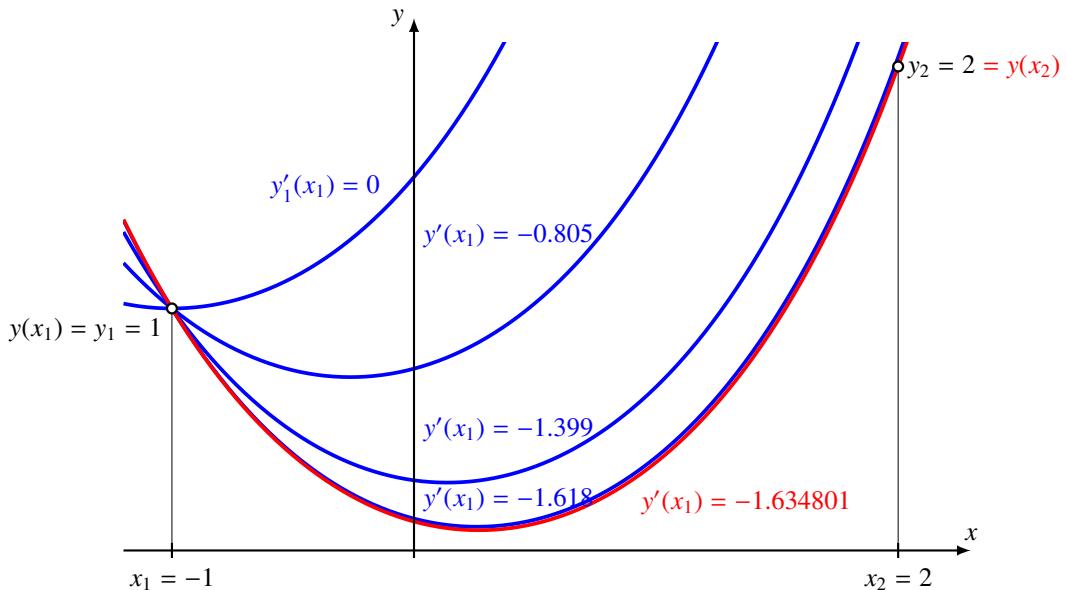


Abbildung 5.8: Iterative Bestimmung der Kettenlinie zwischen den Punkten $(-1, 2)$ und $(2, 2)$ mit Hilfe des Newton-Algorithmus.

```

1 global x1 = -1
2 global x2 = 2
3 global y1 = 1
4 global y2 = 2
5
6 function v = f(m)
7   global x1 x2 y1 y2
8   v = cosh(x2 - x1 + asinh(m)) + y1 - y2 - sqrt(1 + m^2);
9 endfunction
10
11 function v = fprime(m)
12   global x1 x2 y1 y2
13   v = (sinh(x2 - x1 + asinh(m)) - m)/sqrt(1 + m^2);
14 endfunction
15
16 m = 0;
17
18 for i = (1:10)
19   printf("%2d %20.16f %20.16f\n", i, m, f(m));
20   m = m - f(m)/fprime(m);
21 endfor

```

Listing 5.1: Octave-Programm zur Bestimmung der Anfangssteigung m im Kettenlinien-Problem

Abbildung 5.9: Lösungen des Anfangswertproblems (5.34) und (5.35). Das Newton-Verfahren korrigiert v_y derart, dass $h(v_y) = 0$ wird. So wird die Lösung des Randwertproblems (rot) gefunden.

ändert, wenn v_y verändert wird. Dies ist die Ableitung

$$h'(v_y) = \frac{\partial y}{\partial v_y},$$

ein Eintrag der Jacobi-Matrix. In Abschnitt 5.1.5 wurde gezeigt, wie man auch für die Jacobi-Matrix eine Differentialgleichung aufstellen kann, die man natürlich ebenfalls mit den früher beschriebenen numerischen Bibliotheken lösen kann.

Das Randwertproblem kann daher mit folgendem Algorithmus numerisch gelöst werden.

1. Beginne mit einer Schätzung für v_y
2. Finde numerisch die Lösung des Anfangswertproblems mit v_y als anfängliche Vertikalgeschwindigkeit. Berechnet dabei auch die Jacobi-Matrix.
3. Lese die $h(v_y)$ aus der Lösung zur Zeit p/v_x ab und $h'(v_y)$ aus der Jacobi-Matrix und verwendet den Newton-Algorithmus (Satz 2.6), um eine verbesserte Schätzung von v_y zu bekommen.
4. Wiederhole Schritte 2 und 3 bis die Randbedingung für $t = p/v_x$ genügend genau erfüllt ist.
5. Die Lösung des Anfangswertproblems mit diesem v_y ist die Lösung des gestellten Randwertproblems.

Beispiel. Wir führen den eben skizzierten Algorithmus für das Ball-Problem durch. Um die Jacobi-Matrix zu berechnen, müssen wir die Ableitung von f berechnen:

$$\frac{\partial f(x, y)}{\partial y} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (5.39)$$

Da die rechte Seite nicht von y abhängt, können wir die Gleichung für die Jacobi-Matrix ganz unabhängig von y lösen. Da F so einfach ist, kann man das Matrizenprodukt direkt ausrechnen, so wird die Differentialgleichung für J

$$\begin{pmatrix} J'_{11} & J'_{12} & J'_{13} & J'_{14} \\ J'_{21} & J'_{22} & J'_{23} & J'_{24} \\ J'_{31} & J'_{32} & J'_{33} & J'_{34} \\ J'_{41} & J'_{42} & J'_{43} & J'_{44} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ J_{31} & J_{32} & J_{33} & J_{34} \\ J_{41} & J_{42} & J_{43} & J_{44} \end{pmatrix} = \begin{pmatrix} J_{31} & J_{32} & J_{33} & J_{34} \\ J_{41} & J_{42} & J_{43} & J_{44} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (5.40)$$

Daraus kann man ablesen, dass die Elemente J_{3j} und J_{4j} sich nicht ändern, sie bleiben also konstant. Die Werte sind

$$J_{3j}(x) = \delta_{3j} \quad \text{und} \quad J_{4j}(x) = \delta_{4j}.$$

n	v_y	t	$x(t)$	$y(x)$	$\frac{\partial y}{\partial v_y}$	$v_{y,\text{new}}$	Δ
0	7.0000	2.5	20.0	-13.156250	2.5	12.262500000	-5.2625000000
1	12.2625	2.5	20.0	-0.000004	2.5	12.26250145	-0.0000014458
2	12.2625	2.5	20.0	0.000000	2.5	12.26250143	0.0000000204

Tabelle 5.8: Newton-Algorithmus für das Ball-Problem, Resultate der numerischen Rechnung. v_y wird in drei Schritten mit einer Genauigkeit von mehr als 10 Stellen gefunden.

Damit wird die Differentialgleichung reduziert auf

$$\begin{pmatrix} J'_{11} & J'_{12} & J'_{13} & J'_{14} \\ J'_{21} & J'_{22} & J'_{23} & J'_{24} \\ J'_{31} & J'_{32} & J'_{33} & J'_{34} \\ J'_{41} & J'_{42} & J'_{43} & J'_{44} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (5.41)$$

Auch die Funktionen $J_{1j}(x)$ und $J_{2j}(x)$ sind also konstant, einzig die Elemente $J_{13}(x)$ und $J_{24}(x)$ können sich ändern, sie erfüllen die Differentialgleichungen

$$\left. \begin{array}{l} J'_{13}(x) = 1 \\ J'_{24}(x) = 1 \end{array} \right\} \quad \text{mit den Lösungen} \quad \begin{cases} J_{13}(x) = x \\ J_{24}(x) = x. \end{cases}$$

In Matrixform ist die Lösung

$$J(x) = \begin{pmatrix} 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (5.42)$$

Damit haben wir die nötige Information, um den Newton-Algorithmus durchzuführen. In Tabelle 5.8 sind die Resultate der numerischen Rechnung zusammengestellt. Es zeigt sich, dass der korrekte Wert für v_y in drei Iterationen mit 10 Stellen Genauigkeit gefunden werden kann. Damit ist das Randwertproblem numerisch gelöst. \circlearrowright

Übungsaufgaben

5.1. In dieser Aufgabe soll ein implizites Verfahren für die Lösung der Differentialgleichung

$$y' = f(x, y) \quad (5.43)$$

gefunden werden. Gesucht werden also die Funktionswerte y_0, y_1, \dots, y_n zu den x -Werten x_0, x_1, \dots, x_n , wobei wir typischerweise $x_k = x_0 + kh$ wählen.

- Leiten Sie aus der Differentialgleichung (5.43) eine Näherungsgleichung für benachbarte Punkte (x_k, y_k) und (x_{k+1}, y_{k+1}) ab, die beide Steigungen in den Punkten aber keine weiteren Punkte verwendet.
- Wenden Sie dieses Verfahren auf die Gleichung $y' = f(x, y) = y$ an, und drücken Sie y_{k+1} durch $x_k, x_{k+1} = x_k + h$ und y_k aus.

- c) Was erhält man, wenn man für die Differentialgleichung $y' = y$ als Schrittweite $1/n$ wählt und n Schritte ausführt. Wie hängt y_n von y_0 ab? Berechnen Sie den Grenzwert $n \rightarrow \infty$.
- d) Versuchen sie experimentell die Ordnung des Verfahrens zu bestimmen, indem Sie die die Gleichung $y' = y'$ damit lösen und das Verhalten bei Halbierung der Schrittweite untersuchen.

Lösung. a) Die Steigung zwischen den zwei Punkten (x_k, y_k) und (x_{k+1}, y_{k+1}) kann mit den beiden Steigungen $y'_k = f(x_k, y_k)$ und $y'_{k+1} = f(x_{k+1}, y_{k+1})$ verglichen werden. Da keine dieser Steigungen wirklich für das ganze Intervall $[x_k, x_{k+1}]$ repräsentativ sein kann, nehmen wir den Mittelwert, also

$$\frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{1}{2}(f(x_k, y_k) + f(x_{k+1}, y_{k+1})) \quad (5.44)$$

Im Allgemeinen kann man diese Gleichung nicht nach y_{k+1} auflösen.

- b) Setzt man die Differentialgleichung $f(x, y) = y$ ein, erhält man

$$\frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{1}{h}(y_{k+1} - y_k) = \frac{1}{2}(y_k + y_{k+1})$$

was man nach y_{k+1} auflösen kann:

$$2y_{k+1} - 2y_k = hy_k + hy_{k+1} \Rightarrow (2 - h)y_{k+1} = (2 + h)y_k \Rightarrow y_{k+1} = \frac{2+h}{2-h}y_k. \quad (5.45)$$

- c) Mit $h = \frac{1}{n}$ und n Schritten erhält man

$$y_n = y_0 \left(\frac{2 + \frac{1}{n}}{2 - \frac{1}{n}} \right)^n = y_0 \left(\frac{1 + \frac{1}{2n}}{1 - \frac{1}{2n}} \right)^n = y_0 \left(1 + \frac{1}{2n} \right)^n \left(1 + \frac{1}{2n} + \frac{1}{(2n)^2} + \dots \right)^n \simeq y_0 \left(1 + \frac{1}{n} + o\left(\frac{1}{2n}\right) \right)^n \rightarrow y_0 e$$

für $n \rightarrow \infty$. Analog kann man für n Schritte zwischen 0 und t die Formel $y_n = y_0 e^t$ ableiten.

- d) Die Formeln (5.45) erlauben jetzt auch, die Konvergenzgeschwindigkeit zu berechnen. Wir vergleichen dazu die Werte

$$\left(\frac{2 + 2^{-n}}{2 - 2^{-n}} \right)^{2^n} \quad \text{mit} \quad e.$$

Wir erhalten die Werte in Tabelle 5.9, die andeuten, dass ein Verfahren erster Ordnung vorliegt.



5.2. Die Differentialgleichung $y' = e^y$

- a) Berechnen Sie $y(1)$, indem Sie 2^k Euler-Schritte der Länge $h = 2^{-k}$ durchführen mit $k = 0, 1, 2$.
- b) Berechnen Sie $y(1)$, indem Sie 2^k verbesserte Euler-Schritte der Länge $h = 2^{-k}$ machen mit $k = 0, 1$.
- c) Berechnen Sie $y(1)$, indem Sie 2^k vereinfachte Runge-Kutta-Schritte der Länge $h = 2^{-k}$ machen mit $k = 0, 1$.
- d) Berechnen Sie $y(1)$ mit Hilfe eines einzigen Runge-Kutta-Schrittes der Länge $h = 1$.
- e) Finden Sie die exakte Lösung der Differentialgleichung. Was passiert für $x \rightarrow 1$?

k	Wert
1	<u>2.7</u> 77777777777777781
2	<u>2.7</u> 326114119117042
3	<u>2.7</u> 218318928456022
4	<u>2.7</u> 191673488624724
5	<u>2.7</u> 185030842147762
6	<u>2.7</u> 183371346324057
7	<u>2.7</u> 182956545171231
8	<u>2.7</u> 182852849433168
9	<u>2.7</u> 182826925781960
10	<u>2.7</u> 182820444885607
11	<u>2.7</u> 182818824664454
12	<u>2.7</u> 182818419608967
13	<u>2.7</u> 182818318338904
14	<u>2.7</u> 182818293028337
15	<u>2.7</u> 182818286700021
16	<u>2.7</u> 182818285117856
17	<u>2.7</u> 182818285117865
18	<u>2.7</u> 182818284524526
19	<u>2.7</u> 182818284573971
20	<u>2.7</u> 182818284586330
∞	2.71828182845905

Tabelle 5.9: Näherungswerte für $y(1)$ für die Lösung $y(x)$ der Differentialgleichung $y' = y$ mit Anfangswert $y(0) = 1$. Die korrekten Stellen sind unterstrichen. Etwa jede dritte Iteration liefert eine zusätzliche Dezimalstelle Genauigkeit, woraus man ablesen kann, dass ein Verfahren erster Ordnung vorliegt.

f) Verwenden Sie ein Bibliotheksfunktion zur Berechnung der Werte $y(x)$ für $x = 1 - 10^{-k}$.

Lösung. a) Mit einem einzigen Euler-Schritt der Länge 1 erhält man

$$y(1) = y(0) + h \cdot e^0 = y(0) + 1 \cdot 1 = 1.$$

Mit Schritten der Länge $h = \frac{1}{2}$ bekommt man

$$\begin{aligned} y\left(\frac{1}{2}\right) &= y(0) + \frac{1}{2}e^0 = \frac{1}{2} \\ y(1) &= y\left(\frac{1}{2}\right) + \frac{1}{2} \cdot e^{y\left(\frac{1}{2}\right)} = \frac{1}{2} + \frac{1}{2}e^{\frac{1}{2}} = 1.32436063535006407342 \end{aligned}$$

Schliesslich für Schrittweite $h = 0.25$

$$y(0.25) = y(0.00) + 0.25 \cdot e^{y(0.00)} = 0.25$$

$$y(0.50) = y(0.25) + 0.25 \cdot e^{y(0.25)} = 0.57100635417193537101$$

$$y(0.75) = y(0.50) + 0.25 \cdot e^{y(0.50)} = 1.01351821668782980596$$

$$y(1.00) = y(0.75) + 0.25 \cdot e^{y(0.75)} = 1.70233762833485178875$$

k	Euler	Euler verbessert	Runge-Kutta vereinfacht	Runge-Kutta
0	1.00000000	1.64872127	1.85914091	3.10651614
1	1.32436064	2.16998411	2.42505852	3.79915029
2	1.70233763	2.75974208	3.04908191	4.49270245
3	2.12765525	3.39560344	3.70693610	5.18595885
4	2.59253414	4.05855814	4.38244378	5.87912458
5	3.08942706	4.73620728	5.06680668	6.57227443
6	3.61192242	5.42150373	5.75557547	7.26542196
7	4.15492772	6.11070001	6.44653754	7.95856919
8	4.71451145	6.80186535	7.13859327	8.65171638
9	5.28765882	7.49402001	7.83119502	9.34486356
10	5.87204732	8.18667053	8.52406955	10.03801074
11	6.46586955	8.87956928	9.21708043	10.73115792
12	7.06770156	9.57259222	9.91015946	11.42430510
13	7.67640686	10.26567727	10.60327257	12.11745228

Tabelle 5.10: Näherungswerte für $y(1)$ für die Differentialgleichung $y' = e^y$ mit Schritten nach verschiedenen Verfahren der Länge 2^{-k} . Es sind keine Anzeichen von Konvergenz sichtbar.

Es sind keine Anzeichen von Konvergenz sichtbar. In der Tat kann man diese Analyse numerisch auch noch etwas weiter treiben, man findet die Zahlen in Tabelle 5.10

- b) Ein verbesserter Euler-Schritt besteht in einem halben Euler-Schritt zur Ermittlung der Steigung, mit der der Euler-Schritt ausgeführt werden soll:

$$y_{i+1} = y_i + h f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} f(x_i, y_i)\right) = y_i + h e^{y_i + h e^{y_i}/2}$$

Die analoge Rechnung zu a) liefert die Zahlenwerte in Spalte 3 von Tabelle 5.10. Auch in diesem Verfahren ist keine Konvergenz erkennbar.

- c) Ein vereinfachter Runge-Kutta-Schritt verwendet Steigungen am Anfang und nach einem ganzen Euler-Schritt:

$$y_{i+1} = y_i + \frac{h}{2} \left(f(x_i, y_i) + f(x_i + h, y_i + h f(x_i, y_i)) \right) = y_i + \frac{h}{2} (e^{y_i} + e^{y_i + h e^{y_i}}) = y_i + \frac{h}{2} e^{y_i} (1 + e^{h e^{y_i}}).$$

Die numerische Rechnung ergibt die Werte in Spalte 4 von Tabelle 5.10.

- d) Für einen Runge-Kutta-Schritt müssen die Steigungen k_i bestimmt werden

$$\begin{aligned} k_1 &= f(0, 0) = 1 &= 1 \\ k_2 &= f\left(\frac{1}{2}, \frac{1}{2}k_1\right) = e^{\frac{1}{2}} &= 1.64872127070013 \\ k_3 &= f\left(\frac{1}{2}, \frac{1}{2}e^{\frac{1}{2}}\right) = e^{\frac{1}{2}}e^{\frac{1}{2}} &= 2.28042227773497 \\ k_4 &= f(1, k_3) = e^{e^{\frac{1}{2}}e^{\frac{1}{2}}} &= 9.78080975579193 \end{aligned}$$

k	x	numerische Lösung	$-\log(1-x)$
0	0.0000000	0.0000000000000000	0.0000000000000000
1	0.9000000	2.302589125438244	2.302585092994046
2	0.9900000	4.605219624234440	4.605170185988091
3	0.9990000	6.908262594576252	6.907755278982136
4	0.9999000	9.215442117286264	9.210340371976294
5	0.9999900	11.565174615733303	11.512925464974780
6	0.9999990	14.527024809503091	13.815510557935518

Tabelle 5.11: Numerische und exakte Lösung der Differentialgleichung $y' = e^y$ für x -Werte $1 - 10^{-k}$. Die Divergenz der Lösung $y(x)$ für $x \rightarrow 1$ äussert sich in der tatsache, dass das numerische Verfahren (die Funktion `lsode` von Octave) keine konvergente Lösung nahe bei 1 finden kann.

Der Wert von $y(1)$ ist dann

$$y(1) = y_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = \frac{1}{6}(1 + 2e^{\frac{1}{2}} + 2e^{\frac{1}{2}e^{\frac{1}{2}}} + e^{e^{\frac{1}{2}e^{\frac{1}{2}}}}) = 3.10651614211036$$

Die bisher gefundenen Werte für $y(1)$ sind

Verfahren	Schrittweite	$y(1)$
Euler	1	1.0000000
Euler	0.5	1.3243606
Euler	0.25	1.7023376
Euler verbessert	1	1.6487213
Euler verbessert	0.25	2.1699841
Runge-Kutta vereinfacht	1	1.8591409
Runge-Kutta vereinfacht	0.25	2.4250585
Runge-Kutta	1	3.1065161

Die grosse Streueung der Werte ist ein Hinweis darauf, dass die Lösung der Differentialgleichung ein besonderes Problem beinhaltet.

e) Die Differentialgleichung kann separiert werden:

$$\int e^{-y} dy = \int 1 dx \Rightarrow -e^{-y} = x + C$$

Zum Auflösen nach y geht man wie folgt vor

$$\begin{aligned} e^{-y} &= -C - x \\ -y &= \log(-C - x) \\ y &= -\log(-C - x) \end{aligned}$$

Die Konstante C muss etzt so gewählt werden, dass $-\log(-C) = y_0 = 0$ ergibt, oder $-C = e^{y_0} = 1$. Damit ist die Lösung $y(x) = -\log(e^{y_0} - x) = -\log(1 - x)$. Für $x \rightarrow 1$ divergiert $y(x)$ gegen ∞ , so dass die fehlende Konvergenz der numerischen Verfahren nicht mehr überrascht.

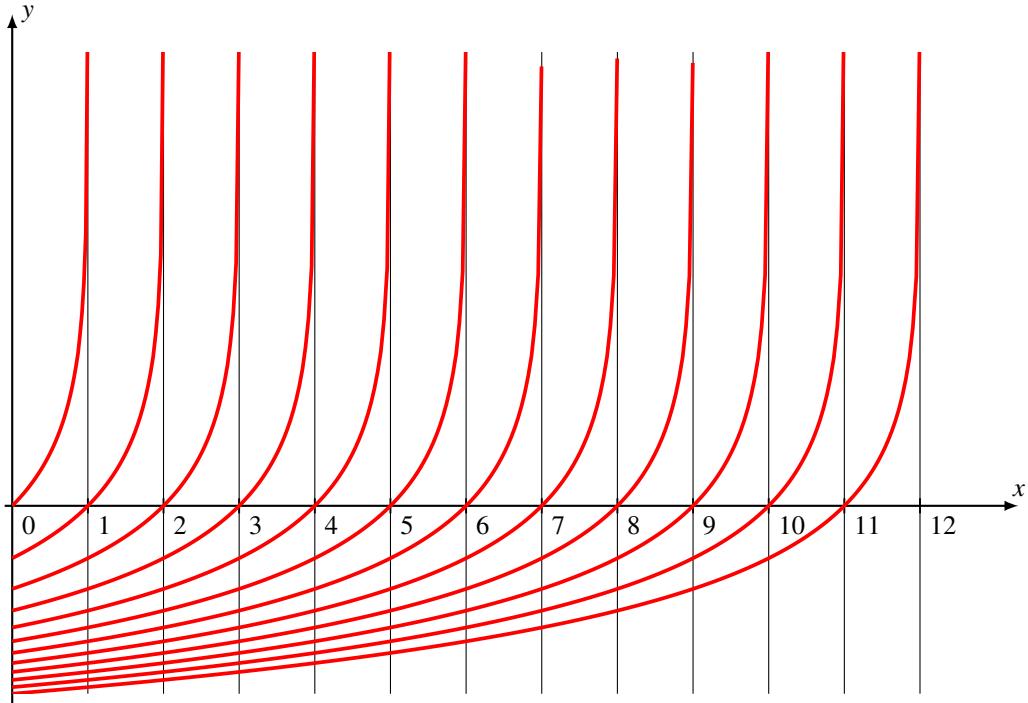


Abbildung 5.10: Die Lösung der Differentialgleichung $y' = e^y$ für verschiedene Anfangswerte $y_0 = -\log(k)$ mit $k = 0, 1, \dots$ haben Singularitäten bei $x = k$.

- f) Tabelle 5.11 zeigt die Resultate der numerischen Rechnung im Vergleich mit der exakten Lösung. Je näher x bei 1 liegt, desto schlechter wird die Genauigkeit der numerischen Lösung.

Die Lösungen der Differentialgleichung dargestellt in Abbildung 5.10 haben Singularitäten, die nichts mit möglichen Singularitäten der Differentialgleichungen zu tun haben, die Funktion e^y ist für alle y definiert. Die Position der Singularität der Lösung hängt von den Anfangswerten ab, sie kann also nur durch exakte Lösung vorhergesagt werden. Die Aufgabe illustriert daher die Schwierigkeiten, die sich stellen, wenn die Lösungen einer Differentialgleichung divergieren. ○

6

Lineare Gleichungssysteme

Die lineare Algebra ist fundamental in vielen Bereichen der angewandten Mathematik. Eine grosse Zahl von Methoden zur Lösung linearer Gleichungssysteme, zur Zerlegung von Matrizen und zur Lösung des Eigenwertproblems sind über die Jahrhunderte entwickelt worden und werden zum Teil bereits in Anfängervorlesungen unterrichtet. Insbesondere der Gaußsche Eliminationsalgorithmus gehört zu den grundlegenden Techniken der numerischen linearen Algebra, er wird hier als bekannt vorausgesetzt.

Die meisten Techniken gehen von relativ kleinen Gleichungssystemen aus. Sie sind aber schlicht nicht leistungsfähig genug oder stabil genug für grosse Systeme, wie sie zum Beispiel bei der Lösung von partiellen Differentialgleichungen oder Simulationen komplexer Systeme auftreten. Auch ist die Laufzeit für die exakte Lösung oft zu lang. Zum Beispiel hat der Gauß-Algorithmus für n Unbekannte Laufzeitkomplexität $O(n^3)$, was für Gleichungssysteme mit einer grossen Zahl von Unbekannten (h. B. $n > 10^5$) zu prohibitiv grossem Rechenaufwand führt. Kompromisse zwischen exakter Lösung und Durchführbarkeit in vernünftiger Zeit sind daher unumgänglich. Bereits Gauß hat zu diesem Zweck iterative numerische Methoden entwickelt.

Dieses Kapitel präsentiert einige wenige Algorithmen des überaus weiten Feldes der numerischen linearen Algebra, welche die Vielfalt dieses Gebietes illustrieren sollen. Eine vertiefte Darstellung kann in [5] gefunden werden. In Abschnitt 6.1 werden einige Spezialfälle diskutiert, für die die Laufzeit des Gauß-Algorithmus weniger schnell als $O(n^3)$ anwächst. In Abschnitt 6.2 wird gezeigt, wie sich Gleichungssysteme unter gewissen Voraussetzungen auch iterativ lösen lassen. Die QR-Zerlegung ist eine andere Formulierung des Problems, eine Basis zu orthonormalisieren, welches schon vom Gram-Schmidt-Algorithmus gelöst wurde, welcher allerdings gewisse Stabilitätsprobleme hat. Der in Abschnitt 6.4 vorgestellte, auf Spiegelungen basierende Algorithmus ist effizienter und stabiler. Eine weitere Möglichkeit, die QR-Zerlegung zu finden, wird in Kapitel 22 ausgeführt. Das Eigenwertproblem für symmetrische Matrizen ist von grundlegender Bedeutung für die Anwendungen, die Lösung über das charakteristische Polynom, welches man oft in den Grundlagenvorlesungen lernt, ist jedoch nur für sehr kleine Matrizen praktikabel. Abschnitt 6.5 stellt das Jacobi-Verfahren zur Diagonalisierung symmetrischer Matrizen vor, welches ebenfalls iterativ arbeitet. Auf das Eigenwertproblem wird in Kapitel 23 vertieft eingegangen. Es erklärt den Francis-Algorithmus, der Basis der meisten Software-Pakete ist, die das Eigenwertproblem lösen können.

6.1 Direkte Verfahren

Das primäre direkte Verfahren zur Lösung linearer Gleichungssysteme ist der Gauss-Algorithmus, der allerdings zur Lösung linearer Gleichungssysteme im Allgemeinen $O(n^3)$ Operationen braucht, was für grosse n zu aufwendig ist.

6.1.1 Thomas-Algorithmus für tridiagonale Matrizen

Es gibt einen Fall, der zum Beispiel bei der Diskretisierung partieller Differentialgleichungen mit nur einer Raumdimension häufig auftritt, nämlich der Fall *tridiagonaler* Koeffizientenmatrix. Dies sind Matrizen der Form

$$A = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 \\ 0 & 0 & a_4 & b_4 & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \dots & a_n & b_n \end{pmatrix}, \quad (6.1)$$

deren Einträge alle 0 sind ausser auf der Diagonalen und den unmittelbar benachbarten Nebendiagonalen. Eine solche Matrix entstand auch im Abschnitt 3.5.3 bei der Bestimmung der Steigungen für die Spline-Interpolationsfunktion.

Da die meisten Einträge der Matrix verschwinden, geben nur die wenigstens Zeilenoperationen des Gauss-Algorithmus überhaupt etwas zu tun. Dadurch sinkt der Aufwand für die Durchführung des Algorithmus.

Vorwärtsreduktion

Die Vorwärtsreduktion entfernt die Elemente unter der Diagonalen und macht die Pivotelemente zu 1. Man muss also nur herausfinden, wie sich die Elemente c_i über der Diagonalen verändern. Bei der Vorwärtsreduktion wird das ursprüngliche Gauss-Tableau daher wie folgt umgewandelt:

$$\left[\begin{array}{cccccc|c} b_1 & c_1 & 0 & 0 & 0 & \dots & d_1 \\ a_2 & b_2 & c_2 & 0 & 0 & \dots & d_2 \\ 0 & a_3 & b_3 & c_3 & 0 & \dots & d_3 \\ 0 & 0 & a_4 & b_4 & c_4 & \dots & d_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{array} \right] \rightarrow \left[\begin{array}{cccccc|c} 1 & c'_1 & 0 & 0 & 0 & \dots & d'_1 \\ 0 & 1 & c'_2 & 0 & 0 & \dots & d'_2 \\ 0 & 0 & 1 & c'_3 & 0 & \dots & d'_3 \\ 0 & 0 & 0 & 1 & c'_4 & \dots & d'_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{array} \right]$$

mit

$$c_i = \begin{cases} \frac{c_1}{b_1} & i = 1 \\ \frac{c_i}{b_i - a_i c'_{i-1}} & i > 1 \end{cases} \quad \text{und} \quad d_i = \begin{cases} \frac{d_1}{b_1} & i = 1 \\ \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & i > 1 \end{cases} \quad (6.2)$$

Die Vorwärtsreduktion ist also mit $O(n)$ Operationen durchführbar.

Beispiel. In der Matrix

$$A = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & \dots \\ 1 & -2 & 1 & 0 & 0 & \dots \\ 0 & 1 & -2 & 1 & 0 & \dots \\ 0 & 0 & 1 & -2 & 1 & \dots \\ 0 & 0 & 0 & 1 & -2 & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{pmatrix} \quad (6.3)$$

gilt $a_i = 1$, $b_i = -2$ und $c_i = 1$ für alle i . Aus den Formeln (6.2) folgt dann

$$\begin{aligned} c'_1 &= -\frac{1}{2} \\ c'_2 &= \frac{1}{-2 - c'_1} = \frac{1}{-2 + \frac{1}{2}} = -\frac{2}{3} \\ c'_3 &= \frac{1}{-2 - c'_2} = \frac{1}{-2 + \frac{2}{3}} = \frac{3}{-2 \cdot 3 + 2} = -\frac{3}{4}. \end{aligned}$$

Daraus lässt sich vermuten, dass

$$c'_k = -\frac{k}{k+1} \quad (6.4)$$

gilt. Tatsächlich lässt sich dies mit vollständiger Induktion beweisen. Zunächst ist $c_1 = -\frac{1}{2} = -\frac{1}{1+1}$ korrekt. Jetzt muss man zeigen, dass aus der Formel (6.4) für k die Formel für $k+1$ folgt. Durch nachrechnen sieht man, dass

$$c'_{k+1} = \frac{1}{-2 - c'_k} = \frac{1}{-2 + \frac{k}{k+1}} = \frac{k+1}{-2(k+1)+k} = -\frac{k+1}{k+2}.$$

Damit ist (6.4) bewiesen.

Wir führen die Vorwärtsreduktion für einen Vektor aus lauter Einsen als rechte Seite durch, also $d_i = 1$. Die Formeln (6.2) gibt dann

$$\begin{aligned} d'_1 &= \frac{d_1}{b_1} = -\frac{1}{2} \\ d'_2 &= \frac{1 - d'_1}{-2 - c'_1} = \frac{\frac{3}{2}}{-2 + \frac{1}{2}} = \frac{\frac{3}{2}}{-\frac{3}{2}} = -1 \\ d'_3 &= \frac{1 - d'_2}{-2 + c'_2} = \frac{1 + 1}{-2 + \frac{2}{3}} = \frac{2 \cdot 3}{-2 \cdot 3 + 2} = -\frac{3}{2}. \end{aligned}$$

Aus diesen Werten lässt sich wieder die Vermutung

$$d'_k = -\frac{k}{2}$$

ableiten. Und tatsächlich beweist die Rechnung

$$d'_{k+1} = \frac{1 - d'_k}{-2 + \frac{k}{k+1}} = \frac{1 + \frac{k}{2}}{-2 + \frac{k}{k+1}} = \frac{1 + \frac{k}{2}}{-2(k+1)+k}(k+1) = \frac{k+2}{-2k-2+k} \cdot \frac{k+1}{2} = \frac{k+2}{-k-2} \cdot \frac{k+1}{2} = -\frac{k+1}{2}$$

diese Vermutung mit vollständiger Induktion. \circ

Rückwärtseinsetzen

Das Rückwärtseinsetzen ist jetzt ebenfalls sehr einfach:

$$x_k = \begin{cases} d'_n & k = n \\ d'_k - c'_k x_{k+1} & k < n \end{cases} \quad (6.5)$$

Auch das Rückwärtseinsetzen ist daher mit $O(n)$ Operationen machbar. Der Thomas-Algorithmus löst also ein tridiagonales Gleichungssystem in $O(n)$ Operationen oder berechnet die inverse Matrix einer tridiagonalen Matrix in $O(n^2)$ Operationen.

Beispiel. Für die Matrix A von (6.3) und die rechte Seite von Einsen aus dem entsprechenden Beispiel können wir jetzt auch das Rückwärtseinsetzen durchführen. Dazu wenden wir die Formeln (6.5) auf die oben gefundenen Werte $c'_k = -k/(k+1)$ und $d'_k = -k/2$ an und erhalten

$$\begin{aligned} x_n &= -\frac{n}{2} &= \frac{1 \cdot (n-0)}{2} \\ x_{n-1} &= -\frac{n-1}{2} - \left(-\frac{n-1}{n}\right) \cdot \left(-\frac{n}{2}\right) = -\frac{n-1}{2} - \frac{n-1}{2} = -(n-1) &= -\frac{2(n-1)}{2} \\ x_{n-2} &= -\frac{n-2}{2} - \left(-\frac{n-2}{n-1}\right) \cdot (-n+1) = -\frac{n-2}{2} - (n-2) = -\frac{3(n-2)}{2} &= -\frac{3(n-2)}{2} \\ x_{n-3} &= -\frac{n-3}{2} - \frac{n-3}{n-2} \frac{3(n-2)}{2} = -\frac{n-3}{2} - 3 \frac{n-3}{2} = \frac{4(n-3)}{2} &= -\frac{4(n-3)}{2} \end{aligned}$$

woraus man die Vermutung

$$x_{n-k} = -\frac{(k+1)(n-k)}{2} \quad (6.6)$$

ableiten. Tatsächlich rechnet man nach

$$\begin{aligned} x_{n-(k+1)} &= d'_{n-(k+1)} - c'_{n-(k+1)} x_{n-k} = -\frac{n-(k+1)}{2} - \frac{n-k-1}{n-k} \cdot \frac{(k+1)(n-k)}{2} \\ &= -\frac{n-(k+1) + (n-k-1)(k+1)}{2} \\ &= -\frac{(k+2)(n-(k+1))}{2}. \end{aligned}$$

Damit ist die Formel für x_n bewiesen.

Man kann natürlich auch direkt verifizieren, dass die x_k das tridiagonale Gleichungssystem lösen:

$$\begin{aligned} x_{n-(k-1)} - 2x_{n-k} + x_{n-(k+1)} &= -\frac{k(n-(k-1)) - 2(k+1)(n-k) + (k+2)(n-(k+1))}{2} \\ &= -\frac{kn - k^2 + k - 2kn - 2n + 2k^2 + 2k + kn + 2n - k^2 - 3k - 2}{2} = 1. \end{aligned}$$

Dies gilt auch für $k = 0$ und $k = n - 1$, weil die Formel (6.6)

$$x_{n+1} = x_{n-(-1)} = -\frac{(-1+1)(n-(-1))}{2} = 0 \quad \text{und} \quad x_0 = x_{n-n} = -\frac{(n+1)(n-n)}{2}$$

zur Folge hat. ○

6.1.2 Zyklisch tridiagonale Matrix

Die Diskretisation des Poisson-Problems oder der Wärmeleitungsgleichung auf einem Ring liefert eine tridiagonale Matrix, in der zusätzlich die Einträge in den Ecken links unten und rechts oben von 0 verschieden sind. Bei der Durchführung des Gauss-Algorithmus werden damit zusätzliche Einträge im Tableau von 0 verschieden sein, so dass zusätzliche Operationen notwendig werden:

- Zusätzlich zu der Zeilenoperation, die die Element a_i zu 0 macht braucht es jeweils noch eine Operation, welche das Element in der letzten Zeile annihielt. Dadurch wird aber ein neuer nicht verschwindender Eintrag erzeugt, so dass eine Operation für jedes Element der letzten Zeile nötig wird, also $O(n)$ zusätzliche Operationen.
- Die Zeilenoperationen erzeugen aus dem Element in der rechten oberen Ecke nicht verschwindende Elemente in der Spalte ganz rechts. Beim Rückwärtseinsetzen sind daher im ersten Schritt $O(n)$ zusätzliche Operationen notwendig.

Auch ein Gleichungssystem mit zyklisch tridiagonaler Matrix lässt sich also mit $O(n)$ Operationen lösen und die Matrix lässt sich mit $O(n^2)$ Operationen invertieren.

Tatsächlich erfordert das Invertieren einer Matrix nur $O(n^2)$ zusätzliche Operationen, wenn die Inverse einer Matrix bereits bekannt ist, die sich in nur einer Stelle unterscheidet. Dies ist der Inhalt der Sherman-Morrison-Woodbury-Formel, die im nächsten Abschnitt untersucht wird.

6.1.3 Sherman-Morrison-Woodbury-Formel

Eine zyklisch tridiagonale Matrix unterscheidet sich nur in zwei Positionen von einer tridiagonalen Matrix. Diese kleine Änderung verdoppelt den Rechenaufwand für die Lösung des Gleichungssystems. Es stellt sich die Frage, ob sich eine Matrix auf billige Art ändern lässt derart, dass die Invertierung mit dem weniger aufwendigen Thomas-Algorithmus möglich wird. Die *Sherman-Morrison-Woodbury-Formel*

Satz 6.1 (Shermann-Morrison-Woodbury). *Ist A eine Matrix mit der Inversen A^{-1} und sind die Vektoren u und v gegeben derart, dass $v^t A^{-1} u \neq 1$ ist, dann gilt*

$$(A - uv^t)^{-1} = A^{-1} + \frac{1}{1 - v^t A^{-1} u} A^{-1} u v^t A^{-1}. \quad (6.7)$$

Beweis. Wir schreiben

$$B = A^{-1} + \frac{1}{1 - v^t A^{-1} u} A^{-1} u v^t A^{-1}$$

für die rechte Seite der Formel (6.7). Wir beweisen die Behauptung durch Nachrechnen. Den Ausdruck $v^t A^{-1} u$ kürzen wir im Folgenden c ab. Es gilt

$$\begin{aligned} (A - uv^t)B &= (A - uv^t) \left(A^{-1} + \frac{1}{1 - v^t A^{-1} u} A^{-1} u v^t A^{-1} \right) \\ &= AA^{-1} - uv^t A^{-1} + \frac{1}{1 - c} \left(AA^{-1} u v^t A^{-1} - u \underbrace{v^t A^{-1} u}_{=c} v^t A^{-1} \right) \\ &= E - uv^t A^{-1} + \frac{1}{1 - c} \left(uv^t - cuv^t \right) A^{-1} \\ &= E - uv^t A^{-1} + uv^t A^{-1} = E. \end{aligned}$$

□

Beispiel. Die Matrix

$$A = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{pmatrix} \quad \text{und} \quad B = \begin{pmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ \color{red}{1} & 0 & 0 & 1 & -2 \end{pmatrix}$$

unterscheiden sich nur durch ein einziges Element in der linken unteren Ecke. Dieses Element kann betrachtet werden als ein Produkt von Standardbasisvektoren

$$E_{51} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} = e_5 e_1^t.$$

Ausserdem ist A^{-1} mit Hilfe des Thomas-Algorithmus leicht berechenbar, sie ist

$$A^{-1} = \begin{pmatrix} -\frac{5}{6} & -\frac{2}{3} & -\frac{1}{2} & -\frac{1}{3} & -\frac{1}{6} \\ -\frac{2}{3} & -\frac{4}{3} & -1 & -\frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{2} & -1 & -\frac{3}{2} & -1 & -\frac{1}{2} \\ -\frac{1}{3} & -\frac{2}{3} & -1 & -\frac{4}{3} & -\frac{2}{3} \\ -\frac{1}{6} & -\frac{1}{3} & -\frac{1}{2} & -\frac{2}{3} & -\frac{5}{6} \end{pmatrix}.$$

Um Satz 6.1 anzuwenden, müssen wir $B = A - uv^t$ schreiben. Dies gelingt mit $u = -e_5$ und $v = e_1$. Die Formel (6.7) ist nur dann anwendbar, wenn $v^t A^{-1} u \neq 1$ ist, wir rechnen daher nach und finden $c = v^t A^{-1} u = \frac{1}{6}$. Damit kann jetzt die Inverse von B bestimmt werden:

$$B^{-1} = (A - uv^t)^{-1} = A^{-1} + \frac{6}{5} A^{-1} u v^t A^{-1} = \begin{pmatrix} -1 & -\frac{4}{5} & -\frac{3}{5} & -\frac{2}{5} & -\frac{1}{5} \\ -1 & -\frac{8}{5} & -\frac{6}{5} & -\frac{4}{5} & -\frac{2}{5} \\ -1 & -\frac{7}{5} & -\frac{9}{5} & -\frac{6}{5} & -\frac{3}{5} \\ -1 & -\frac{6}{5} & -\frac{7}{5} & -\frac{8}{5} & -\frac{4}{5} \\ -1 & -1 & -1 & -1 & -1 \end{pmatrix}. \quad \circ$$

Man beachte, dass sich jede Matrix mit Rang 1 in der Form uv^t schreiben lässt. Da Rang $C = 1$ ist, hat der Nullraum $N = \ker C$ von C die Dimension $n - 1$ und es gibt einen Einheitsvektor v , der orthogonal auf allen Vektoren des Nullraumes ist. Jeder Vektor $x \in \mathbb{R}^n$ kann zerlegt werden in $x = x_{\parallel} + x_{\perp}$ mit $x_{\perp} \in \ker C$ und $x_{\parallel} = v(v \cdot x) = vv^t x$, es gilt dann $Cx = Cx_{\parallel} + Cx_{\perp} = Cx_{\parallel}$. Setzen wir $u = Cv$, dann folgt wegen $x_{\parallel} = v(v \cdot x)$

$$Cx = Cx_{\parallel} = Cv v^t x = uv^t x \quad \Rightarrow \quad C = uv^t.$$

Satz 6.1 besagt daher, dass die Inverse einer Matrix B , die sich nur um eine Matrix vom Rang 1 von A unterscheidet, mit Rechenaufwand $O(n^2)$ aus der Inversen A^{-1} ermittelt werden kann. Dies bringt allerdings nur einen Nutzen, wenn der Aufwand zur Berechnung von (6.7) geringer ist als eine volle Durchführung des Gauss-Algorithmus. Dazu beachtet man, dass ein Produkt Matrix \times Vektor mit $O(n^2)$ Operationen möglich ist. Sowohl $A^{-1}u$ und $v^t A^{-1}$ lassen sich daher mit $O(n^2)$ Operationen bestimmen. Das Produkt des Spaltenvektors $A^{-1}u$ mit dem Zeilenvektor $v^t A^{-1}$ ist sogar in $O(n)$ Operationen möglich. Auf diese Weise lässt sich die Formel 6.7 mit $O(n^2)$ Operationen durchführen und ist daher weniger aufwändig als der Gauss-Algorithmus.

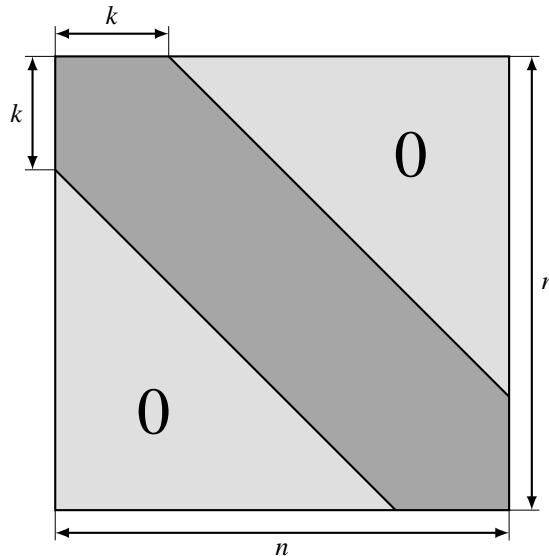


Abbildung 6.1: Eine Bandmatrix ist eine $n \times n$ -Matrix, deren Einträge ausserhalb eines Bandes um die Diagonale verschwinden.

6.1.4 Bandmatrizen

Tridiagonale Matrizen waren einfach zu invertieren, weil es nur wenige von 0 verschiedene Einträge nahe der Diagonalen gab. Diese Bedingung erfüllen auch sogenannte *Bandmatrizen*. Eine $n \times n$ -Matrix A mit Einträgen a_{ij} ist eine Bandmatrix der Breite k , wenn $a_{ij} = 0$ für $|i - j| > k$. Schematisch kann man eine solche Matrix wie in Abbildung 6.1 darstellen.

Der Gauss-Algorithmus braucht auch in einer Bandmatrix nicht die volle Zahl von Operationen. Bei der Pivotdivision müssen höchstens $2k + 1$ Divisionen ausgeführt werden. Danach sind genau k Zeilenoperationen notwendig, um die Spalte unter dem Pivotelement zu 0 zu machen. Dabei werden nur die Elemente in $2k + 1$ Spalten modifiziert, der Aufwand für jeden Schritt der Vorwärtsreduktion ist daher (k^2) . Nach n Schritten wurde daher Rechenzeit $O(nk^2)$ aufgewendet. Beim Rückwärts-einsetzen werden in n Iteration jeweils k Zeilenoperationen in genau zwei Spalten durchgeführt, es fallen also nur $O(nk)$ Operationen an. Insgesamt sind also nur $O(nk^2)$ Operationen nötig, um ein Gleichungssystem zu lösen, oder $O(n^2k^2)$ für eine Matrixinvertierung.

Die Diskretisation einer partiellen Differentialgleichung in der Ebene mit finiten Differenzen (siehe auch Abschnitt 7.2) führt auf natürliche Weise zu einem linearen Gleichungssystem mit einer Bandmatrix mit $k \approx \sqrt{n}$. Ein solches Gleichungssystem lässt sich mit Aufwand $O(n^2)$ statt $O(n^3)$ lösen.

6.2 Iterative Gleichungslösung

Gegeben ist eine lineares Gleichungssystem von n Gleichungen mit n Unbekannten, welches wir als

$$\begin{array}{llllll} a_{11}x_1 + & a_{12}x_2 + & \dots + & a_{1n}x_n = & b_1 \\ a_{21}x_1 + & a_{22}x_2 + & \dots + & a_{2n}x_n = & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}x_1 + & a_{n2}x_2 + & \dots + & a_{nn}x_n = & b_n \end{array} \quad (6.8)$$

schreiben. Abgekürzt wird das Gleichungssystem auch $Ax = b$ notiert, wobei $A = (a_{ij})$ die Koeffizientenmatrix ist, $x = (x_k)$ der Vektor der Unbekannten und $b = (b_k)$ der Vektor der rechten Seiten.

6.2.1 Iterative Lösung nach Gauss-Seidel

Jede der Gleichungen (6.8) kann nach jeder Variablen aufgelöst werden, deren zugehöriger Koeffizient von 0 verschieden ist. Die Gleichung k in (6.8) ist

$$a_{k1}x_1 + a_{k2}x_2 + \dots + a_{kk}x_k + \dots + a_{kn}x_n = b_k$$

Aufgelöst nach x_k ist dies

$$\textcolor{red}{x}_k = \frac{1}{a_{kk}}(b_k - a_{k1}x_1 - a_{k2}x_2 - \dots - a_{kn}x_n),$$

sofern $a_{kk} \neq 0$. Diese Gleichung kann dazu verwendet werden, die Werte für die Unbekannten zu verbessern. Wir verwenden daher die Notation $x^{(m)}$ für die m -te Approximation der Lösung.

Satz 6.2 (Gauss-Seidel-Iteration). *Unter geeigneten Voraussetzungen konvergiert die Folge $x^{(m)}$ definiert durch*

$$\textcolor{red}{x}_k^{(m)} = \frac{1}{a_{kk}}(b_k - a_{k1}x_1^{(m)} - \dots - a_{k,k-1}x_{k-1}^{(m)} - a_{k,k+1}x_{k+1}^{(m-1)} - \dots - a_{kn}x_n^{(m-1)}) \quad (6.9)$$

mit Startwert $x^{(0)} = 0$ gegen die Lösung x des Gleichungssystems $Ax = b$.

In Abschnitt 6.2.3 werden die Bedingungen genauer untersucht, die Konvergenz des Verfahrens gegen die Lösung garantieren können.

Beispiel. Sei das Gleichungssystem gegeben durch

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 7 \\ 6 \\ 5 \end{pmatrix}. \quad (6.10)$$

Die Berechnung der Folge $x^{(m)}$ nach (6.9) liefert die Werte in Tabelle 6.1. Die Konvergenz scheint linear zu sein. ○

m	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$
0	0.0000000	0.0000000	0.0000000
1	3.5000000	0.8333333	0.1666667
2	3.0000000	0.9444444	0.2638889
3	2.8958333	0.9467593	0.2893519
4	2.8819444	0.9429012	0.2937886
5	2.8816552	0.9415187	0.2942065
6	2.8821373	0.9412187	0.2941610
7	2.8823102	0.9411762	0.2941284
8	2.8823476	0.9411747	0.2941194
9	2.8823531	0.9411759	0.2941177
10	2.8823531	0.9411764	0.2941176
∞	2.8823529	0.9411764	0.2941176

Tabelle 6.1: Lösung des Gleichungssystems mit Koeffizientenmatrix A und rechter Seite b aus (6.10) mit Hilfe des Gauss-Seidel-Algorithmus. In der letzten Zeile die exakten Resultate, erhalten mit dem Gauss-Algorithmus.

6.2.2 Matrixformulierung

Die Iterationsformel (6.9) verknüpft bei der Berechnung von $\textcolor{red}{x}^{(m)}$ Komponenten von $x^{(m-1)}$ und $\textcolor{red}{x}^{(m)}$, was es etwas schwieriger macht, die Iteration als Fixpunktiteration der Form $\textcolor{red}{x}^{(m)} = Fx^{(m-1)}$ mit einer $n \times n$ -Matrix F zu schreiben. Um dies zu erreichen, zerlegen wir die Matrix A in drei Summanden $A = L + D + U$, wobei L eine untere Dreiecksmatrix mit Nullen auf der Diagonalen sein soll, D eine Diagonalmatrix und U eine obere Dreiecksmatrix mit Nullen auf der Diagonalen, also

$$L = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ a_{21} & 0 & 0 & \dots & 0 & 0 \\ a_{31} & a_{3,2} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \dots & 0 & 0 \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{n,n-1} & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & \dots & a_{1,n-2} & a_{1,n-1} & a_{1n} \\ 0 & 0 & \dots & a_{1,n-2} & a_{2,n-1} & a_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-2,n-1} & a_{n-2,n} \\ 0 & 0 & \dots & 0 & 0 & a_{n-1,n} \\ 0 & 0 & \dots & 0 & 0 & 0 \end{pmatrix}$$

und

$$D = \text{diag}(a_{11}, a_{22}, \dots, a_{n-1,n-1}, a_{nn}).$$

Die Iterationsformel (6.9) lässt sich mit diesen Matrizen schreiben als

$$D\textcolor{red}{x}^{(m)} = b - L\textcolor{red}{x}^{(m)} - UX^{(m-1)}.$$

Auflösen nach $x^{(m)}$ führt auf

$$\textcolor{red}{x}^{(m)} = (D + L)^{-1}(b - UX^{(m-1)}). \quad (6.11)$$

Die Form (6.11) für das Gauss-Seidel-Iterationsverfahren ist jetzt die einer Fixpunkt-Iteration.

6.2.3 Konvergenzbedingung

In Kapitel 1 haben wir gelernt, dass eine Fixpunktiteration konvergiert, wenn der Betrag der Ableitung < 1 ist. Hier liegt jedoch eine Matrix-Iteration mit der Abbildung

$$F(x) = \underbrace{(D + L)^{-1}b - (D + L)^{-1}Ux}_{= c} = c - (D + L)^{-1}Ux$$

vor. Die Ableitung ist daher ebenfalls eine Matrix, nämlich

$$D_x F = (D + L)^{-1}U,$$

und der Fehler der Iteration m ist

$$\delta_m = (D + L)^{-1}U\delta_{m-1}. \quad (6.12)$$

Konvergenz kann also nur vorliegen, wenn dieser Vektor im Laufe der Iteration immer kleiner wird. Dies ist zum Beispiel dann der Fall, wenn die *Norm* der Matrix kleiner als 1 ist:

Definition 6.3. *Die Norm einer Matrix M ist*

$$\|M\| = \max\{|Mx| \mid x \in \mathbb{R}^n \wedge |x| = 1\}.$$

Für einen Vektor $x \in \mathbb{R}^n$ gilt $|Mx| \leq \|M\| \cdot |x|$.

Die Bedingung (6.12) bedeutet jedoch nicht, dass die Norm der Ableitung < 1 sein muss, es genügt, wenn genügend hohe Potenzen der Ableitung eine Norm < 1 haben.

Beispiel. Die Matrix

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

hat Norm

$$\|M\| = \max_{|x|=1} |Mx| = \max_{t \in \mathbb{R}} \sqrt{2^2 \cos^2 t + \frac{1}{3^2} \sin^2 t} \geq 2.$$

Da aber

$$M^2 = \begin{pmatrix} \frac{2}{3} & 0 \\ 0 & \frac{2}{3} \end{pmatrix} \quad \Rightarrow \quad \|M^2\| = \frac{2}{3}$$

ist, wird eine Iteration mit Ableitungsmatrix M trotzdem konvergieren, weil der Fehler nach jedem zweiten Schritt um den Faktor $\frac{2}{3}$ kleiner geworden ist. \circlearrowright

Dies führt uns auf die Grösse

$$\pi(M) = \limsup_{n \rightarrow \infty} \|M^n\|^{\frac{1}{n}}. \quad (6.13)$$

Ist $\pi(M) > 1$, dann gibt es Anfangsvektoren v für die Iteration, für die $M^k v$ über alle Grenzen wächst. Ist $\pi(M) < 1$, dann wird jeder Anfangsvektor v zu einer Iterationsfolge $M^k v$ führen, die gegen 0 konvergiert. Die Kennzahl $\pi(M)$ erlaubt also zu entscheiden, ob ein Iterationsverfahren konvergent ist.

Die Berechnung von $\pi(M)$ als Grenzwert ist sehr unhandlich. Viel einfacher ist der Begriff des Spektralradius.

Definition 6.4. *Der Spektralradius der Matrix M ist der Betrag des betragsgrössten Eigenwertes.*

Wir werden später in Satz 6.7 zeigen, dass $\pi(M) = \varrho(M)$, dies ist bekannt als der Satz von Gelfand. Um die beiden Begriffe bis dann auseinander halten zu können, nennen wir den Grenzwert 6.13 den *Gelfand-Radius* $\pi(M)$ der Matrix M . Wir erlauben uns aber, die Überlegungen zu den Iterationsverfahren mit dem Spektralradius zu formulieren.

Das Gauss-Seidel-Iterationsverfahren ist also genau dann für alle Startwerte x_0 linear konvergent, wenn der Spektralradius

$$\varrho((L + D)^{-1} U) < 1$$

ist.

6.2.4 Zerlegungsverfahren

Das Gauss-Seidel-Verfahren ist nur ein Beispiel einer ganzen Familie von iterativen Lösungsverfahren für lineare Gleichungssysteme. Sie basieren alle auf einer Zerlegung $A = B + C$ der Matrix A . Das Gauss-Seidel-Verfahren ist der Fall

$$A = \underbrace{D + L}_{= B} + \underbrace{R}_{= C}.$$

Das ursprüngliche Gleichungssystem $Ax = b$ wird jetzt ebenfalls aufgeteilt in $Bx + Cx = b$. Ein iteratives Verfahren ergibt sich jetzt dadurch, dass für die beiden x in der Aufteilung verschiedene Iterationen der Lösung verwendet werden, also $Bx^{(m+1)} + Cx^{(m)} = b$. Wir erhalten somit das Iterationsverfahren

$$\textcolor{red}{x}^{(m+1)} = B^{-1}(b - Cx^{(m)}) = b_0 - B^{-1}Cx^{(m)} \quad (6.14)$$

Dies ist natürlich nur sinnvoll, wenn sich die Matrix B wesentlich leichter invertieren lässt als A , da andernfalls die Bestimmung von $x^{(m+1)}$ nicht einfacher ist als die Lösung des ursprünglichen Gleichungssystems.

Iteriert wird also die Anwendung der Matrix $B^{-1}C$. Aus der oben entwickelten Theorie lesen wir ab, dass das Zerlegungsverfahren genau dann konvergiert, wenn der Spektralradius $\varrho(B^{-1}C)$ von $B^{-1}C$ kleiner ist als 1.

Das Verfahren von Jacobi

Beim Gauss-Seidel-Verfahren wurde jede einzelne Gleichung des Gleichungssystems nach einer der Variablen aufgelöst und der neue Wert bei der nächsten Iteration gleich wieder verwendet. Man hätte natürlich auch erst aus jeder Gleichung einen neuen Wert für alle Variablen bestimmen können, bevor man diese neuen Werte in der nächsten Iteration verwendet. Schreibt man das Gleichungssystem wieder als

$$(L + D + U)x = Lx + D\textcolor{red}{x} + Ux = b$$

dann bedeutet dies, dass man nach der roten Variablen $\textcolor{red}{x}$ auflöst, also

$$D\textcolor{red}{x} = Lx + Ux + b \quad \Rightarrow \quad \textcolor{red}{x} = D^{-1}(L + U)x + D^{-1}b.$$

Auch dieses nach *Jacobi* benannte Verfahren ist also ein Zerlegungsverfahren, diesmal mit $B = D$ und $C = L + U$. Es konvergiert genau dann, wenn $\varrho(D^{-1}(L + U)) < 1$. Dieser Fall tritt dann ein, wenn die Diagonalelemente sehr viel grösser sind als der Rest der Matrix.

Richardson-Verfahren

Das *Richardson-Verfahren* ist besonders gut geeignet für den Fall, dass die Matrix A nahe an einem Vielfachen der Einheitsmatrix ist. Man verwendet dazu die Aufspaltung

$$A = B + C = \tau E + (A - \tau E).$$

Die Matrix $B = \tau E$ ist sehr einfach zu invertieren: $B^{-1} = \frac{1}{\tau}E$. Das Richardson-Verfahren zeichnet sich also durch sehr geringen Aufwand bei der Invertierung aus. Die zu iterierende Matrix ist

$$B^{-1}C = \frac{1}{\tau}E(A - \tau E) = \frac{1}{\tau}A - E.$$

Je näher die Matrix A bei τE liegt, desto näher werden die Eigenwerte von A bei denen von τE also bei τ liegen, und desto näher werden die Eigenwerte von $B^{-1}C$ bei 0 liegen, was zu Konvergenz des Verfahrens führt.

Die Iterationsformel für die Lösung ist

$$\textcolor{red}{x} = B^{-1}b - B^{-1}Cx = \frac{1}{\tau}(b - (A - \tau E)x) = \frac{1}{\tau}(b - Ax) + x. \quad (6.15)$$

In jedem Schritt wird x also ein Vielfaches von $b - Ax$ korrigiert. Die Differenz $b - Ax$ misst, um wieviel die Gleichungen nicht erfüllt ist, sie heißt auch *Residuum*.

Man kann die Iterationsformel (6.15) auch als eine Variante des Newton-Verfahrens verstehen. Gesucht wird eine Nullstelle der Funktion $f(x) = b - Ax$. Das Newton-Verfahren besagt, dass man x um $-Df(x)^{-1} \cdot f(x)$ korrigieren muss. Allerdings ist $Df(x) = -A$, so dass die Newton-Iterationsformel zu

$$\textcolor{red}{x} = x - Df(x)^{-1} \cdot f(x) = x + A^{-1} \cdot (b - Ax)$$

wird. Genau die Berechnung von A^{-1} soll aber vermieden werden, daher approximiert man $A \approx \tau E$, also

$$\textcolor{red}{x} = x + \frac{1}{\tau}(b - Ax).$$

Dies ist wieder die Iterationsformel (6.15) für das Richardson-Verfahren.

Im Kapitel 24 wird die Methode der konjugierten Gradienten vorgeführt. Mit ihr lässt sich für symmetrische, positiv definite Matrizen A ein auf einem Minimalproblem basierendes iteratives Verfahren konstruieren lässt, welches ebenfalls genau die Residuen als Verbesserungsrichtungen für die Lösung verwendet.

Successive Overrelaxation (SOR)

Das Gauss-Seidel-Verfahren versucht jede einzelne Variable ohne Rücksicht auf alle anderen zu korrigieren. Die einzelne Korrektur soll also den ganzen verbleibenden Fehler ausbügeln. Es ist klar, dass diese Korrektur zwar ungefähr in die richtige Richtung erfolgt, aber nicht vom richtigen Betrag sein wird. Das Richardson-Verfahren enthält einen Parameter, mit dem man die Konvergenz beeinflussen kann. Man muss also versuchen, das Ausmass der Korrektur im Gauss-Seidel-Verfahren zu parametrisieren und den Parameter so zu wählen, dass die Konvergenzgeschwindigkeit optimiert werden kann.

In einem Iterationsschritt des Gauss-Seidel-Verfahrens wird die Variable x_k durch

$$\textcolor{red}{x}_k = \frac{1}{a_{kk}} \left(b_k - \sum_{i \neq k} a_{ki} x_i \right)$$

ersetzt, also um den Betrag

$$\delta_k = \textcolor{red}{x}_k - x_k = \frac{1}{a_{kk}} \left(b_k - \sum_{i \neq k} a_{ki} x_i \right) - x_k$$

korrigiert. Statt um δ_k korrigieren wir jetzt um $\omega \delta_k$, also auf

$$\textcolor{red}{x}_k = x_k + \omega \left(\frac{1}{a_{kk}} \left(b_k - \sum_{i \neq k} a_{ki} x_i \right) - x_k \right).$$

Wie im Gauss-Seidel-Verfahren werden die korrigierten Variablen bereits bei der nächsten Gleichung wieder verwendet, es ist also

$$\textcolor{red}{x}_k = x_k + \omega \left(\frac{1}{a_{kk}} \left(b_k - \sum_{i=1}^{k-1} a_{ki} \textcolor{red}{x}_i - \sum_{i=k+1}^n a_{ki} x_i \right) - x_k \right).$$

In Matrixform geschrieben wird dies zu

$$\textcolor{red}{x} = x + \omega(D^{-1}(b - L\textcolor{red}{x} - Ux) - x).$$

Nach Multiplikation von links mit D erhalten wir

$$D\textcolor{red}{x} = Dx + \omega b - \omega L\textcolor{red}{x} - \omega Ux - \omega Dx.$$

Um den Zusammenhang mit den Zerlegungsverfahren herzustellen, dividieren wir durch ω und bringen alle Terme mit x und $\textcolor{red}{x}$ auf die linke Seite, was

$$\frac{1}{\omega} D\textcolor{red}{x} - \frac{1}{\omega} Dx + L\textcolor{red}{x} + Ux + Dx = b$$

ergibt. Dies können wir etwas klarer als

$$\underbrace{\left(L + \frac{1}{\omega} D \right)}_{= B_\omega} \textcolor{red}{x} + \underbrace{\left(U + \left(1 - \frac{1}{\omega} \right) D \right)}_{= C_\omega} x = b$$

umordnen. Wir haben damit ein Zerlegungsverfahren basierend auf der Zerlegung $A = B_\omega + C_\omega$ gefunden.

Die Korrektur der Variablen x_k nach Gauss-Seidel wird oft Relaxation genannt. Ein Parameter-Wert $\omega > 1$ führt eine grössere Korrektur aus, man bezeichnet dies als Überrelaxation. Die Korrektur wird wie beim Gauss-Seidel-Verfahren nacheinander auf alle Variablen angewendet wird, daher heisst dieses Verfahren *Successive Overrelaxation* oder kurz SOR. Geignete Werte von ω liegen zwischen 0 und 2, das Gauss-Seidel-Verfahren ist der Fall $\omega = 1$. Man kann zeigen, dass dieses Verfahren für symmetrische Matrizen immer konvergiert.

Tabelle 6.2 stellt die in diesem Abschnitt beschriebenen, auf dem Zerlegungsprinzip basierenden Verfahren zusammen.

Beispiel. Zur Illustration berechnen wir die verschiedenen Spektralradien für eine symmetrische 30×30 -Matrix der Form $E + R$, wobei R aus zufälligen Werten im Intervall $[0, 0.1]$ besteht. In einer typischen Rechnung ergaben sich die Werte

Gauss-Seidel: $\varrho((D + L)^{-1}U) = 0.29160$ konvergiert

Jacobi: $\varrho(D^{-1}(L + U)) = 1.3724$ konvergiert nicht.

Methode	B	C	Iterationsformel
Jacobi	D	$L + U$	$x = D^{-1}b - D^{-1}(L + U)x$
Gauss-Seidel	$D + L$	U	$x = (D + L)^{-1}b - (D + L)^{-1}Ux$
Richardson	τE	$A - \tau E$	$x = \frac{1}{\tau}(b - Ax) + x$
SOR	$B_\omega = \frac{1}{\omega}D + L$	$C_\omega = \left(1 - \frac{1}{\omega}\right)D + U$	$x = B_\omega^{-1}b - B_\omega^{-1}C_\omega x$

Tabelle 6.2: Zusammenstellung iterativer Verfahren für die Lösung eines Gleichungssystems $Ax = b$, welche auf einer Zerlegung der Matrix A in $A = B + C$ basieren.

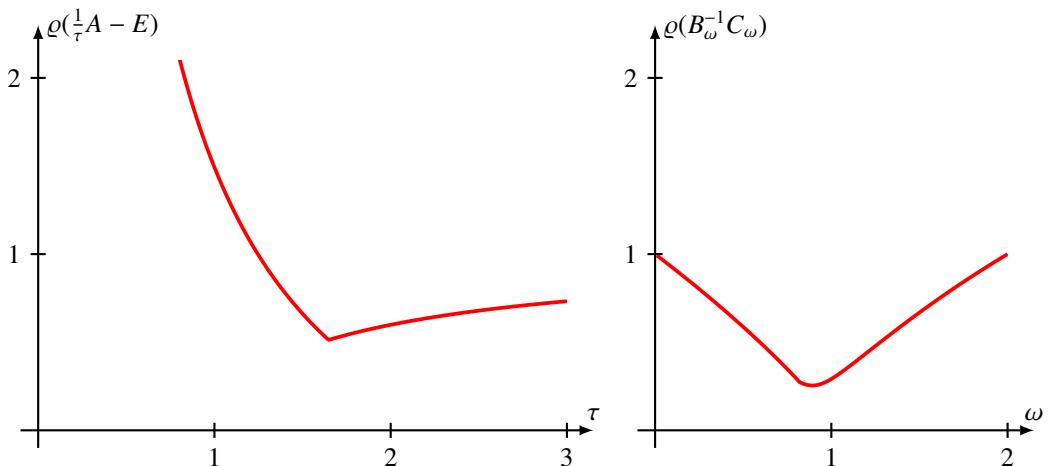


Abbildung 6.2: Spektralradius für das Richardson-Verfahren in Abhängigkeit von τ und für SOR in Abhängigkeit von ω . In beiden Fällen gibt es einen Parameterwert, für den die Konvergenzgeschwindigkeit maximal ist.

In Abbildung 6.2 ist der Spektralradius der Iterationsmatrix für das Richardson-Verfahren und für SOR dargestellt. In beiden Fällen gibt es einen Wert für den Parameter, für den der Spektralradius minimal und damit die Konvergenzgeschwindigkeit am grössten wird. Es stellt sich heraus, dass das SOR-Verfahren in diesem speziellen Fall für ein $\omega < 1$ am schnellsten konvergiert, also für Unterrelaxation. \circlearrowright

6.3 Kondition

Die Diskussion der iterativen Verfahren in Abschnitt 6.2 hat gezeigt, dass der Spektralradius aus Definition 6.4 Auskunft gibt darüber, ob ein iteratives Verfahren konvergiert. Insbesondere wurde behauptet, dass Spektralradius $\varrho(M)$ und Gelfand-Radius $\pi(M)$ übereinstimmen. In diesem Abschnitt sollen diese Kennzahl noch etwas vertieft untersucht werden.

6.3.1 Gelfand-Radius und Eigenwerte

In Abschnitt 6.2.3 ist der Gelfand-Radius mit Hilfe eines Grenzwertes definiert worden. Nur dieser Grenzwert ist in der Lage, über die Konvergenz eines Iterationsverfahrens Auskunft zu geben. Der Grenzwert ist aber sehr mühsam zu berechnen. Es wurde angedeutet, dass der Gelfand-Radius mit dem Spektralradius übereinstimmt, dem Betrag des des betragsgrößten Eigenwertes. Dies hat uns ein vergleichsweise einfach auszuwertendes Konvergenzkriterium geliefert. In diesem Abschnitt soll diese Identität zunächst an Spezialfällen und später ganz allgemein gezeigt werden.

Spezialfall: Diagonalisierbare Matrizen

Ist eine Matrix A diagonalisierbar, dann kann Sie durch eine Wahl einer geeigneten Basis in Diagonalfom

$$A' = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

gebracht werden, wobei die Eigenwerte λ_i möglicherweise auch komplex sein können. Die Bezeichnungen sollen so gewählt sein, dass λ_1 der betragsgrößte Eigenwert ist, dass also

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|.$$

Wir nehmen für die folgende, einführende Diskussion ausserdem an, dass sogar $|\lambda_1| > |\lambda_2|$ gilt.

Unter den genannten Voraussetzungen kann man jetzt den Gelfand-Radius von A berechnen. Dazu muss man $|A^n v|$ für einen beliebigen Vektor v und für beliebiges n berechnen. Der Vektor v lässt sich in der Eigenbasis von A zerlegen, also als Summe

$$v = v_1 + v_2 + \dots + v_n$$

schreiben, wobei v_i Eigenvektoren zum Eigenwert λ_i sind oder Nullvektoren. Die Anwendung von A^k ergibt dann

$$A^k v = A^k v_1 + A^k v_2 + \dots + A^k v_n = \lambda_1^k v_1 + \lambda_2^k v_2 + \dots + \lambda_n^k v_n.$$

Für den Grenzwert braucht man die Norm von $A^k v$, also

$$\begin{aligned} |A^k v| &= |\lambda_1^k v_1 + \lambda_2^k v_2 + \dots + \lambda_3 v_3| \\ \Rightarrow \frac{|A^k v|}{\lambda_1^k} &= \left| v_1 + \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \dots + \left(\frac{\lambda_n}{\lambda_1} \right)^k v_n \right|. \end{aligned} \quad (6.16)$$

Da alle Quotienten $|\lambda_i/\lambda_1| < 1$ sind für $i \geq 2$, konvergieren alle Terme auf der rechten Seite von (6.16) ausser dem ersten gegen 0. Folglich ist

$$\lim_{k \rightarrow \infty} \frac{|A^k v|}{\lambda_1^k} = |v_1| \quad \Rightarrow \quad \lim_{k \rightarrow \infty} \frac{|A^k v|^{\frac{1}{k}}}{|\lambda_1|} = \lim_{k \rightarrow \infty} |v_1|^{\frac{1}{k}} = 1.$$

Dies gilt für alle Vektoren v , für die $v_1 \neq 0$ ist. Der maximale Wert dafür wird erreicht, wenn man für v einen Eigenvektor der Länge 1 zum Eigenwert λ_1 einsetzt, dann ist $v = v_1$. Es folgt dann

$$\pi(A) = \lim_{k \rightarrow \infty} \|A^k\|^{\frac{1}{k}} = \lim_{k \rightarrow \infty} |A^k v|^{\frac{1}{k}} = |\lambda_1| = \varrho(A).$$

Damit ist gezeigt, dass im Spezialfall einer diagonalisierbaren Matrix der Gelfand-Radius tatsächlich der Betrag des betragsgrößten Eigenwertes ist.

Blockmatrizen

Wir betrachten jetzt eine $(n+m) \times (n+m)$ -Blockmatrix der Form

$$A = \begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix} \quad (6.17)$$

mit einer $n \times n$ -Matrix B und einer $m \times m$ -Matrix C . Ihre Potenzen haben ebenfalls Blockform:

$$A^k = \begin{pmatrix} B^k & 0 \\ 0 & C^k \end{pmatrix}.$$

Ein Vektor v kann in die zwei Summanden v_1 bestehen aus den ersten n Komponenten und v_2 bestehen aus den letzten m Komponenten zerlegen. Dann ist

$$A^k v = B^k v_1 + C^k v_2. \quad \Rightarrow \quad |A^k v| \leq |B^k v_1| + |C^k v_2| \leq \pi(B)^k |v_1| + \pi(C)^k |v_2|.$$

Insbesondere haben wir das folgende Lemma gezeigt:

Lemma 6.5. Eine diagonale Blockmatrix A (6.17) Blöcken B und C hat Gelfand-Radius

$$\pi(A) = \max(\pi(B), \pi(C))$$

Selbstverständlich lässt sich das Lemma auf Blockmatrizen mit beliebig vielen diagonalen Blöcken verallgemeinern.

Für Diagonalmatrizen der genannten Art sind aber auch die Eigenwerte leicht zu bestimmen. Hat B die Eigenwerte $\lambda_i^{(B)}$ mit $1 \leq i \leq n$ und C die Eigenwerte $\lambda_j^{(C)}$ mit $1 \leq j \leq m$, dann ist das charakteristische Polynom der Blockmatrix A natürlich

$$\chi_A(\lambda) = \chi_B(\lambda)\chi_C(\lambda),$$

woraus folgt, dass die Eigenwerte von A die Vereinigung der Eigenwerte von B und C sind. Daher gilt auch für die Spektralradius die Formel

$$\varrho(A) = \max(\varrho(B), \varrho(C)).$$

Jordan-Blöcke

Nicht jede Matrix ist diagonalisierbar, die bekanntesten Beispiele sind die Matrizen

$$J_n(\lambda) = \begin{pmatrix} \lambda & 1 & & & \\ & \lambda & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & \lambda & 1 \\ & & & & & \lambda \end{pmatrix}, \quad (6.18)$$

wobei $\lambda \in \mathbb{C}$ eine beliebige komplexe Zahl ist. Wir nennen diese Matrizen *Jordan-Matrizen*. Es ist klar, dass $J_n(\lambda)$ nur den n -fachen Eigenwert λ hat und dass der erste Standardbasisvektor ein Eigenvektor zu diesem Eigenwert ist.

In der linearen Algebra lernt man, dass jede Matrix durch Wahl einer geeigneten Basis als Blockmatrix der Form

$$A = \begin{pmatrix} J_{n_1}(\lambda_1) & 0 & \dots & 0 \\ 0 & J_{n_2}(\lambda_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & J_{n_l}(\lambda_l) \end{pmatrix}$$

geschrieben werden kann¹. Die früheren Beobachtungen über den Spektralradius und den Gelfand-Radius von Blockmatrizen zeigen uns daher, dass nur gezeigt werden muss, dass nur die Gleichheit des Gelfand-Radius und des Spektral-Radius von Jordan-Blöcken gezeigt werden muss.

Iterationsfolgen

Satz 6.6. *Sei A eine $n \times n$ -Matrix mit Spektralradius $\varrho(A)$. Dann ist $\varrho(A) < 1$ genau dann, wenn*

$$\lim_{k \rightarrow \infty} A^k = 0.$$

Ist andererseits $\varrho(A) > 1$, dann ist

$$\lim_{k \rightarrow \infty} \|A^k\| = \infty.$$

Beweis. Wie bereits angedeutet reicht es, diese Aussagen für einen einzelnen Jordan-Block mit Eigenwert λ zu beweisen. Die k -te Potenz von $J_n(\lambda)$ ist

$$J_n(\lambda)^k = \begin{pmatrix} \lambda^k & \binom{k}{1}\lambda^{k-1} & \binom{k}{2}\lambda^{k-2} & \dots & \binom{k}{n-1}\lambda^{k-n+1} \\ 0 & \lambda^k & \binom{k}{1}\lambda^{k-1} & \dots & \binom{k}{n-2}\lambda^{k-n+2} \\ 0 & 0 & \lambda^k & \dots & \binom{k}{n-k+3}\lambda^{k-n+3} \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & \lambda^k \end{pmatrix}.$$

Falls $|\lambda| < 1$ ist, gehen alle Potenzen von λ exponentiell schnell gegen 0, während die Binomialkoeffizienten nur polynomiell schnell anwachsen. In diesem Fall folgt also $J_n(\lambda) \rightarrow 0$.

Falls $|\lambda| > 1$ divergieren bereits die Elemente auf der Diagonalen, also ist $\|J_n(\lambda)^k\| \rightarrow \infty$ mit welcher Norm auch immer man man die Matrix misst. \square

Aus dem Beweis kann man noch mehr ablesen. Für $\varrho(A) < 1$ ist die Norm $\|A^k\| \leq M\varrho(A)^k$ für eine geeignete Konstante M , für $\varrho(A) > 1$ gibt es eine Konstante m mit $\|A^k\| \geq m\varrho(A)^k$.

Der Satz von Gelfand

Der Satz von Gelfand ergibt sich jetzt als direkte Folge aus dem Satz 6.6.

Satz 6.7 (Gelfand). *Für jede komplexe $n \times n$ -Matrix A gilt*

$$\pi(A) = \lim_{k \rightarrow \infty} \|A^k\|^{\frac{1}{k}} = \varrho(A).$$

¹Sofern die Matrix komplexe Eigenwerte hat muss man auch komplexe Basisvektoren zulassen.

Beweis. Der Satz 6.6 zeigt, dass der Spektralradius ein scharfes Kriterium dafür ist, ob $\|A^k\|$ gegen 0 oder ∞ konvergiert. Andererseits ändert ein Faktor t in der Matrix A den Spektralradius ebenfalls um den gleichen Faktor, also $\varrho(tA) = t\varrho(A)$. Natürlich gilt auch

$$\pi(tA) = \lim_{k \rightarrow \infty} \|t^k A^k\|^{\frac{1}{k}} = \lim_{k \rightarrow \infty} t \|A^k\|^{\frac{1}{k}} = t \lim_{k \rightarrow \infty} \|A^k\|^{\frac{1}{k}} = t\pi(A).$$

Wir betrachten jetzt die Matrix

$$A(\varepsilon) = \frac{A}{\varrho(A) + \varepsilon}.$$

Der Spektralradius von $A(\varepsilon)$ ist

$$\varrho(A(\varepsilon)) = \frac{\varrho(A)}{\varrho(A) + \varepsilon},$$

er ist also > 1 für negatives ε und < 1 für positives ε . Aus dem Satz 6.6 liest man daher ab, dass $\|A(\varepsilon)^k\|$ genau dann gegen 0 konvergiert, wenn $\varepsilon > 0$ ist und divergiert genau dann, wenn $\varepsilon < 0$ ist.

Aus der Bemerkung nach dem Beweis von Satz 6.6 schliesst man daher, dass es im Fall $\varepsilon > 0$ eine Konstante M gibt mit

$$\begin{aligned} \|A(\varepsilon)^k\| \leq M \varrho(A(\varepsilon))^k &\Rightarrow \|A(\varepsilon)^k\|^{\frac{1}{k}} \leq M^{\frac{1}{k}} \varrho(A(\varepsilon)) \\ &\Rightarrow \pi(A) \leq \varrho(A(\varepsilon)) \underbrace{\lim_{k \rightarrow \infty} M^{\frac{1}{k}}}_{=1} = \varrho(A(\varepsilon)) = \varrho(A) + \varepsilon. \end{aligned}$$

Dies gilt für beliebige $\varepsilon > 0$, es folgt daher $\pi(A) \leq \varrho(A)$.

Andererseits gibt es für $\varepsilon < 0$ eine Konstante m mit

$$\begin{aligned} \|A(\varepsilon)^k\| \geq m \varrho(A(\varepsilon))^k &\Rightarrow \|A(\varepsilon)^k\|^{\frac{1}{k}} \geq m^{\frac{1}{k}} \varrho(A(\varepsilon)) \\ &\Rightarrow \pi(A) \geq \varrho(A(\varepsilon)) \underbrace{\lim_{k \rightarrow \infty} m^{\frac{1}{k}}}_{=1} = \varrho(A(\varepsilon)) = \varrho(A) + \varepsilon. \end{aligned}$$

Dies gilt für beliebige $\varepsilon < 0$, es folgt daher $\pi(A) \geq \varrho(A)$. Zusammen mit $\pi(A) \leq \varrho(A)$ folgt $\pi(A) = \varrho(A)$. \square

6.3.2 Konditionszahl

Der Spektralradius $\varrho(A)$ einer Matrix A gibt darüber Auskunft, ob ein auf A basierendes Iterationsverfahren konvergiert. Er sagt aber nicht einmal, ob die Matrix zum Beispiel regulär. Doch auch diese Information lässt sich aus den Eigenwerten ablesen. Die Matrix ist genau dann regulär, wenn alle Eigenwerte von 0 verschieden sind. Ein Problem entsteht also dann, wenn einzelne Eigenwerte sehr klein sind. Die Kombination besonders grosser und besonders kleiner Eigenwerte ist also ein Indikator, der auf mögliche numerische Probleme hinweisen kann.

Die Eigenwerte von A^{-1} sind die Reziproken der Eigenwerte von A . Ist λ ein Eigenwert von A , dann ist λ^{-1} ein Eigenwert von A^{-1} . Der betragskleinste Eigenwert von A ist der betragsgrösste Eigenwert von A^{-1} . Numerische Probleme werden also dadurch angezeigt, dass $\varrho(A)$ gross ist, oder $\varrho(A^{-1})$ klein.

Definition 6.8. Die Konditionszahl einer Matrix A ist der Quotient

$$\kappa(A) = \frac{\varrho(A)}{\varrho(A^{-1})} = \frac{|\lambda_1|}{|\lambda_n|},$$

wenn λ_1 der betragsgrößte und λ_n der betragskleinste Eigenwert von A ist.

Die Konditionszahl ist also immer ≥ 1 , dieser minimale Wert 1 wird zum Beispiel für die Einheitsmatrix erreicht. Schlechte Kondition tritt auf, wenn die Eigenwerte sehr grosse Betragsunterschiede aufweisen.

Beispiel. Die Kahan-Matrix

$$A = \begin{pmatrix} 1000 & 999 \\ 999 & 998 \end{pmatrix}$$

besteht aus zwei fast gleichen Zeilen, sie ist als fast singulär, die Determinante muss sehr klein sein. Andererseits sind alle Einträge von der Größenordnung 10^3 , man erwartet also einen Spektralradius in der selben Größenordnung. Die Konditionszahl dürfte daher weit grösser als 10^3 sein. Die numerische Rechnung ergibt für die Eigenwerte

$$\lambda_1 = 1998.00050050375, \quad \lambda_2 = -0.000500500375,$$

was eine Konditionszahl von

$$\kappa(A) \approx 3992006 \approx 4 \cdot 10^6$$

ergibt. Diese einfache Matrix hat also sehr schlechte Kondition.

Die Determinante von A ist

$$\det A = 1000 \cdot 998 - 999^2 = -1.$$

Damit kann man auch die Inverse algebraisch leicht angeben:

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} 998 & -999 \\ -999 & 1000 \end{pmatrix}.$$

Die numerische Rechnung offenbart aber die Schwierigkeiten, die die schlechte Kondition verursacht. Das Produkt AA^{-1} müsste die Einheitsmatrix ergeben, die numerische Rechnung mit dem double-Typ ergibt jedoch

$$\begin{pmatrix} 1.00000000000074920 & -0.00000000000052182 \\ 0.0000000000003410 & 1.00000000000020236 \end{pmatrix},$$

der numerische Fehler ist also etwa um den Faktor 10^4 grösser als bei einer gut konditionierten Matrix erwartet werden kann. \circlearrowright

6.4 QR-Zerlegung mit Spiegelungen

Das Orthonormalisierungsproblem verlangt, dass zu gegebenen Vektoren $a_1, \dots, a_n \in \mathbb{R}^l$ orthonormierte Vektoren q_1, \dots, q_n gefunden werden derart, dass

$$a_k = \langle q_1, \dots, q_k \rangle \quad \forall 1 \leq k \leq n. \tag{6.19}$$

Die Bedingung (6.19) bedeutet, dass es Zahlen r_{ik} mit $i \leq k$ derart, dass

$$a_k = q_1 r_{11} + q_2 r_{12} + \cdots + q_k r_{1k} \quad (6.20)$$

Schreiben wir die Komponenten des Vektors a_k als $l \times n$ -Matrix A mit Einträgen a_{ik} und analog für die Vektoren q_k , dann kann (6.20) in Matrixform geschrieben werden als

$$A = QR, \quad \text{mit} \quad R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ 0 & 0 & \dots & r_{3n} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix}$$

Darin ist Q eine $l \times n$ -Matrix und R ist eine $n \times n$ -Matrix. Die Aufgabe, eine Menge von Vektoren $\{a_1, \dots, a_n\}$ zu orthonormieren, ist also gleichbedeutend damit, für die Matrix A eine Zerlegung $A = QR$ zu finden, wobei Q orthogonal und R eine obere Dreiecksmatrix sein soll.

6.4.1 Gram-Schmidt-Orthonormalisierung

Das einfachste und anschaulichste Orthonormalisierungsverfahren, der Gram-Schmidt-Prozess verwendet die Formeln

$$\begin{aligned} q_1 &= \frac{a_1}{|a_1|} \\ q_2 &= \frac{a_2 - (q_1 \cdot a_2)q_1}{|a_2 - (q_1 \cdot a_2)q_1|} \\ q_3 &= \frac{a_3 - (q_1 \cdot a_3)q_1 - (q_2 \cdot a_3)q_2}{|a_3 - (q_1 \cdot a_3)q_1 - (q_2 \cdot a_3)q_2|} \\ &\vdots \\ q_k &= \frac{a_k - (q_1 \cdot a_k)q_1 - \cdots - (q_{k-1} \cdot a_k)q_{k-1}}{|a_k - (q_1 \cdot a_k)q_1 - \cdots - (q_{k-1} \cdot a_k)q_{k-1}|}, \end{aligned}$$

um die orthonormierten Vektoren q_k zu finden. Wegen der Differenzen im Zähler und Nenner besteht die Gefahr von Auslöschung, falls der Winkel zwischen a_k und $\langle a_1, \dots, a_{k-1} \rangle$ sehr klein ist. Der Gram-Schmidt-Prozess ist daher in seiner Grundform nicht immer stabil. In Kapitel 22 wird an einem Simulationsbeispiel gezeigt, wie sich die Instabilität des Gram-Schmidt-Algorithmus äusseren kann.

6.4.2 Spiegelungen

Mit dem Skalarprodukt kann zu jedem Paar a, b von Vektoren gleicher Länge eine Matrix gefunden werden, welche eine Spiegelung beschreibt, die a auf b abbildet, alle Vektoren orthogonal zu a und b aber unverändert lässt.

Sei $n = (b - a)/|b - a|$ der Einheitsvektor in Richtung $b - a$. Es ist $(a - b) \cdot (a + b) = |a|^2 - |b|^2$, also auch $n \cdot (a + b) = 0$.

Die Abbildung

$$s : x \mapsto x - 2n(n \cdot x)$$

hält auf n orthogonale Vektoren x wegen $n \cdot x = 0$ fest. Für die Vektoren a und b ist $(b - a) \cdot a = -(b - a) \cdot b$, so dass

$$\begin{aligned} a &= \underbrace{\frac{1}{2}(a+b)}_{=u} + \underbrace{\frac{1}{2}(a-b)}_{=v} = u+v \\ b &= \frac{1}{2}(a+b) - \frac{1}{2}(a-b) = u-v \end{aligned}$$

gilt mit orthogonalen Vektoren $u = \frac{1}{2}(a+b)$ und $v = \frac{1}{2}(a-b)$. Für die beiden Summanden gilt

$$\begin{aligned} s(u) &= u, \\ s(v) &= -v \end{aligned}$$

und daher

$$\begin{aligned} s(a) &= s(u+v) = s(u)+s(v) = u-v = b, \\ s(b) &= s(u-v) = s(u)-s(v) = u+v = a. \end{aligned}$$

Die Abbildung s vertauscht also die beiden Vektoren, sie ist eine Spiegelung in der von a und b aufgespannten Ebene an einer Geraden mit der Normalen n .

Der Vektor n ermöglicht auch, die lineare Abbildung s mit Hilfe einer Matrix s zu schreiben. Das Skalarprodukt $n \cdot x$ ist das Matrixprodukt $n^t x$, der Vektor $n(n \cdot x)$ als Matrixprodukt $nn^t x$ berechnet werden. Damit wird

$$s(x) = x - 2n(n \cdot x) = Ex - 2(nn^t)x = (E - 2nn^t)x \quad \Rightarrow \quad S = E - 2nn^t$$

die Matrix der Abbildung s . Wir haben damit den folgenden Satz bewiesen.

Satz 6.9. Zu gegebenen Vektoren a und b gleicher, nicht verschwindender Länge gibt es eine orthogonale Matrix

$$S_{a,b} = E - 2nn^t \quad \text{mit } n = (b-a)/|b-a|,$$

die zu einer Spiegelung gehört, die a auf b abbildet und umgekehrt und ausserdem alle Vektoren senkrecht auf a und b fest lässt.

6.4.3 QR-Zerlegung mit Spiegelungen

Aus der QR-Zerlegung $A = QR$ erhält man durch Multiplikation mit Q^t die Gleichung $Q^t A = R$. Die Aufgabe, die QR-Zerlegung zu finden, ist also gleichbedeutend damit, eine Matrix durch Multiplizieren mit orthogonalen Matrizen auf Dreiecksform zu bringen. Das Produkt der orthogonalen Matrizen ist Q^t . In diesem Abschnitt wird ein Algorithmus basierend auf den Matrizen $S_{a,b}$ vorgestellt, der genau dies leistet.

Sei $a \in \mathbb{R}^l$ ein nicht verschwindender Vektor. Dann ist der Vektor $b = (|a|, 0, \dots, 0)^t$ ein Vektor gleicher Länge. Nach Satz 6.9 ist die Matrix $S_{a,b}$ eine Spiegelung, die a auf b abbildet.

Sei jetzt ein Vektor $a \in \mathbb{R}^l$ gegeben und eine Zahl $k < l$. Wir zerlegen den Vektor a in zwei Teile

$$a = \begin{pmatrix} \tilde{a} \\ \bar{a} \end{pmatrix} \quad \text{mit} \quad \tilde{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \end{pmatrix}, \bar{a} = \begin{pmatrix} a_k \\ \vdots \\ a_l \end{pmatrix}.$$

Die Matrix $Q_{\bar{a}}$ bildet \bar{a} auf einen Vektor ab, von dem nur die erste Komponente von 0 verschieden ist. Daraus wird die Matrix

$$Q_a = \begin{pmatrix} E & 0 \\ 0 & Q_{\bar{a}} \end{pmatrix}, \quad (6.21)$$

die den Vektor

$$a = \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \\ a_k \\ \vdots \\ a_n \end{pmatrix} \quad \text{auf} \quad Q_a a = \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \\ |\bar{a}| \\ \vdots \\ 0 \end{pmatrix}$$

abbildet. Q_a lässt Vektoren x , für die $\bar{x} = 0$ ist, unverändert.

Daraus lässt sich jetzt ein QR-Algorithmus konstruieren.

Satz 6.10 (QR-Zerlegung mit Spiegelungen). *Gegeben ist die $l \times n$ -Matrix A mit $n \leq l$. Dann gibt es orthogonale Matrizen Q_1, \dots, Q_n derart, dass in jeder Matrix der Folge*

$$A_0 = A, \quad A_i = Q_i A_{i-1} \quad \text{für } 0 < i \leq n$$

die ersten i Spalten die Form einer oberen Dreiecksmatrix haben. Mit $Q = Q_1^t \dots Q_n^t$ und $R = A_n = Q_n \dots Q_1 A$ gilt $A = QR$.

Beweis. Wir müssen die Matrix Q_i aus A_{i-1} konstruieren. Sei a die i -te Matrix von A_{i-1} , dann setzen wir $Q_i = Q_a$ mit der Matrix Q_a nach (6.21). Die Multiplikation mit Q_i bringt die Spalte i der Matrix A_{i-1} in die für A_i verlangte Form, ohne die Spalten mit Spaltenindex $< i$ zu verändern, da deren Komponenten dort, wo Q_a etwas bewirkt, alle verschwinden. \square

Beispiel. Die Vektoren

$$a_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad a_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \quad a_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

sollen orthonormalisiert werden. Die Matrix Q_1 spiegelt den Vektor

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \text{auf} \quad \begin{pmatrix} \sqrt{3} \\ 0 \\ 0 \end{pmatrix}$$

ab. Die zugehörige Matrix ist

$$Q_1 = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{1}{2} - \frac{1}{2\sqrt{3}} & -\frac{1}{2} - \frac{1}{2\sqrt{3}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{2} - \frac{1}{2\sqrt{3}} & \frac{1}{2} - \frac{1}{2\sqrt{3}} \end{pmatrix} \quad A_1 = Q_1 A = \begin{pmatrix} \sqrt{3} & \frac{2}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & -\frac{1}{\sqrt{3}} & -\frac{1}{2} - \frac{1}{2\sqrt{3}} \\ 0 & -\frac{1}{\sqrt{3}} & \frac{1}{2} - \frac{1}{2\sqrt{3}} \end{pmatrix}.$$

Die zweite Matrix Q_2 ist

$$Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}, \quad R = A_2 = Q_2 A_1 = \begin{pmatrix} \sqrt{3} & \frac{2}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ 0 & \sqrt{\frac{2}{3}} & \frac{1}{\sqrt{6}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Wegen $R = (Q_2 Q_1)A$ folgt $A = (Q_2 Q_1)^t R$ und damit

$$Q = (Q_2 Q_1)^t = \begin{pmatrix} \frac{1}{\sqrt{3}} & -\sqrt{\frac{2}{3}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Damit ist die QR-Zerlegung der Matrix A gefunden. Tatsächlich bilden die Spalten von Q eine orthonormierte Basis von \mathbb{R}^3 , sie stimmt mit der Basis überein, die der Gram-Schmidt-Prozess aus den Spalten der Matrix A ableitet. \circ

Der Nachteil dieser Methode ist, dass die Matrizen $S_{a,b}$ typischerweise sehr viele Einträge haben und dass damit die Matrixprodukte mit $S_{a,b}$ aufwendig zu berechnen sind. Dies kann durch die Verwendung von Drehmatrizen, sogenannten Givens-Rotationen, verbessert werden, wie dies in Kapitel 22 durchgeführt wird.

6.5 Diagonalisierung mit dem Jacobi-Verfahren

Die Diskretisierung linearer partieller Differentialgleichungen wie zum Beispiel der Wellengleichung führen immer auf symmetrische Eigenwertprobleme, also auf Gleichungen der Form $Av = \lambda v$ mit $A = A^t$. Aus der linearen Algebra ist bekannt, dass es in diesem Fall eine orthogonale Matrix O gibt mit

$$\begin{pmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{pmatrix} = O^t A O$$

Die Matrix $O^t A O$ hat also die Basisvektoren $e_i = (0, \dots, 1, \dots, 0)^t$ als Eigenvektoren. O bildet e_i auf den i -ten Eigenvektor von A ab, in der i -ten Spalten von O steht der i -te Eigenvektor von A . Findet man O , kann man daraus die Eigenvektoren ablesen.

Das Jacobi-Verfahren versucht, O als Zusammensetzung von Drehungen in jeweils zwei Dimensionen aufzubauen. Gleichzeitig wird die Matrix A “in place” auf Diagonalform reduziert, so dass man dort die Eigenwerte ablesen kann.

6.5.1 Jacobi-Verfahren in zwei Dimensionen

In zwei Dimensionen hat eine orthogonale Matrix O immer die Form

$$O = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}.$$

Die symmetrische Matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

soll damit auf Diagonalform gebracht werden. In

$$O^t A O = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}$$

müssen die Elemente ausserhalb der Diagonalen zu 0 werden, dann stehen auf der Diagonalen die gesuchten Eigenwerte. Durch Nachrechnen findet man für α die Bedingung

$$\begin{aligned} 0 &= a_{11} \sin \alpha \cos \alpha + a_{12}(\cos^2 \alpha - \sin^2 \alpha) - a_{22} \sin \alpha \cos \alpha \\ &= (a_{11} - a_{22}) \frac{1}{2} \sin 2\alpha + a_{12} \cos 2\alpha \\ \cot 2\alpha &= \frac{a_{22} - a_{11}}{2a_{12}}. \end{aligned}$$

Mit Hilfe der goniometrischen Beziehung

$$\vartheta = \cot 2\alpha = \frac{1 - \tan^2 \alpha}{2 \tan \alpha}$$

kann man sie als quadratische Gleichung für $\tan \alpha$ betrachten, nämlich

$$\tan^2 \alpha + 2\vartheta \tan \alpha - 1 = 0,$$

welche die Lösungen

$$\tan \alpha = \vartheta \pm \sqrt{\vartheta^2 + 1}$$

hat. In der Matrix O werden nur Sinus und Cosinus benötigt, diese kann man durch algebraische Ausdrücke berechnen:

$$\cos \alpha = \frac{1}{\sqrt{1 + \tan^2 \alpha}} \quad (6.22)$$

$$\sin \alpha = \frac{\tan \alpha}{\sqrt{1 + \tan^2 \alpha}} \quad (6.23)$$

Insbesondere sind keine numerisch aufwendigen trigonometrischen Operationen notwendig, ausser der Wurzel in (6.22) und (6.23) sind alle Schritte mit den Grundoperationen durchführbar.

6.5.2 Beliebige Dimension

Für $n > 2$ lässt sich die Reduktion auf Diagonalform nicht mehr in einem Schritt durchführen. Der folgende Algorithmus führt jedoch zum Erfolg.

1. Initialisiere die Matrix O als Einheitsmatrix: $O = I$
2. Für jedes Paar von Indizes (p, q) mit $q > p$ führe die folgenden zwei Schritte aus.
3. Finde eine Matrix

$$O_{pq} = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}$$

wie in Abschnitt 6.5.1, welche die Teilmatrix

$$A_{pq} = \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}$$

auf Diagonalform bringt: $O_{pq}^t A_{pq} O_{pq}$.

4. Bilde die Matrix

$$O' = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & 0 \\ \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cos \alpha & \dots & \sin \alpha & \dots & 0 \\ & \vdots & \ddots & \vdots & & \vdots \\ 0 & -\sin \alpha & \dots & \cos \alpha & \dots & 0 \\ & \vdots & & \vdots & \ddots & \\ 0 & \dots & 0 & 0 & & 1 \end{pmatrix},$$

wobei die trigonometrischen Funktionen in den Zeilen und Spalten mit Indizes p und q stehen.

5. Setzte $A := O''AO'$ und $O := OO'$.

6. Wiederhole das Verfahren ab Schritt 2, falls noch Indexpaare (p, q) mit $|a_{pq}| > \varepsilon$ vorkommen.

Am Ende dieses Verfahrens steht in A die Diagonalmatrix mit den Eigenwerten, in O steht die orthogonale Matrix, die A auf Diagonalform gebracht, sie enthält die Eigenvektoren in den Spalten.

Dieses Verfahren ist nur auf symmetrische Matrizen anwendbar. In Kapitel 23 wird gezeigt, wie der Francis-Algorithmus die Ideen der Abschnitte 6.4 und 6.5 zu einem funktionierenden Eigenwert-Algorithmus kombiniert, der für beliebige Matrizen funktioniert.

Übungsaufgaben

6.1. Berechnen Sie die Eigenwerte der Matrix

$$A = \begin{pmatrix} a & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}.$$

a) Wie gross darf a sein, damit $\varrho(A) < 1$ ist?

b) Berechnen Sie die Konditionszahl $\kappa(A)$ in Abhängigkeit von a .

Lösung. Das charakteristische Polynom ist

$$\chi_A(\lambda) = \det(A - \lambda E) = \begin{vmatrix} a - \lambda & \frac{1}{2} \\ \frac{1}{2} & -\lambda \end{vmatrix} = (a - \lambda)(-\lambda) - \frac{1}{4} = \lambda^2 - a\lambda - \frac{1}{4}$$

und hat die Nullstellen

$$\lambda_{\pm}(a) = \frac{a}{2} \pm \sqrt{\frac{a^2}{4} + \frac{1}{4}} = \frac{a \pm \sqrt{a^2 + 1}}{2}.$$

Abbildung (6.3) zeigt die Graphen von λ_{\pm} in Abhängigkeit von a .

a) Es ist leicht zu sehen, dass es ein um den Nullpunkt symmetrisches Interval von a -Werten gibt, in dem $|\lambda_{\pm}| < 1$ ist. Man kann Interval finden, indem man die Schranken ± 1 für λ in das charakteristische Polynom einsetzt:

$$0 = \chi_A(\pm 1) = 1 \mp a - \frac{1}{4} = \frac{3}{4} \mp a \quad \Rightarrow \quad a = \pm \frac{3}{4}.$$

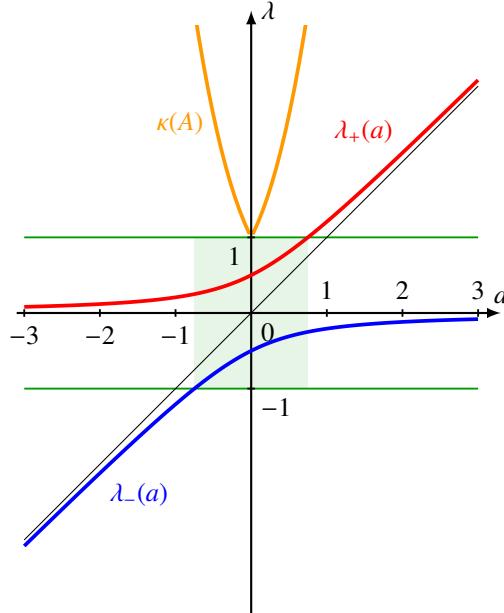


Abbildung 6.3: Eigenwerte λ_{\pm} von A in Abhängigkeit vom Parameter a sowie die Konditionszahl $\kappa(A)$.

Für $a \in (-\frac{3}{4}, \frac{3}{4})$ ist der Spektralradius also $\varrho(A) < 1$. Man kann natürlich auch eine Formel für den Spektralradius angeben, es ist

$$\varrho(A) = \frac{|a| + \sqrt{a^2 + 1}}{2}.$$

Daraus kann man auch ablesen, dass der kleinstmögliche Spektralradius $\frac{1}{2}$ ist.

- b) Die Konditionszahl ist das Verhältnis des grössten zum kleinsten Eigenwert. Für $a > 0$ ist dies $-\lambda_+/\lambda_-$, also

$$\begin{aligned}\kappa(A) &= -\frac{a + \sqrt{a^2 + 1}}{a - \sqrt{a^2 + 1}} = \frac{\sqrt{a^2 + 1} + a}{\sqrt{a^2 + 1} - a} = \frac{(\sqrt{a^2 + 1} + a)^2}{(a^2 + 1) - a^2} \\ &= a^2 + 2a\sqrt{a^2 + 1} + a^2 + 1 = 2a(a + \sqrt{a^2 + 1}) + 1.\end{aligned}\quad (6.24)$$

Für negative a ist λ_- der betragsgrößere Eigenwert. Indem man a wo nötig durch $|a|$ ersetzt, kann man die Formel (6.24) zu einer geraden Funktion machen. Damit wird die Konditionszahl von A zu

$$\kappa(A) = 2(a^2 + |a|\sqrt{a^2 + 1}) + 1$$

für beliebige a . ○

6.2. Betrachten Sie die Matrix A und den Vektor b

$$A = \begin{pmatrix} 3 & 0 & 2 \\ 5 & 1 & -1 \\ 9 & 1 & 2 \end{pmatrix}, \quad \text{und} \quad b = \begin{pmatrix} 11 \\ 16 \\ 31 \end{pmatrix}$$

- a) Finden Sie die Lösung der Gleichung $Ax = b$.
- b) Ist das Jacobi-Verfahren für diese Matrix konvergent?
- c) Ist das Gauss-Seidel-Verfahren für diese Matrix konvergent?
- d) Verifizieren Sie Ihre Resultate von b) und c) numerisch.

Lösung. a) Die Inverse von A und die Lösung x des Gleichungssystems $Ax = b$ sind

$$A^{-1} = \begin{pmatrix} 3 & 2 & -2 \\ -19 & -12 & 13 \\ -4 & -3 & 3 \end{pmatrix} \quad \Rightarrow \quad x = A^{-1}b = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}.$$

- b) Das Jacobi-Verfahren verwendet die Matrix

$$B_{\text{Jacobi}} = D^{-1}(L + R) = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 & 2 \\ 5 & 0 & -1 \\ 9 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{2}{3} \\ 5 & 0 & -1 \\ \frac{9}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Numerisch findet man die Eigenwerte

$$\lambda_1 = 1.84484, \quad \lambda_{2,3} = -0.92242 \pm 0.22927i.$$

Der Spektralradius von B_{Jacobi} ist daher $\varrho(B_{\text{Jacobi}}) = 1.8448$, das Verfahren kann daher nicht konvergieren.

- c) Das Gauss-Seidel-Verfahren verwendet Iteration der Matrix

$$B_{\text{Gauss-Seidel}} = (L + D)^{-1}R = \begin{pmatrix} 3 & 0 & 0 \\ 5 & 1 & 0 \\ 9 & 1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{2}{3} \\ 0 & 0 & -\frac{13}{3} \\ 0 & 0 & -\frac{5}{6} \end{pmatrix}$$

Diese Matrix hat den doppelten Eigenwert 0 und den einfachen Eigenwert $-\frac{5}{6}$. Der Spektralradius ist also $\varrho(B_{\text{Gauss-Seidel}}) = \frac{5}{6} < 1$. Das Gauss-Seidel-Verfahren konvergiert also.

- d) Die Resultate der numerischen Iteration

$$\begin{aligned} x_{k+1} &= (L + D)^{-1}b - (L + D)^{-1}Rx_k && \text{Gauss-Seidel} \\ y_{k+1} &= D^{-1}b - D^{-1}(L + R)y_k && \text{Jacobi} \end{aligned} \tag{6.25}$$

sind in Tabelle 6.3 zusammengestellt. ○

k	Gauss-Seidel			Jacobi		
	x_{1k}	x_{2k}	x_{3k}	y_{1k}	y_{2k}	y_{3k}
	0.0379	0.3621	0.0699	3.0001	2.0001	1.0001
1	3.6201	-2.0306	0.2249	2.9999	1.9996	0.9995
2	3.5167	-1.3588	0.3541	3.0003	1.9998	1.0005
3	3.4306	-0.7990	0.4617	2.9997	1.9988	0.9986
4	3.3589	-0.3325	0.5514	3.0009	2.0002	1.0021
5	3.2990	0.0562	0.6262	2.9986	1.9974	0.9956
6	3.2492	0.3802	0.6885	3.0029	2.0026	1.0076
7	3.2077	0.6502	0.7404	2.9950	1.9930	0.9856
8	3.1731	0.8751	0.7837	3.0096	2.0108	1.0262
9	3.1442	1.0626	0.8197	2.9825	1.9782	0.9514
10	3.1202	1.2188	0.8498	3.0324	2.0387	1.0896
11	3.1001	1.3490	0.8748	2.9403	1.9274	0.8347
12	3.0835	1.4575	0.8957	3.1102	2.1332	1.3050
13	3.0695	1.5479	0.9131	2.7967	1.7540	0.4375
14	3.0580	1.6233	0.9276	3.3750	2.4540	2.0379
15	3.0483	1.6861	0.9396	2.3081	1.1629	-0.9145
16	3.0402	1.7384	0.9497	4.2763	3.5451	4.5322
17	3.0335	1.7820	0.9581	0.6452	-0.8494	-5.5160
18	3.0279	1.8183	0.9651	7.3440	7.2579	13.0212
19	3.0233	1.8486	0.9709	-5.0141	-7.6987	-21.1768
20	3.0194	1.8738	0.9757	17.7846	19.8937	41.9129
21	3.0162	1.8949	0.9798	-24.2752	-31.0099	-74.4774
22	3.0135	1.9124	0.9832	53.3183	62.8989	140.2436
23	3.0112	1.9270	0.9860	-89.8291	-110.3477	-255.8816
24	3.0094	1.9392	0.9883	174.2544	209.2637	474.9046
25	3.0078	1.9493	0.9902	-312.9364	-380.3673	-873.2766
26	3.0065	1.9577	0.9919	585.8511	707.4054	1613.8975
27	3.0054	1.9648	0.9932	-1072.2650	-1299.3579	-2974.5325
28	3.0045	1.9707	0.9944	1986.6883	2402.7924	5490.3713
29	3.0038	1.9755	0.9953	-3656.5809	-4427.0703	-10125.9937
30	3.0031	1.9796	0.9961	6754.3291	8172.9108	18683.6492
∞	3.0000	2.0000	1.0000	∞	∞	∞

Tabelle 6.3: Resultate der Iteration nach Gauss-Seidel und Jacobi mit Hilfe der Iterationsformeln (6.25). Im Falle des stabilen Gauss-Seidel-Verfahrens werden zufällige Startwerte verwendet, im Jacobi-Verfahren liegen die Startwerte in unmittelbarer Nähe der Lösung. Trotzdem divergiert der Vektor im Jacobi-Verfahren ganz offensichtlich, während das Gauss-Seidel-Verfahren konvergiert.

7

Partielle Differentialgleichungen

Elektrische oder magnetische Felder sind Funktionen der Ortskoordinaten und der Zeit. Die Veränderung dieser Felder mit der Zeit hängt ab von der Anwesenheit oder der Bewegung von Ladungen, wie sie von den Maxwellschen Gleichungen

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0}$$

$$\nabla \cdot \vec{B} = 0$$

$$\nabla \times \vec{E} = - \frac{\partial \vec{B}}{\partial t}$$

$$\nabla \times \vec{B} = \mu_0 \vec{j} + \frac{1}{c^2} \frac{\partial \vec{E}}{\partial t}$$

beschrieben werden. Diese Gleichungen zwischen den Komponenten der Felder enthalten die partiellen Ableitungen der gesuchten Funktionen nach Ortskoordinaten und der Zeit. Solche *partiellen Differentialgleichungen* können nicht mit den Methoden gelöst werden, die für gewöhnliche Differentialgleichungen entwickelt worden sind. In diesem Kapitel sollen die wichtigsten Ideen zusammengetragen werden, die besser geeigneten Verfahren zu Grunde liegen.

7.1 Problemstellung

Gewöhnliche Differentialgleichungen werden durch eine einzige Funktion $f: \mathbb{R} \times \mathbb{R}^n : (t, x) \mapsto f(t, x)$ beschrieben, werden. Eine Lösung der Differentialgleichung

$$\frac{dx}{dt} = f(t, x) \tag{7.1}$$

mit der Anfangsbedingung

$$x(t_0) = x_0$$

ist eine Funktion $x(t)$ mit $x(t_0) = x_0$ derart, dass

$$\frac{dx(t)}{dt} = f(t, x(t)).$$

Das Definitionsgebiet der Lösungsfunktion $x(t)$ ist ein Intervall der Form $[t_0, a]$ mit $a > t_0$.

Für eine partielle Differentialgleichung ist die Situation wesentlich komplizierter. Zunächst gibt es Ableitungen der gesuchten Funktion $u(x_1, \dots, x_n)$ nach allen unabhängigen Variablen zu berücksichtigen, so dass eine explizite Form der Differentialgleichung wie in (7.1) grundsätzlich nicht mehr möglich ist. Das Definitionsgebiet ist eine fast beliebige Teilmenge $\Omega \subset \mathbb{R}^n$ eines n -dimensionalen Raumes. Insbesondere kann das Definitionsgebiet sehr viel komplizierter sein als im Falle einer gewöhnlichen Differentialgleichungen. Die Form des Gebietes hat einen wesentlichen Einfluss auf die Lösungen der Differentialgleichung. Schliesslich wird es nicht mehr genügen, Werte in nur einem Randpunkt des Gebietes zu kennen, wie das bei einer gewöhnlichen Differentialgleichung der Fall war. Vielmehr ist es eine nichttriviale Frage, auf welchem Teil des Randes $\partial\Omega$ von Ω welche Funktions- oder Ableitungswerte vorgegeben werden müssen, damit die Lösung der Differentialgleichung eindeutig bestimmt ist.

Der Einfachheit halber betrachten wir in diesem Kapitel nur partielle Differentialgleichungen für eine skalare Funktion $u = u(x_1, \dots, x_n)$. In diesem Abschnitt geht es darum zu klären, wie genau ein solches Problem gestellt werden muss. In Abschnitt 7.1.1 studieren wir ein sinnvolles Definitionsgebiet für die Differentialgleichung. In Abschnitt 7.1.2 klassifizieren wir mögliche Differentialgleichungen bevor wir in Abschnitt 7.1.3 Randbedingungen und in Abschnitt 7.1.4 Lösungen beschreiben.

7.1.1 Gebiet und Rand

Eine partielle Differentialgleichung sucht eine Funktion

$$u: \Omega \rightarrow \mathbb{R}$$

$u(x_1, \dots, x_n)$, für die Beziehungen zwischen den partiellen Ableitungen nach den unabhängigen Variablen gelten. Für alle Punkte einer Menge $\Omega \subset \mathbb{R}^n$ muss als eine Gleichung ähnlich wie zum Beispiel

$$\Delta u = \frac{\partial^2 u}{\partial x_1^2} + \dots + \frac{\partial^2 u}{\partial x_n^2} = 0 \quad \forall (x_1, \dots, x_n) \in \Omega$$

gelten, Δ ist der *Laplace-Operator*. Diese Beobachtung schränkt die Art der Menge Ω bereits ein, wie im Folgenden diskutiert werden soll.

Die Ableitung einer Funktion $u(x_1, \dots, x_n)$ in einem Punkt $x_0 = (x_{0,1}, \dots, x_{0,n})$ ist eine lineare Funktion $Du(x_{0,1}, \dots, x_{0,n})$ derart, dass

$$u(x_1, \dots, x_n) - u(x_{0,1}, \dots, x_{0,n}) = Du(x_{0,1}, \dots, x_{0,n}) \cdot (x - x_0) + o(|x - x_0|).$$

Das Symbol $o(|x - x_0|)$ beschreibt eine Funktion, die schneller als ihr Argument gegen 0 geht, so dass für den Grenzwert des Quotienten

$$\lim_{x \rightarrow x_0} \frac{o(|x - x_0|)}{|x - x_0|} = 0$$

gilt. Der Grenzwert bedeutet, dass es für jedes $\varepsilon > 0$ eine Umgebung $U_\delta(x_0) = \{x \mid |x - x_0| < \delta\}$ gibt derart, dass

$$|u(x) - u(x_0) - Du(x_0) \cdot (x - x_0)| < \varepsilon \cdot |x - x_0| \quad \forall x \in U_\delta(x_0)$$

ist. Dies ist nur sinnvoll, wenn die ganze Umgebung $U_\delta(x_0) \subset \Omega$ im Definitionsgebiet Ω der Differentialgleichung vorhanden ist. Dies führt auf die folgende Definition (siehe auch Abbildung 7.1).

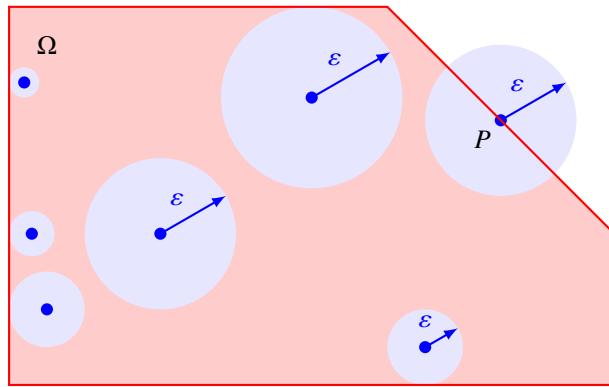


Abbildung 7.1: Jeder innere Punkt einer Menge hat eine ε -Umgebung, die ebenfalls in der Menge enthalten ist. Jede Umgebung des Punktes P enthält aber auch Punkte ausserhalb der Menge Ω , P ist daher ein Randpunkt.

Definition 7.1. Eine Menge Ω heisst offen, wenn mit jedem Punkt $x \in \Omega$ auch eine offene Umgebung $U_\delta(x) \subset \Omega$ darin enthalten ist. Ein Gebiet ist eine offene Menge in \mathbb{R}^n .

Gebiete sind also genau die sinnvollen Definitionsgebiete für eine partielle Differentialgleichung. Es reicht allerdings nicht, dass die Funktion u auf Ω definiert ist, da ja auch noch Randwerte erfüllt werden müssen.

Definition 7.2. Der Abschluss $\bar{\Omega}$ einer Menge $\Omega \subset \mathbb{R}^n$ ist die Menge aller Punkte in \mathbb{R}^n , die Grenzwerte von Folgen in Ω sind. Das Innere $\overset{\circ}{A}$ einer Menge A ist die Menge aller Punkte x derart, dass es eine Umgebung $U_\delta(x)$ gibt, die ganz in A enthalten ist: $U_\delta(x) \subset A$.

Alternativ kann man den Abschluss auch charakterisieren als die Menge aller Punkte $x \in \mathbb{R}^n$, für die jede beliebige Umgebung $U_\delta(x)$ auch Punkte von Ω enthält, also $U_\delta(x) \cap \Omega \neq \emptyset$. Ein Gebiet ist offen, daher ist $\overset{\circ}{\Omega} = \Omega$.

Die Lösung einer partiellen Differentialgleichung wird im Allgemeinen erst festgelegt sein, wenn zusätzlich Werte auf Teilen des “Randes” des Gebietes festgelegt worden sind. Nur Punkte, die als Grenzwerte von Punkten in Ω erreicht werden können, können zu diesem Zweck hinzugezogen werden.

Definition 7.3. Der Rand ∂A einer Menge $A \subset \mathbb{R}^n$ ist $\partial A = \bar{A} \setminus \overset{\circ}{A}$.

Während also eine Differentialgleichung typischerweise auf einem Gebiet Ω definiert ist, muss die gesuchte Lösungsfunktion sogar auf dem Abschluss $\bar{\Omega}$ definiert sein. Die Lösung wird im Allgemeinen erst dadurch festgelegt, dass zusätzlich Werte oder Ableitungen auf Teilen des Randes $\partial\Omega$ vorgegeben werden.

7.1.2 Klassifikation der partiellen Differentialgleichungen

Eine partielle Differentialgleichung beschreibt eine Beziehung zwischen den partiellen Ableitungen der Funktion $u(x_1, \dots, x_n)$. Dies ist auf sehr vielfältige Arten möglich, in die dieser Abschnitt etwas Ordnung bringen soll.

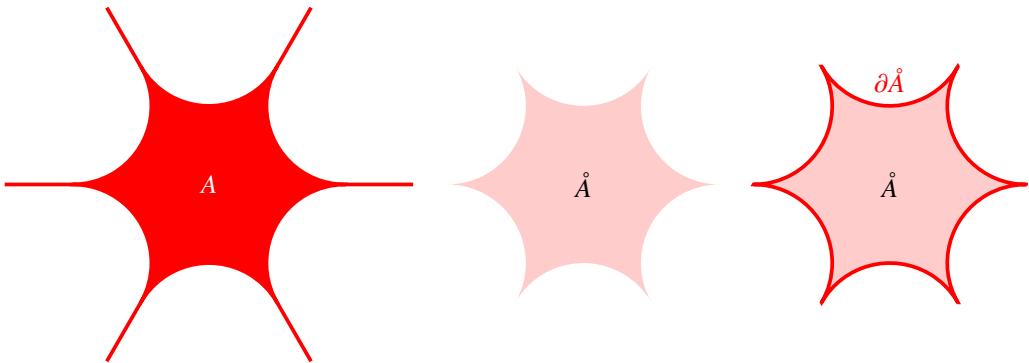


Abbildung 7.2: Inneres und Rand einer Punktmenge A in der Ebene. Die sechs Strahlen der Menge A links können nichts zu den Randwerten einer partiellen Differentialgleichung beitragen, weil sie keinen Einfluss auf Werte in inneren Punkten (Mitte) haben können. Randwerte müssen daher nur auf einem Teil des Randes $\partial\mathring{A}$ des Inneren (rechts) spezifiziert werden.

Ordnung

Wie bei den gewöhnlichen Differentialgleichungen klassifizieren wir auch partielle Differentialgleichungen nach der Ordnung.

Definition 7.4. Die Ordnung einer partiellen Differentialgleichung ist die Ordnung der höchsten Ableitung, die in der Differentialgleichung vorkommt.

Für die folgende Diskussion reicht es, partielle Differentialgleichungen zweiter Ordnung zu betrachten. Die meisten in den Anwendungen vorkommenden Differentialgleichungen sind von dieser Art, so dass dies eine unwesentliche Einschränkung ist. Die Erweiterungen für höhere Ordnung sind offensichtlich.

Linearität

Die Beziehung zwischen den Ableitungen kann beschrieben werden durch eine Funktion

$$F(x_1, \dots, x_n, u, p_1, \dots, p_n, t_{11}, t_{12}, \dots, t_{nn})$$

derart, dass nach der Substitution

$$\begin{aligned} u &\rightarrow u(x_1, \dots, x_n) \\ p_i &\rightarrow \frac{\partial u}{\partial x_i} \\ t_{ij} &\rightarrow \frac{\partial^2 u}{\partial x_i \partial x_j} \end{aligned}$$

die Differentialgleichung

$$F\left(x_1, \dots, x_n, u(x_1, \dots, x_n), \frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \dots, \frac{\partial^2 u}{\partial x_n^2}\right) = 0$$

entsteht. Für höhere Ordnung werden weitere Variablen benötigt, die für die höheren Ableitungen stehen. Die Funktion F kann also dazu verwendet werden, die verschiedenen möglichen partiellen Differentialgleichungen zu klassifizieren.

Eine partielle Differentialgleichung heisst *linear*, wenn die Funktion F linear ist in den Argumenten u , p_i und t_{ij} , wenn also gilt

$$F(\dots, \lambda u' + \mu u'', \dots) = \lambda F(\dots, u', \dots) + \mu F(\dots, u'', \dots) \quad (7.2)$$

$$F(\dots, \lambda p'_i + \mu p''_i, \dots) = \lambda F(\dots, p'_i, \dots) + \mu F(\dots, p''_i, \dots) \quad \forall i \quad (7.3)$$

$$F(\dots, \lambda t'_{ij} + \mu t''_{ij}, \dots) = \lambda F(\dots, t'_{ij}, \dots) + \mu F(\dots, t''_{ij}, \dots) \quad \forall i, j. \quad (7.4)$$

Eine partielle Differentialgleichung heisst *quasilinear*, wenn die Funktion F linear ist in den Argumenten p_i und t_{ij} , wenn also die Bedingungen (7.3) und (7.4) gelten. Linearität in u , also die Bedingung (7.2) ist für eine quasilineare partielle Differentialgleichung nicht verlangt.

Beispiel. Die Differentialgleichung

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = 0$$

zweiter Ordnung wird beschrieben durch die Funktion

$$F(x_1, x_2, u, p_1, p_2, t_{11}, t_{12}, t_{22}) = t_{11} + t_{22}.$$

Diese Funktion ist linear in t_{11} und t_{22} . ○

Beispiel. Die partielle Differentialgleichung erster Ordnung von Burgers (Siehe auch Kapitel 20)

$$\frac{\partial u}{\partial x_1} + u \frac{\partial u}{\partial x_2} = 0 \quad (7.5)$$

wird beschrieben durch die Funktion

$$F(x_1, x_2, u, p_1, p_2) = p_1 + up_2.$$

Diese Funktion ist nicht linear in u und p_2 , aber linear in p_1 und p_2 . (7.5) ist also eine quasilineare partielle Differentialgleichung. ○

Lineare Differentialgleichungen zweiter Ordnung

Ein für die Anwendungen besonders wichtiger Fall sind lineare Differentialgleichungen zweiter Ordnung. Für eine solche Gleichung ist die Funktion F immer von der Form

$$F(x_i, u, p_i, t_{ij}) = \sum_{i,j=1}^n a_{ij}(x_1, \dots, x_n) t_{ij} + \sum_{i=1}^n b_i(x_1, \dots, x_n) p_i + c(x_1, \dots, x_n) u$$

Die Koeffizienten a_{ij} , b_i und c dürfen also von den Variablen x_1, \dots, x_n abhängen, nicht aber von u oder den Ableitungen. Die Differentialgleichung hat also die Form

$$\sum_{i,j=1}^n a_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^n b_i \frac{\partial u}{\partial x_i} + cu = f \quad (7.6)$$

Da es in den zweiten Ableitungen nicht auf die Reihenfolge ankommt, kann die Matrix a_{ij} immer symmetrisch gewählt werden. Die Matrix $A = (a_{ij})$ heisst die *Symbolmatrix* der Differentialgleichung.

Es stellt sich heraus, dass die Terme zweiter Ordnung, also die Matrix A , das Verhalten der Lösung wesentlich beeinflussen. Eine symmetrische Matrix kann durch eine Drehung immer in eine Diagonalmatrix transformiert werden. Durch Wechsel des Koordinatensystems kann man also erreichen, dass

$$A = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

ist. In diesen Koordinaten sind es nur noch die Eigenwerte $\lambda_1, \dots, \lambda_n$ der Matrix A , die das Verhalten der Lösung bestimmen.

Definition 7.5. Die Differentialgleichung (7.6) heisst elliptisch, wenn alle Eigenwerte positiv sind. Sie heisst hyperbolisch, wenn alle Eigenwerte bis auf einen negativen positiv sind. Sie heisst parabolisch, wenn alle Eigenwerte bis auf einen verschwindenden positiv sind.

Beispiel. Die Wellengleichung

$$\frac{1}{a^2} \frac{\partial^2 u}{\partial t^2} = \Delta u \quad \Rightarrow \quad \Delta u - \frac{1}{a^2} \frac{\partial^2 u}{\partial t^2} = 0$$

hat die Symbolmatrix

$$A = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & -\frac{1}{a^2} \end{pmatrix}$$

n positive und einem negativen Eigenwert, diese Gleichung ist also hyperbolisch. Die Lösungen zeigen Wellencharakter und die Werte und ersten Ableitungen von u zu einem Zeitpunkt t_0 bestimmen das Verhalten der Lösung vollständig. Eine hyperbolische Differentialgleichung hat also eine natürlich “Entwicklungsrichtung”, das Gebiet kann in Richtung “Zukunft” offen sein, ohne dass die Eindeutigkeit der Lösung dadurch gefährdet wird. \circ

Beispiel. Die Gleichung für das Potential einer Ladungsverteilung

$$\Delta u = f \quad \Rightarrow \quad A = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} = E$$

hat nur positive Eigenwerte, sie ist also elliptisch. Die Lösungen dieser Gleichung sind nur dann bestimmt, wenn Werte von u auf dem gesamten Rand des Gebietes vorgegeben werden. Es ist also nicht möglich, eine “Zeitrichtung” für die Entwicklung der Lösung einer solchen Differentialgleichung zu finden. \circ

Die Verschiedenartigkeit der Lösungen in Abhängigkeit vom Typ der Differentialgleichung hat zur Folge, dass verschiedene Lösungsverfahren zum Einsatz kommen müssen. Die Klassifikation einer Differentialgleichung ist also Vorbedingung für die Wahl eines geeigneten Lösungsverfahrens.

7.1.3 Randbedingungen

Die Lösung einer partiellen Differentialgleichung ist erst eindeutig bestimmt, wenn Werte von u oder von Ableitungen von u auf geeignet gewählten Teilen des Randes $\partial\Omega$ des Gebietes Ω vorgegeben sind. Die Theorie der partiellen Differentialgleichungen studiert ausführlich, welche Art von Randbedingungen wo auf dem Rand zu spezifizieren sind.

Dirichlet-Randbedingungen

Werte der Funktion auf dem Rand können vorgegeben werden, indem eine Funktion $g: \partial\Omega \rightarrow \mathbb{R}$ spezifiziert wird, so dass für alle Punkte $x \in \partial\Omega$ die Gleichung $u(x) = g(x)$ gilt. Diese Art von Randbedingungen heisst *Dirichlet-Randbedingungen*.

Neumann-Randbedingungen

Die Theorie des Anfangswertproblems für gewöhnliche Differentialgleichungen besagt, dass die Lösung einer Differentialgleichung n -ter Ordnung erst festgelegt ist, wenn der Anfangswert und $n - 1$ -Ableitungen vorgegeben sind. Es ist zu erwarten, dass es auch bei gewissen partiellen Differentialgleichungen der Ordnung $n \geq 2$ nötig sein wird, Ableitungen auf dem Rand vorzugeben.

Im Allgemeinen wird es nicht genügen, nur Ableitungen vorzugeben, da sie Funktionen nur bis auf eine Konstante festlegen können. Es wird also nötig sein, mindestens in einem Punkt zusätzlich einen Funktionswert vorzugeben.

Es ist jedoch nicht sinnvoll, beliebige Ableitungen auf dem Rand vorzugeben. Wir illustrieren dies an einem einfachen Beispiel. Als Gebiet wählen wir $\Omega = \{(x, y) \in \mathbb{R}^2 \mid x > 0\}$, der Rand ist also die y -Achse. Nehmen wir an, dass Werte der Ableitung $\partial u / \partial y$ auf der y -Achse vorgegeben sind, also

$$\frac{\partial u}{\partial y}(0, y) = g(y).$$

Ausserdem nehmen wir an, dass der Funktionswert $u(0, 0) = u_0$ vorgegeben ist. Für die Funktion $f(y) = u(0, y)$ gilt daher $f'(y) = g(y)$ und $f(0) = u_0$. Daraus lässt sich die Funktion f aber durch das Integral

$$f(y) = u_0 + \int_0^y g(\eta) d\eta$$

bestimmen. Die Vorgabe der Ableitung $\partial u / \partial y$ auf dem Rand und eines Wertes ist also gleichbedeutend mit der Vorgabe aller Werte $f(y)$ auf dem Rand. Statt der Ableitungen $\partial u / \partial y$ hätten wird daher auch Dirichlet-Randbedingungen $u(0, y) = f(y)$ vorgeben können. Es ist also nur sinnvoll, die Ableitung $\partial u / \partial x$ auf dem Rand vorzugeben.

Weiter oben hat ein spezielles Beispiel bezeigt, dass Ableitungen entlang des Randes gegenüber Dirichlet-Randbedingungen keine neue Information liefern können. Nur die Ableitung in die Richtung senkrecht auf den Rand kann zusätzliche Information liefern. Dazu muss der Rand des Gebietes ausreichend glatt sein, so dass die Normale auf den Rand wohldefiniert ist.

Definition 7.6. Ist u eine Funktion, die auf $\bar{\Omega}$ definiert ist. Sei n die Normale auf den Rand $\partial\Omega$ in einem Punkt $x \in \partial\Omega$. Die Normalableitung

$$\frac{\partial u}{\partial n} = \lim_{t \rightarrow 0^+} \frac{u(x + tn) - u(x)}{t}$$

in x ist die Richtungsableitung der Funktion u in Richtung n .

Im Falle der Wellengleichung

$$\frac{\partial^2 u}{\partial x^2} - \frac{1}{a^2} \frac{\partial^2 u}{\partial t^2} = 0$$

beschreibt $u(t, x)$ zum Beispiel die Auslenkung einer gespannten Saite aus der Ruhelage. Es ist aus physikalischen Überlegungen klar, dass die Bewegung der Saite erst dann festgelegt ist, wenn Auslenkung und Geschwindigkeit der Saite zur Zeit $t = 0$ vorgegeben werden. Die Vorgabe der Auslenkung zur Zeit $t = 0$

$$u(0, x) = f(x)$$

ist eine Dirichlet-Randbedingung. Die Richtung n der Zeitachse ist senkrecht auf dem Rand $t = 0$, die Zeitableitung von u ist also genau eine Normalableitung. Die Vorgabe der Geschwindigkeit

$$\frac{\partial u}{\partial n}(0, x) = \frac{\partial u}{\partial t}(0, x) = g(x)$$

ist also eine Vorgabe der Normalableitung.

Definition 7.7. Die Vorgabe der Normalableitung auf dem Rand $\partial\Omega$

$$\frac{\partial u}{\partial n}(x) = h(x) \quad \forall x \in \partial\Omega$$

heisst eine Neumann-Randbedingung.

7.1.4 Lösungen

Ein vollständig gestelltes Problem mit partiellen Differentialgleichungen beginnt also immer mit einer Definition des Gebietes Ω , also einer offenen Menge in \mathbb{R}^n . Die Differentialgleichung wird gegeben durch eine Funktion F wie in Abschnitt 7.1.2 dargestellt. Ausserdem müssen Randbedingungen vorgegeben werden, wie in Abschnitt 7.1.3 dargestellt. Gesucht ist dann eine Funktion u , welche alle diese Bedingungen erfüllen soll. Dazu muss die Funktion nicht nur in Ω definiert sein, sondern auch auf dem Rand.

Definition 7.8. Eine Lösung einer partiellen Differentialgleichung ist eine Funktion

$$u: \bar{\Omega} \rightarrow \mathbb{R} : (x_1, \dots, x_n) \mapsto u(x_1, \dots, x_n)$$

derart, dass die Differentialgleichung im Inneren, also in Ω , erfüllt ist, und die Randbedingungen auf $\partial\Omega$.

7.2 Finite Differenzen

Jedes Lösungsverfahren für partielle Differentialgleichungen muss die unendlich vielen Freiheitsgrade, die eine Funktion $u: \Omega \rightarrow \mathbb{R}$ enthalten kann, auf eine endlich Zahl von Variablen reduzieren, für die sich ein Gleichungssystem aufstellen lässt, welches dann mit einer der früher studierten Methoden gelöst werden kann.

7.2.1 Gitter und Ableitungen

Eine einfache Methode, die Funktion u eine endliche Menge von Parametern zu reduzieren, ist, nur die Werte in einzelnen Punkten des Gebietes Ω zu verwenden. Als Beispiel betrachten wir ein Gebiet $\Omega \subset \mathbb{R}^2$ in der Ebene. Dazu betrachten wir die Punkte

$$x_{ik} = (ih_x, kh_y) \in \mathbb{R}^2 \quad i, k \in \mathbb{Z},$$

sie bilden ein Gitter Γ mit Gitterkonstante oder Schrittweite h_x in x -Richtung und h_y in y -Richtung.

Die Funktion $u(x, y)$ nimmt in den Punkten x_{ik} die Werte $u_{ik} = u(x_{ik})$ an. Eine approximative Lösung der Differentialgleichung ist also die Bestimmung der Werte u_{ik} für die Punkte x_{ik} , die in Ω liegen, für die also $x_{ik} \in \Omega$ gilt. Dazu müssen jetzt die Randbedingungen und die Differentialgleichung in Gleichungen für die Unbekannten u_{ik} übersetzt werden.

Dirichlet-Randbedingungen

Dirichlet-Randbedingungen geben die Werte auf dem Rand vor. Liegt ein Punkt x_{ik} des Gitters Γ auf dem Rand $\partial\Omega$ von Ω , dann geben die Dirichlet-Randbedingungen den Wert dieser Variablen vor. Es ist daher anzustreben, das Gitter Γ so zu wählen, dass der Rand durch Gitterpunkte verläuft. Ein gekrümmter Rand wird daher im Allgemeinen durch eine Kurve durch die Gitterpunkte approximiert werden müssen.

Erste Ableitungen

Die naheliegendste Approximation für die Differentialgleichung besteht darin, die Ableitungen durch Differenzenquotienten zu ersetzen (siehe Abbildung 7.3). Mit der oben eingeführten Notation können die ersten Ableitungen durch die sogenannten *Vorwärtsdifferenzen*

$$\frac{\partial u}{\partial x}(x_{ik}) \approx \frac{u(x_{i+1,k}) - u(x_{ik})}{h_x} \quad \text{und} \quad \frac{\partial u}{\partial y}(x_{ik}) \approx \frac{u(x_{i,k+1}) - u(x_{ik})}{h_y} \quad (7.7)$$

approximiert werden (Abbildung 7.3 Mitte). Die Genauigkeit der Approximation kann offenbar verbessert werden, indem h_x und h_y verkleinert werden.

Der Fehler der Approximation (7.7) ergibt sich aus dem Mittelwertsatz der Differentialrechnung. Es gibt Zahlen ξ und η zwischen ih_x und $(i+1)h_x$ bzw. kh_y und $(k+1)h_y$ derart, dass

$$\frac{u(x_{i+1,k}) - u(x_{ik})}{h_x} = \frac{\partial u}{\partial x}(\xi, kh_y) \quad \text{und} \quad \frac{u(x_{i,k+1}) - u(x_{ik})}{h_y} = \frac{\partial u}{\partial y}(ih_x, \eta).$$

Dies zeigt, dass die Approximation (7.7) nicht für die Werte der Ableitungen im Punkt x_{ik} repräsentativ sein kann. Alternativ könnten statt der Vorwärtsdifferenzen die *Rückwärtsdifferenzen*

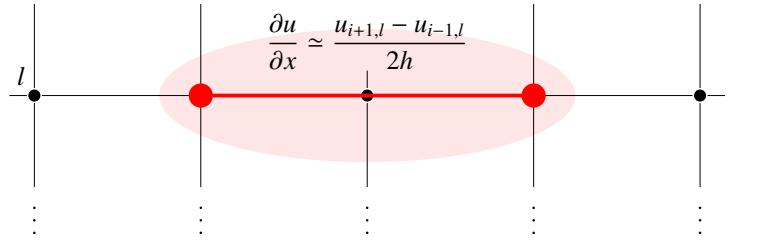
$$\frac{\partial u}{\partial x}(x_{ik}) \approx \frac{u(x_{ik}) - u(x_{i-1,k})}{h_x} \quad \text{und} \quad \frac{\partial u}{\partial y}(x_{ik}) \approx \frac{u(x_{ik}) - u(x_{i,k-1})}{h_y} \quad (7.8)$$

verwendet werden (Abbildung 7.3 unten). Die Genauigkeit wird dadurch jedoch nicht verbessert, die Punkte (ξ, kh_y) und (ih_x, η) , an dem diese Ableitungswerte angenommen werden, liegen jetzt einfach links bzw. unterhalb von x_{ik} .

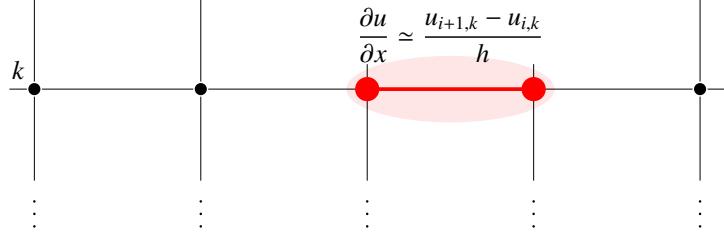
Die Vorwärtsdifferenzen (7.7) sind also genauso fehlerhaft wie die Rückwärtsdifferenzen (7.8), wenngleich in eine andere Richtung. Ein Mittelweg könnte ein Differenzenquotient

$$\frac{\partial u}{\partial x}(x_{ik}) = \frac{u_{i+1,k} - u_{i-1,k}}{2h_x} \quad \text{und} \quad \frac{\partial u}{\partial y}(x_{ik}) = \frac{u_{i,k+1} - u_{i,k-1}}{2h_y}$$

Symmetrische Differenz:



Vorwärtsdifferenz:



Rückwärtsdifferenz:

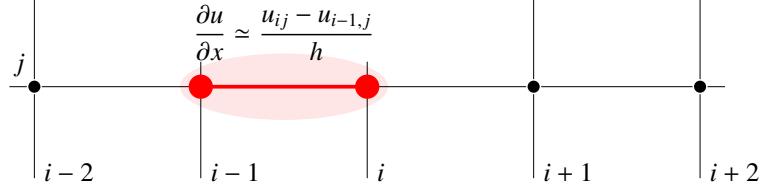


Abbildung 7.3: Approximation der ersten Ableitung mit verschiedenen Differenzausdrücken

über ein symmetrisches Intervall der doppelten Länge. Diese sogenannten *symmetrische Differenzen* (Abbildung 7.3) sind eher repräsentativ für die Steigung im Punkt x_{ik} , dafür ist die Genauigkeit wegen des doppelt so langen Intervalls kleiner.

Beispiel. Um die Unterschiede zwischen den Fehlern der verschiedenen Differenzapproximationen besser zu verstehen, approximieren wir die Ableitungen der Funktion $f(x) = x^2$ im Punkt x_0 und bestimmen den Fehler sowie den Punkt, in dem die Approximation den korrekten Ableitungswert annimmt. Die Vorwärts-, Rückwärts- und symmetrischen Differenzen sind

$$\begin{aligned}
 f'(x_0) &\approx \frac{f(x_0 + h) - f(x_0)}{h} = \frac{(x_0 + h)^2 - x_0^2}{h} = 2x_0 + h &= f'(x_0 + \frac{1}{2}h) \\
 &\approx \frac{f(x_0) - f(x_0 - h)}{h} = \frac{x_0^2 - (x_0 - h)^2}{h} = 2x_0 - h &= f'(x_0 - \frac{1}{2}h) \\
 &\approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} = \frac{(x_0 + h)^2 - (x_0 - h)^2}{2h} = \frac{4x_0 h}{2h} = 2x_0 &= f'(x_0).
 \end{aligned}$$

Für eine quadratische Funktion liefert also die symmetrische Differenz den exakten Wert der Ableitung im Punkt x_0 , während die Vorwärts- und Rückwärtsdifferenzen die Ableitungen in Punkten genau in der Mitte zwischen den Gitterpunkten rechts bzw. links von x_0 (Abbildung 7.4). \circlearrowright

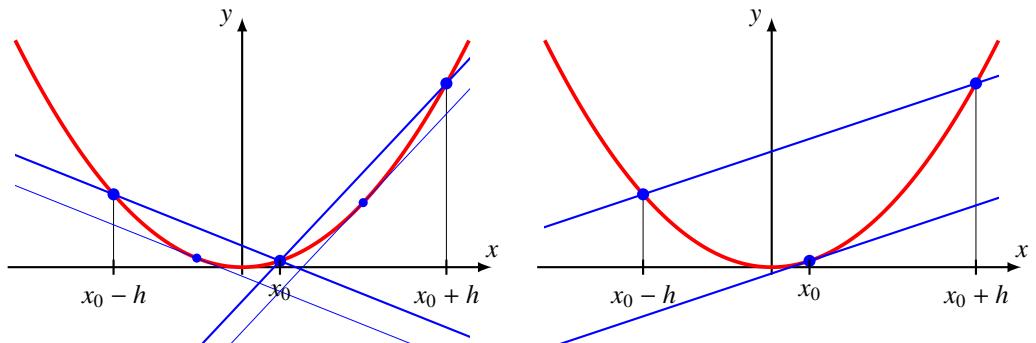


Abbildung 7.4: Differenzenquotienten für die Funktion $f(x) = x^2$. Die Vorwärts- und Rückwärtendifferenzen im Punkt x_0 ergeben Approximationen für die Ableitungen, die mit der wahren Ableitung im Punkt $x_0 \pm \frac{h}{2}$ übereinstimmen. Die symmetrische Differenz im Punkt x_0 ergibt genau die Steigung von $f(x)$ im Punkt x_0 .

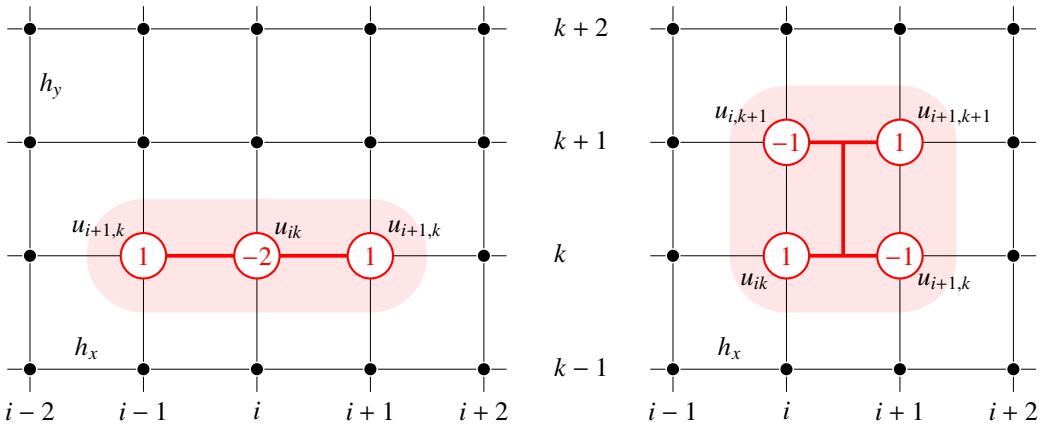


Abbildung 7.5: Differenzenquotienten für die zweiten Ableitungen. Links die zweite Ableitung nach x , rechts die gemischte Ableitung. In den roten Kreisen das Gewicht des zugehörigen Wertes der Funktion im Differenzenquotienten.

Zweite Ableitungen

Eine Approximation für die zweite Ableitung können wir als Differenzenquotient aus der Vorwärts- und der Rückwärtsdifferenz erhalten:

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2}(x_{ik}) &\approx \frac{1}{h_x} \left(\frac{\partial u}{\partial x}(x_{i-\frac{1}{2},k}) - \frac{\partial u}{\partial x}(x_{i+\frac{1}{2},k}) \right) = \frac{1}{h_x} \cdot \left(\frac{u_{i+1,k} - u_{ik}}{h_x} - \frac{u_{ik} - u_{i-1,k}}{h_x} \right) \\ &= \frac{u_{i+1,k} - 2u_{ik} + u_{i-1,k}}{h_x^2}.\end{aligned}$$

Das Problem wird aber schwieriger, wenn eine gemischte Ableitung approximiert werden soll. Hier kann jede beliebige Kombination von Vorwärts- und Rückwärtsdifferenzen verwendet werden. Mit Vorwärtsdifferenzen erhält man zum Beispiel

$$\begin{aligned}\frac{\partial^2 u}{\partial x \partial y}(x_{ik}) &\approx \frac{1}{h_x} \left(\frac{\partial u}{\partial y}(x_{i+1,k}) - \frac{\partial u}{\partial y}(x_{ik}) \right) \\ &= \frac{1}{h_x} \left(\frac{u_{i+1,k+1} - u_{i+1,k}}{h_y} - \frac{u_{i,k+1} - u_{ik}}{h_y} \right) \\ &= \frac{1}{h_x h_y} (u_{i+1,k+1} - u_{i+1,k} - u_{i,k+1} + u_{ik}).\end{aligned}$$

Die in Abbildung 7.5 dargestellten ‘‘Schablonen’’ zeigen schematisch, wie die Ableitungen berechnet werden. In den roten Kreisen um die beteiligten Knotenvariablen stehen die Gewichte, mit denen die Knotenvariablen multipliziert werden müssen.

Neumann-Randbedingungen

Neumann-Randbedingungen geben die Ableitungen in Richtung der Normalen auf dem Rand vor. Die Diskretisation auf ein Gitter führt dazu, dass der Rand aus geraden Teilstücken besteht, wo eine Normalenrichtung leicht zu definieren ist, und Teilstücken, wo zusätzlicher Aufwand getrieben werden muss, überhaupt die Richtung der Normalen zu definieren. Für gerade Teilstücke des Randes können für die Approximation der Normalableitung Vorwärts- oder Rückwärtsdifferenzen verwendet werden.

Beispiel. Zur Illustration des Vorgehens approximieren wir die Normalableitungen für das Rechteckgebiet mit Rändern $x = 0$, $x = Nh_x$, $y = 0$ und $y = Mh_y$, welches in Abbildung 7.6 dargestellt ist. In Punkten x_{0k} und x_{Nk} auf den vertikalen Rändern oder in Punkten x_{i0} und x_{iM} auf den horizontalen Rändern können wir Vorwärts- bzw. Rückwärtsdifferenzen verwenden:

$$\begin{aligned}\frac{\partial u}{\partial n}(x_{0k}) &= \frac{\partial u}{\partial x}(x_{0k}) \approx \frac{u_{1k} - u_{0k}}{h_x} && \text{und} && \frac{\partial u}{\partial n}(x_{Nk}) &= \frac{\partial u}{\partial x}(x_{Nk}) \approx \frac{u_{Nk} - u_{N-1,k}}{h_x}, \\ \frac{\partial u}{\partial n}(x_{i0}) &= \frac{\partial u}{\partial y}(x_{i0}) \approx \frac{u_{i1} - u_{i0}}{h_y} && \text{und} && \frac{\partial u}{\partial n}(x_{iM}) &= \frac{\partial u}{\partial y}(x_{iM}) \approx \frac{u_{iM} - u_{i,M-1}}{h_y}.\end{aligned}$$

Natürlich gelten die oben formulierten Vorbehalte bezüglich der Zuverlässigkeit dieser Approximation, wir haben aber nicht die Möglichkeit, symmetrische Differenzen zu verwenden, da keine Funktionswerte ausserhalb des Gebietes bekannt sind. \circ

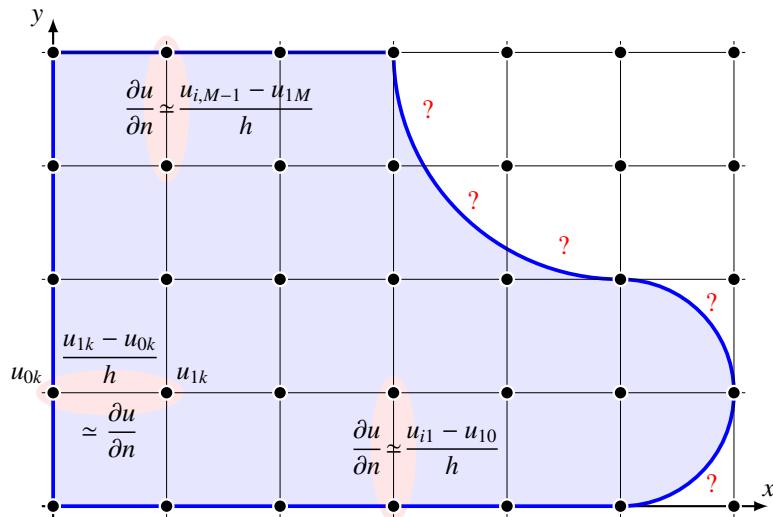


Abbildung 7.6: Neumann-Randbedingungen sind in einem diskretisierten Gebiet einfach auszudrücken, solange die Gitterlinien mit dem Rand des Gebietes zusammenfallen. Sie ergeben eine zusätzliche Gleichung zwischen den rot hinterlegten Punkten. Für die runden Teile des Randes gibt es keine einfachen Lösungen.

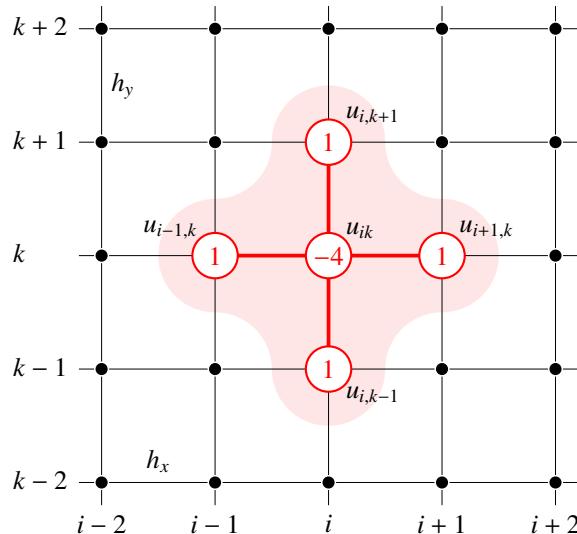


Abbildung 7.7: Approximation des Laplace-Operators mit Summen von symmetrischen Differenzen. In den roten Kreisen die Gewichte, mit denen die Knotenvariablen multipliziert werden müssen, um den Ausdruck (7.10) zu ergeben.

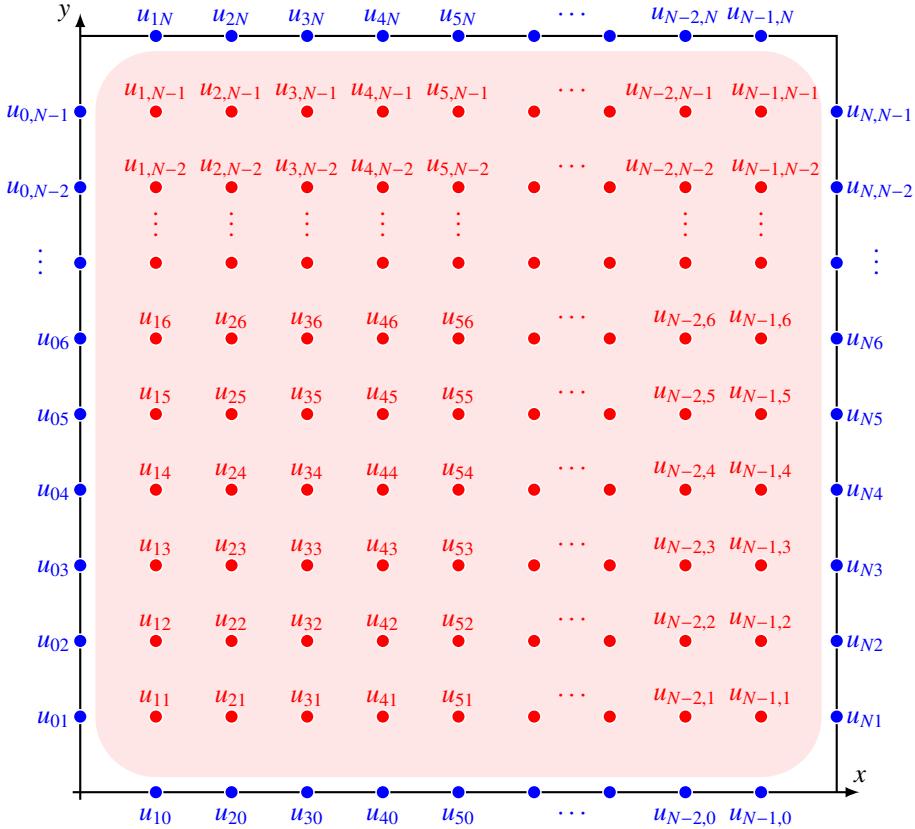


Abbildung 7.8: Diskretisiertes Gebiet für die Differentialgleichung. Die Werte in den blauen Punkten auf dem Rand sind durch die Randbedingungen gegeben, nur die Werte in den roten Punkten müssen bestimmt werden. (7.9).

Das Poisson-Problem

Das Poisson-Problem ist die Differentialgleichung

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \quad (7.9)$$

auf einem Gebiet Ω , wobei wir als Beispiel ein Quadrat $\Omega = (0, 1) \times (0, 1)$ wählen. Die abstrakte Theorie sagt, dass die Lösung der Differentialgleichung eindeutig bestimmt ist, Randwerte $u(x) = g(x)$ für $x \in \partial\Omega$ vorgegeben werden.

Wir diskretisieren das Gebiet (Abbildung 7.8) mit Hilfe des Gitters mit Gitterkonstanten $h = h_x = h_y = 1/N$. Wir erhalten die $(N+1)^2$ Unbekannten u_{ik} mit $0 \leq i \leq N$ und $0 \leq k \leq N$. Die Randbedingungen legen die Werte

$$u_{0k} = g(0, kh)$$

$$u_{Nk} = g(1, kh)$$

$$1 \leq k \leq N$$

$$u_{i0} = g(ih, 0)$$

$$u_{iN} = g(ih, N)$$

$$1 \leq i \leq N$$

fest. $4N$ Unbekannte sind also bereits bestimmt, es bleiben noch $(N+1)^2 - 4N = N^2 - 2N + 1 = (N-1)^2$ innere Werte zubestimmen.

Die Differentialgleichung kann für jeden Punkt im Inneren des Gebietes Ω aufgestellt werden (die roten Punkte in Abbildung 7.8), es gilt

$$\begin{aligned}\Delta u &= \frac{\partial^2 u}{\partial x^2}(u_{ik}) + \frac{\partial^2 u}{\partial y^2}(u_{ik}) = \frac{u_{i+1,k} - 2u_{ik} + u_{i-1,k}}{h^2} + \frac{u_{i,k+1} - 2u_{ik} + u_{i,k-1}}{h^2} \\ f_{ik} &= f(x_{ik}) = \frac{1}{h^2}(u_{i+1,k} + u_{i-1,k} + u_{i,k+1} + u_{i,k-1} - 4u_{ik}).\end{aligned}\quad (7.10)$$

Alle diese $(N-1)^2$ Gleichungen sind linear. Insbesondere haben wir gleich viele Gleichungen wie Unbekannte und dürfen daher davon ausgehen, dass, wie sich auch beweisen lässt, das lineare Gleichungssystem (7.10) für die verbleibenden Unbekannten regulär ist. Die Diskretisierung führt also die Lösung der partiellen Differentialgleichung auf die Lösung eines linearen Gleichungssystems zurück.

7.2.2 Wärmeleitungsgleichung

Als etwas ausführlicheres Beispiel soll in den folgenden Abschnitten das Wärmeleitungsproblem auf einem Stab mit verschiedenen Randbedingungen und Methoden gelöst werden. Dieser Abschnitt beginnt damit, das Problem und seine Diskretisierung zu formulieren. In den folgenden Abschnitten werden die Gleichungen dann für verschiedene Randbedingungen numerisch gelöst.

Die Wärmeleitungsgleichung

Die Temperaturverteilung $u(x, t)$ auf einem Stab mit x -Koordinaten zwischen 0 und 1 zur Zeit t wird durch eine partielle Differentialgleichung auf dem Gebiet

$$\Omega = \{(x, t) \mid 0 < x < 1 \wedge 0 < t\}$$

beschrieben. In der Wärmeleitungsgleichung

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2} \quad (7.11)$$

ist die κ eine Konstante, die Wärmeleitfähigkeit und Wärmekapazität des Materials charakterisiert.

Randbedingungen

Zudem müssen Randbedingungen zur Zeit $t = 0$ und an den Enden des Stabes bei $x = 0$ oder $x = 1$ erfüllt sein. Zur Zeit $t = 0$ muss die initiale Temperaturverteilung $f(x)$ spezifiziert werden.

Dirichletrandbedingungen

$$u(x, 0) = f(x) \quad x \in [0, 1]$$

am Rand des Intervalls bedeuten, dass die Enden des Stabes mit Wärmereservoirs verbunden sind, welche ihn auf einer vorgegebenen Temperatur halten.

Neumann-Randbedingungen

$$\frac{\partial u}{\partial n}(x, t) = \frac{\partial u}{\partial x}(x, t) = g(x) \quad x \in \{0, 1\}.$$

spezifizieren den Temperaturgradienten am Rande und legen damit fest, wieviel Wärmeenergie in den Stab hineinfließt oder ihn verlässt. Im Falle $g(x) = 0$ verschwindet der Gradient und Wärmefluss ist unterbunden. Dies entspricht einem thermisch isolierten Stab.

Energie

Die physikalische Interpretation der Gleichung (7.11) erlaubt, das Verhalten der Lösung abzuschätzen. Dazu berechnen wir die Gesamtenergie

$$U(t) = \int_0^1 u(x, t) dx,$$

die im Intervall enthalten ist. Die Wärmeleitungsgleichung 7.11 erlaubt nun, die Änderung der Energie mit der Zeit zu bestimmen. Es gilt

$$\frac{dU(t)}{dt} = \frac{d}{dt} \int_0^1 u(x, t) dx = \int_0^1 \frac{\partial u}{\partial t}(x, t) dx = \kappa \int_0^1 \frac{\partial^2 u}{\partial x^2}(x, t) dx = \kappa \left[\frac{\partial u}{\partial x} \right]_0^1.$$

Daraus lässt sich zum Beispiel ablesen, dass sich die Energie in einem isolierten Stab, wo die Ableitungen an den Intervallenden verschwinden, nicht ändert. Anders ausgedrückt: die Energie bleibt dann konstant, wenn die Steigung in beiden Enden gleich gross ist, was gleichbedeutend ist damit, dass der Wärmefluss durch beide Intervallenden gleich gross ist.

In allen Fällen kann man das Verhalten der Lösung für $t \rightarrow \infty$ bestimmen. Dirichlet-Randbedingungen

$$u(0, t) = u_0 \quad \text{und} \quad u(1, t) = u_1$$

sagen zum Beispiel, dass die beiden Enden des Stabes auf Temperatur u_0 und u_1 gehalten werden. Mit der Zeit wird sich eine stationäre Temperaturverteilung einstellen, also eine Temperaturverteilung, die sich mit der Zeit nicht mehr ändert. Eine solche hat zweite Ableitung

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{\kappa} \frac{\partial u}{\partial t} = 0.$$

Die Funktion $x \mapsto u(x, t)$ muss also linear sein, was auf die stationäre Temperaturverteilung

$$u_\infty(x) = xu_1 + (1 - x)u_0$$

führt.

Für konstante Neumann-Randbedingungen

$$\frac{\partial u}{\partial t}(0) = v_0 \quad \text{und} \quad \frac{\partial u}{\partial t}(1) = v_1$$

ist der Wärmefluss in den Stab konstant, es gilt

$$\frac{dU(t)}{dt} = (v_1 - v_0)t + U(0).$$

Insbesondere kann man nicht erwarten, dass es eine stationäre Lösung gibt. Vielmehr erwartet man eine Lösung, die linear mit der Zeit anwächst. Eine lineare Funktion von x kann nicht funktionieren, weil diese auf gleichen Wärmefluss durch die beiden Intervallenden führen würde. Wir versuchen daher den quadratischen Ausdruck.

$$u_s(x, t) = ax^2 + bx + c + d \cdot t.$$

Tatsächlich sind die Ableitungen

$$\frac{\partial u_s}{\partial t} = d$$

$$\frac{\partial^2 u_s}{\partial x^2} = 2a.$$

Eingesetzt in die Differentialgleichung finden wir den Wert von a

$$d = 2a\kappa \quad \Leftrightarrow \quad a = \frac{d}{2\kappa}.$$

Um die Werte von a und b zu bestimmen, müssen wir die Randbedingungen bemühen:

$$\begin{aligned} v_0 &= \frac{\partial u}{\partial x}(0) = b & \Rightarrow & & b &= v_0 \\ v_1 &= \frac{\partial u}{\partial x}(1) = 2a + b & \Rightarrow & & a &= \frac{v_1 - v_0}{2} \end{aligned}$$

Damit ist auch $d = \kappa(v_1 - v_0)$ bestimmt. Einzig c lässt sich auf diesem Weg nicht bestimmen, dazu sind weitere Dirichlet-Randbedingungen erforderlich.

Diskretisation

Zur Diskretisation verwenden wir ein Gitter mit Gitterkonstanten $h_x = 1/N$ und h_t . Die zu bestimmenden Unbekannten sind die u_{ik} mit $0 \leq i \leq N$ und $k \geq 0$. Die Randbedingungen auf dem Rand $t = 0$ geben die Werte

$$u_{i0} = u(ih_x, 0) = f(ih_x) =: f_i$$

vor.

Für Dirichlet-Randbedingungen an den Intervallenden legen die Werte von u_{ik} für $i = 0$ und $i = N$ fest. Die Variablen u_{0k} und u_{Nk} sind also nicht Unbekannte sondern vorgegebene Werte.

Die Neumann-Randbedingungen am linken und rechten Rand können mit Hilfe der Vorwärts- bzw. Rückwärts-Differenzen approximiert werden:

$$\frac{\partial u}{\partial n}(x_{0k}) \approx \frac{u_{1k} - u_{0k}}{h_x} = 0 \quad \text{und} \quad \frac{\partial u}{\partial n}(x_{Nk}) \approx \frac{u_{Nk} - u_{N-1,k}}{h_x} = 0$$

Dies bedeutet, dass die beiden Werte nahe des Randes gleich sind

$$u_{0k} = u_{1k} \quad \text{und} \quad u_{Nk} = u_{N-1,k}.$$

Die Differentialgleichung verwendet die zweite Ableitung nach x , für die wir die Approximation

$$\frac{\partial^2 u}{\partial x^2}(x_{ik}) = \frac{u_{i+1,k} - 2u_{ik} + u_{i-1,k}}{h_x^2}$$

verwenden können.

Matrixform

Da die Werte $u(x, 0) = f(x)$ von u zur Zeit $t = 0$ bereits bekannt sind, wird die numerische Lösung nacheinander die Variablen u_{ik} mit $k = 1, 2, 3, \dots$ bestimmen. Der Schritt $k \rightarrow k+1$ kann etwas kompakter beschrieben und vor allem leichter analysiert werden, wenn wir die Werte u_{ik} für gegebenes

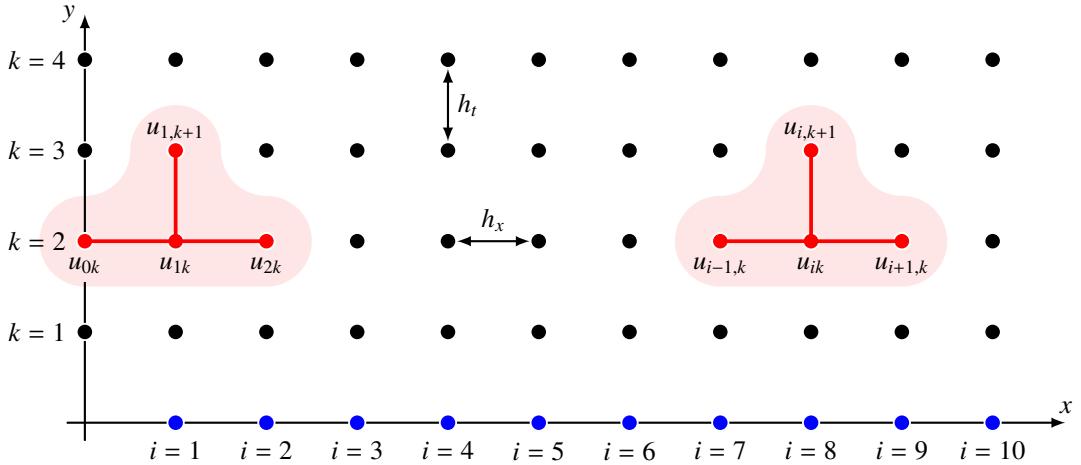


Abbildung 7.9: Euler-Verfahren für das Wärmeleitungsproblem.

k in den Vektor

$$u_k = \begin{pmatrix} u_{0k} \\ u_{1k} \\ \vdots \\ u_{ik} \\ \vdots \\ u_{Nk} \end{pmatrix}$$

zusammenfassen. Für homogene Randbedingungen ist der Zusammenhang zwischen u_k und u_{k+1} eine lineare Funktion, es muss also eine Matrix geben, welche $u_{k+1} = Au_k$.

Sind die Randbedingungen nicht homogen, kann man nicht mehr unbedingt eine zeitunabhängige Matrix A finden, vielmehr kann die Matrix in jedem Zeitschritt verschieden sein. Außerdem kann in jedem Zeitschritt ein konstanter Wert hinzukommen. Insgesamt gibt es also eine Matrix A_k und ein Vektor b_k derart, dass $u_{k+1} = A_k u_k + b_k$.

Für die erste Ableitung nach der Zeit könnten Vorwärts- oder Rückwärtsdifferenzen verwendet werden, in beiden Fällen entsteht ein unvermeidbarer Fehler und ein jeweils anderes numerisches Lösungsverfahren.

Euler-Verfahren

Wir müssen jetzt die Differentialgleichung des Wärmeleitungsproblems diskretisieren. Für die zweite Ableitung verwenden wir zweite Differenzen. Für die ersten Ableitungen haben wir verschiedene Optionen, wir verwenden Vorwärtsdifferenzen, also

$$\begin{aligned} \frac{\partial u}{\partial x}(x_{ik}) &\approx \frac{u_{i,k+1} - u_{ik}}{h_t} = \kappa \frac{\partial^2 u}{\partial x^2}(x_{ik}) \approx \kappa \frac{u_{i,k-1} - 2u_{ik} + u_{i,k+1}}{h_x^2} \\ u_{i,k+1} &= u_{ik} + \frac{h_t \kappa}{h_x^2} (u_{i-1,k} - 2u_{ik} + u_{i+1,k}) \end{aligned} \quad (7.12)$$

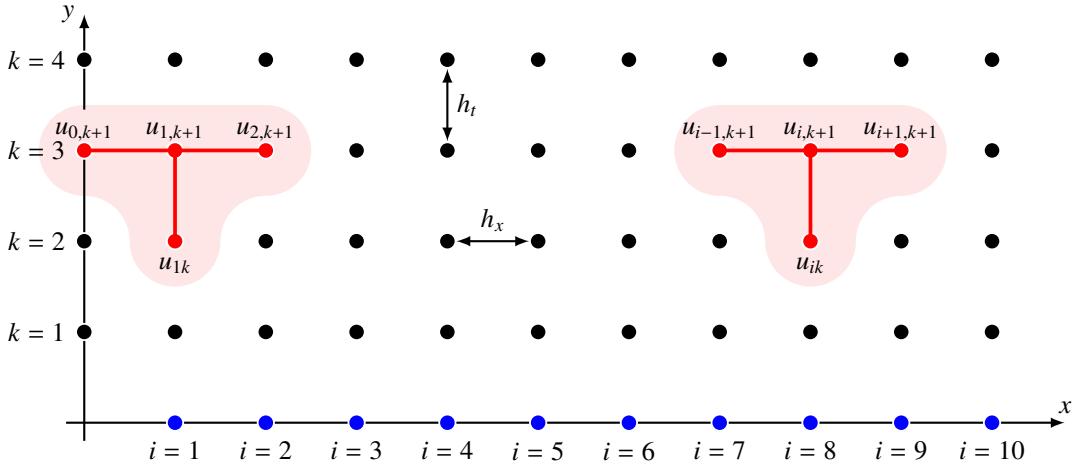


Abbildung 7.10: Verwendung von Rückwärts-Differenzen für die Wärmeleitungsgleichung

für die inneren Punkte. In Matrixform kann dies als

$$\begin{pmatrix} \vdots \\ u_{i,k+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 & \dots & \frac{h_t \kappa}{h_x^2} & 1 - \frac{2h_t \kappa}{h_x^2} & \frac{h_t \kappa}{h_x^2} & \dots & 0 \end{pmatrix} \begin{pmatrix} \vdots \\ u_{i-1,k} \\ u_{ik} \\ u_{i+1,k} \\ \vdots \end{pmatrix}$$

geschrieben werden. Wir kürzen den gemeinsamen Faktor $c = h_t \kappa / h_x^2$ ab und erhalten die Form

$$u_{k+1} = \begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ & c & 1-2c & c & & & \\ & & c & 1-2c & c & & \\ & & & c & 1-2c & c & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix} u_k.$$

Die Lösung wird sich durch Iteration dieser Matrix finden lassen, Konvergenz wird allein von c abhängen. Wir erwarten also, ein Konvergenzkriterium basierend auf c .

Die Gleichung (7.12) gilt nur für innere Punkte. Für die Variablen $u_{0k}, u_{1k}, u_{N-1,k}$ und u_{Nk} müssen wir aus den Randbedingungen zusätzliche Gleichungen für die Randwerte ableiten.

Rückwärts-Verfahren

Statt der Vorwärts-Differenz kann man auch die Rückwärts-Differenz für die erste Ableitung nach der Zeit verwenden. Die diskretisierte Gleichung wird dann

$$\frac{u_{ik} - u_{i,k-1}}{h_t} \approx \frac{\partial u}{\partial x}(x_{ik}) = \kappa \frac{\partial^2 u}{\partial x^2}(x_{ik}) \approx \frac{u_{i-1,k} - 2u_{ik} + u_{i+1,k}}{h_x^2}.$$

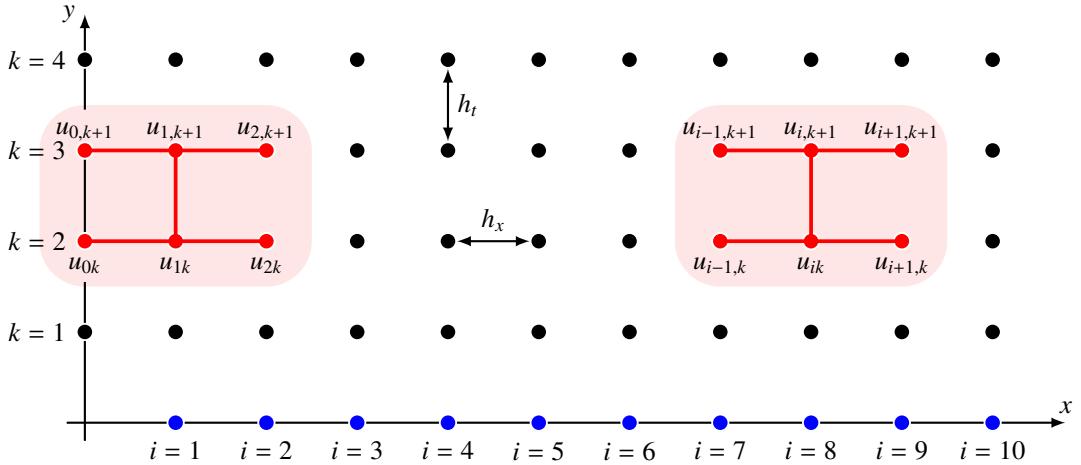


Abbildung 7.11: Verwendung der Knotenvariablen im Crank-Nicholson-Verfahren.

$$-cu_{i-1,k} + (1 + 2c)u_{ik} - cu_{i+1,k} = u_{i,k-1}.$$

In Matrixform kann man dies als

$$\begin{pmatrix} -c & 1+2c & -c \\ -c & 1+2c & -c \\ -c & 1+2c & -c \\ \ddots & \ddots & \ddots \end{pmatrix} u_k = u_{k-1}$$

schreiben. Wieder gelten diese Gleichungen nur für innere Punkte. Abbildung 7.10 zeigt, wie die Gleichungen die Knotenwerte untereinander verknüpfen.

Crank-Nicholson-Verfahren

Sowohl Vorwärts- wie auch Rückwärts-Differenzen bestimmen die erste Ableitung nach der Zeit eigentlich für einen Zeitpunkt zwischen den Vielfachen von h_t . Für diese Zeitpunkte stehen aber keine Funktionswerte zur Verfügung, um damit die Gleichung aufzustellen. Das Crank-Nicholson-Verfahren schlägt daher vor, den Mittelwert der zweiten Ableitungen zu benachbarten Zeitpunkten als Wert für den Zwischenzeitpunkt zu verwenden. Damit wird die Wärmeleitungsgleichung approximiert durch

$$\begin{aligned} \frac{u_{i,k+1} - u_{ik}}{h_t} &\approx \frac{\partial u}{\partial x}(x_{ik}) = \kappa \frac{\partial^2 u}{\partial x^2}(x_{ik}) \approx \frac{\kappa}{2} \left(\frac{u_{i-1,k+1} - 2u_{i,k+1} + u_{i+1,k+1}}{h_x^2} + \frac{u_{i-1,k} - 2u_{i,k} + u_{i+1,k}}{h_x^2} \right) \\ \Rightarrow \quad 2u_{i,k+1} - 2u_{ik} &= c(u_{i-1,k+1} - 2u_{i,k+1} + u_{i+1,k+1} + u_{i-1,k} - 2u_{i,k} + u_{i+1,k}) \end{aligned}$$

Verschiebt man die Terme nach Zeitpunkt auf die beiden Seiten der Gleichung erhält man

$$-cu_{i-1,k+1} + 2(1+c)u_{i,k+1} - cu_{i+1,k+1} = cu_{i-1,k} + 2(1-c)u_{i,k} + cu_{i+1,k}, \quad (7.13)$$

was sich leichter in Matrixform

$$\left(\begin{array}{cccccc} \ddots & \ddots & & & & \\ -c & 2(1+c) & -c & & & \\ & -c & 2(1+c) & -c & & \\ & & -c & 2(1+c) & -c & \\ & & & \ddots & \ddots & \\ \end{array} \right) u_{k+1} = \left(\begin{array}{cccccc} \ddots & \ddots & & & & \\ c & 2(1-c) & c & & & \\ & c & 2(1-c) & c & & \\ & & c & 2(1-c) & c & \\ & & & \ddots & \ddots & \\ \end{array} \right) u_k \quad (7.14)$$

bringen lässt.

7.2.3 Wärmeleitungsgleichung mit Dirichlet-Randbedingungen

Die bisher formulierten Gleichungen berücksichtigen die Randbedingungen nicht. Dirichlet-Randbedingungen am linken und rechten Rand des Intervalls legen die Werte u_{0k} und u_{Nk} fest. Wir möchten dies wieder in Matrixform schreiben. Da die Randbedingungen nicht von u abhängen, muss dies in der Form $u_{k+1} = Au_k + b_k$ möglich sein, was nur geht, wenn

$$u_{k+1} = \underbrace{\left(\begin{array}{cccccc} 0 & 0 & 0 & & & \\ c & 1-2c & c & & & \\ & c & 1-2c & c & & \\ & & \ddots & \ddots & \ddots & \\ & & & c & 1-2c & c & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & c & 1-2c & c & \\ & & & & & c & 1-2c & c & \\ & & & & & & 0 & 0 & 0 \end{array} \right)}_A u_k + \underbrace{\begin{pmatrix} g_{0k} \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}}_{b_k} \quad (7.15)$$

für die Randwerte g .

Für homogene Randbedingungen wird die Situation etwas einfacher zu analysieren, denn dann ist $b_k = 0$. Wir erwarten, dass die Lösung exponentiell gegen 0 konvergiert, das System "kühl aus". Da die Randbedingungen $u_{0k} = u_{Nk} = 0$ erzwingen, kann man diese Variablen weglassen, die Vektoren u und die Matrix A verkürzen sich auf

$$\tilde{u}_{k+1} = \underbrace{\begin{pmatrix} u_{1,k+1} \\ u_{2,k+1} \\ \vdots \\ u_{i,k+1} \\ \vdots \\ u_{N-1,k+1} \\ u_{N,k+1} \end{pmatrix}}_{\tilde{A}} = \underbrace{\left(\begin{array}{cccccc} 1-2c & c & & & & \\ c & 1-2c & c & & & \\ & \ddots & \ddots & \ddots & & \\ & & c & 1-2c & c & \\ & & & \ddots & \ddots & \ddots & \\ & & & & c & 1-2c & c & \\ & & & & & c & 1-2c & \\ & & & & & & 0 & 0 & 0 \end{array} \right)}_{\tilde{A}} \begin{pmatrix} u_{1k} \\ u_{2k} \\ \vdots \\ u_{ik} \\ \vdots \\ u_{N-1,k} \\ u_{Nk} \end{pmatrix} = A \tilde{u}_k. \quad (7.16)$$

Die Lösung u entsteht also durch Iteration mit der Matrix \tilde{A} . Der Spektralradius der Matrix gibt Auskunft darüber, ob das Verfahren konvergiert. Da die Matrix \tilde{A} symmetrisch ist, sind alle Eigenwerte

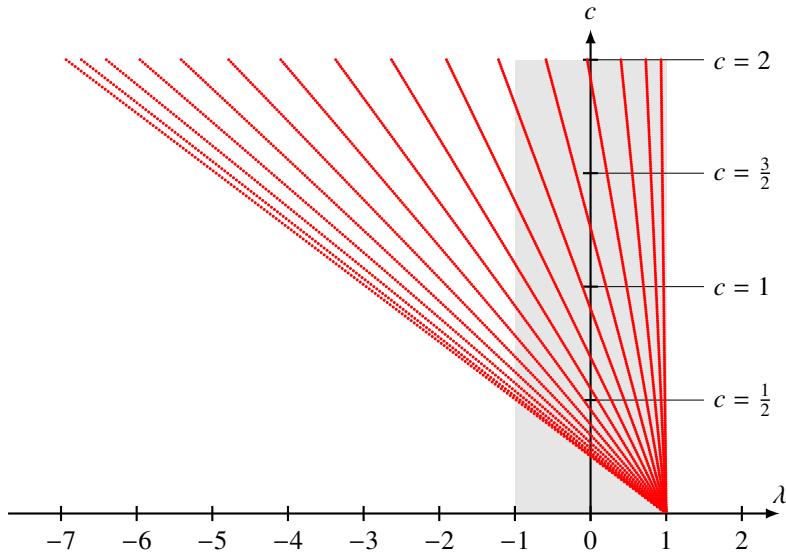


Abbildung 7.12: Spektrum der Matrix \tilde{A} für $N = 16$, die sich für das Euler-Verfahren mit homogenen Dirichlet-Randwerten ergibt. Nur für $c < 2$ ist der Spektralradius < 1 .

reell. In Abbildung 7.12 sind die Eigenwerte von \tilde{A} dargestellt. Man kann erkennen, dass nur für $c < \frac{1}{2}$ der Spektralradius < 1 wird. Daraus ergibt sich das Kriterium

$$\frac{2\kappa h_t}{h_x^2} < 1$$

für die Konvergenz des Verfahrens.

Für das Rückwärtsverfahren sieht die Matrix etwas anders aus, es gilt

$$\left(\begin{array}{cccccc} 0 & 0 & 0 & & & \\ -c & 1+2c & -c & & & \\ -c & 1+2c & -c & & & \\ \ddots & \ddots & \ddots & & & \\ -c & 1+2c & -c & & & \\ \ddots & \ddots & \ddots & & & \\ -c & 1+2c & -c & & & \\ -c & -c & 1+2c & -c & & \\ 0 & 0 & 0 & 0 & & \end{array} \right) u_{k+1} = Bu_{k+1} = u_k + b_k. \quad (7.17)$$

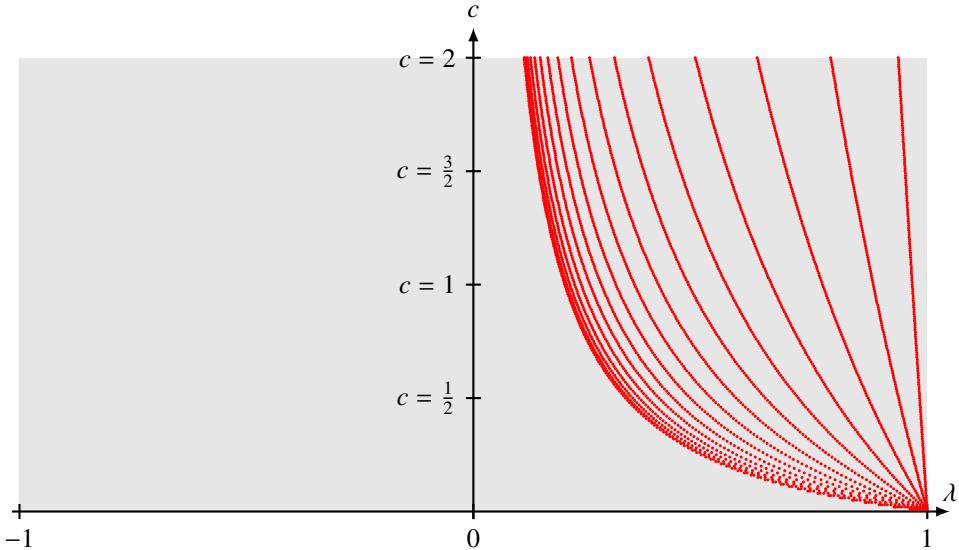


Abbildung 7.13: Eigenwertspektrum der Matrix \tilde{B}^{-1} für das Rückwärts-Verfahren. Der Spektralradius ist unabhängig von c immer < 1 .

Für den Fall homogener Randbedingungen werden auch hier wieder nur die “inneren” Koeffizienten

$$\tilde{B} = \begin{pmatrix} 1+2c & -c & & & \\ -c & 1+2c & -c & & \\ & \ddots & \ddots & \ddots & \\ & & -c & 1+2c & -c \\ & & & \ddots & \ddots & \ddots \\ & & & & -c & 1+2c & -c \\ & & & & & -c & 1+2c \end{pmatrix}$$

der Matrix B nötig. Die Lösung findet man dann aus $\tilde{B}\tilde{u}_{k+1} = \tilde{u}_k$ durch Iteration $\tilde{u}_{k+1} = \tilde{B}^{-1}\tilde{u}_k$. Konvergenz wird wieder bestimmt durch den Spektralradius $\varrho(\tilde{B})$, der, wie Abbildung 7.13 zeigt, immer < 1 ist. Das Rückwärtsverfahren ist also immer konvergent, ganz unabhängig von den relativen Schrittweiten.

Für das Crank-Nicholson-Verfahren ist, welches wir hier nur für homogene Randbedingungen betrachten wollen. Aus der Gleichung

$$\tilde{C}u_{k+1} = \begin{pmatrix} 2(1+c) & -c & & & \\ -c & 2(1+c) & -c & & \\ & -c & 2(1+c) & -c & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & \ddots \\ & & & & -c & 2(1+c) & -c \\ & & & & & -c & 2(1+c) \end{pmatrix} u_{k+1}$$

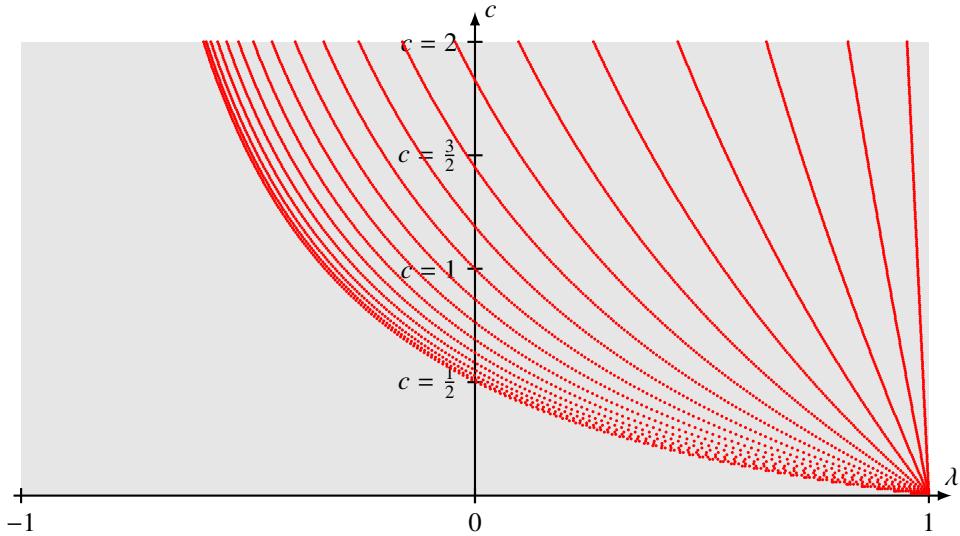


Abbildung 7.14: Eigenwertspektrum für das Crank-Nicholson-Verfahren

$$= \tilde{D}u_k = \begin{pmatrix} 2(1-c) & c & & & \\ c & 2(1-c) & c & & \\ & c & 2(1-c) & c & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots \\ & & & & c & 2(1-c) & c \\ & & & & & c & 2(1-c) \end{pmatrix} u_k$$

folgern wir, dass die Iteration $u_{k+1} = \tilde{C}^{-1}\tilde{D}u_k$ die Lösung der Gleichung liefert. Das Spektrum von $\tilde{C}^{-1}\tilde{D}$ ist in Abbildung 7.14 dargestellt und zeigt, dass auch das Crank-Nicholson-Verfahren immer konvergiert unabhängig von der Schrittweite h_t .

7.2.4 Wärmeleitungsgleichung mit Neumann-Randbedingungen

Die Neumann-Randbedingungen legen nicht Werte in den Randpunkten fest, sondern liefern nur Gleichungen zwischen den Variablen nahe dem Rand. Sie ändern damit die Koeffizientenmatrix des Gleichungssystems und haben nicht nur Einfluss auf konstante Vektoren b_k wie in Abschnitt 7.2.3.

Neumann-Randbedingungen

Wir betrachten in diesem Abschnitt nur homogene Neumann-Randbedingungen. Die Randbedingung am linken und rechten Rand verlangt, dass $u_{0k} = u_{1k}$ und $u_{N-1,k} = u_{Nk}$ gelten muss. Dies bedeutet, dass die Randwerte u_{0k} und u_{Nk} mit der gleichen Formel berechnet werden können wie die Werte für u_{1k} und $u_{N-1,k}$. Wir können damit die Variablen u_{0k} und u_{Nk} aus allen Gleichungen eliminieren, indem wir sie durch u_{1k} und $u_{N-1,k}$ ersetzen.

Euler-Verfahren

Aus den Gleichungen

$$\begin{aligned} u_{1,k+1} &= cu_{0k} + (1 - 2c)u_{1k} + cu_{2k} \\ u_{N-1,k+1} &= cu_{N-2,k} + (1 - 2c)u_{N-1,k} + cu_{Nk} \end{aligned}$$

am Rand des Gebietes werden nach den Ersetzungen $u_{0k} \rightarrow u_{1k}$ und $u_{Nk} \rightarrow u_{N-1,k}$ die Gleichungen

$$\begin{aligned} u_{1,k+1} &= (1 - c)u_{1k} + cu_{2k} \\ u_{N-1,k+1} &= cu_{N-2,k} + (1 - c)u_{N-1,k}. \end{aligned}$$

Die Matrix des Euler-Verfahrens wird damit zu

$$\tilde{u}_{k+1} = \underbrace{\begin{pmatrix} 1 - c & c & & & & \\ c & 1 - 2c & c & & & \\ & \ddots & \ddots & \ddots & & \\ & & c & 1 - 2c & c & \\ & & & \ddots & \ddots & \ddots \\ & & & & c & 1 - 2c & c \\ & & & & & c & 1 - c \end{pmatrix}}_A \tilde{u}_k. \quad (7.18)$$

Die Konvergenz des Verfahrens hängt wieder vom Spektralradius ab. Diesmal können wir aber nicht erwarten, dass alle Eigenwerte Betrag < 1 haben werden. Die Zeilen- und Spaltensumme der Matrix A ist immer 1, d. h. die ein konstanter Vektor wird auf sich selbst abgebildet. Insbesondere gibt es einen Eigenvektor zum Eigenwert 1, der Spektralradius kann also nicht kleiner als 1 sein. Für $c < 1$ konvergiert das Verfahren daher gegen die konstante Temperaturverteilung. Die numerische Durchführung des Verfahrens mit geeigneten Werten von h_x und h_t derart, dass $c < \frac{1}{2}$ ist, führt auf die Lösungsfläche in Abbildung 7.16.

Rückwärtsverfahren

Auch für das Rückwärtsverfahren können wir für Neumann-Randbedingungen die Matrixgleichung

$$Bu_k = \underbrace{\begin{pmatrix} 1 + c & -c & & & & \\ -c & 1 + 2c & -c & & & \\ & -c & 1 + 2c & -c & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & -c & 1 + 2c & -c \\ & & & & & -c & 1 + 2c & -c \\ & & & & & & -c & 1 + c \end{pmatrix}}_{B} u_k = u_{k-1} \quad (7.19)$$

finden. Auch diese Matrix hat den konstanten Vektor als einzigen Eigenvektor zum Eigenwert 1. Wie bei Dirichlet-Randbedingungen ist das Spektrum, dargestellt in Abbildung 7.17, bis auf den einen Eigenwert 1 im Intervall $(0, 1)$ enthalten, so dass das Verfahren konvergiert. Die Lösung ist dargestellt in Abbildung 7.18.

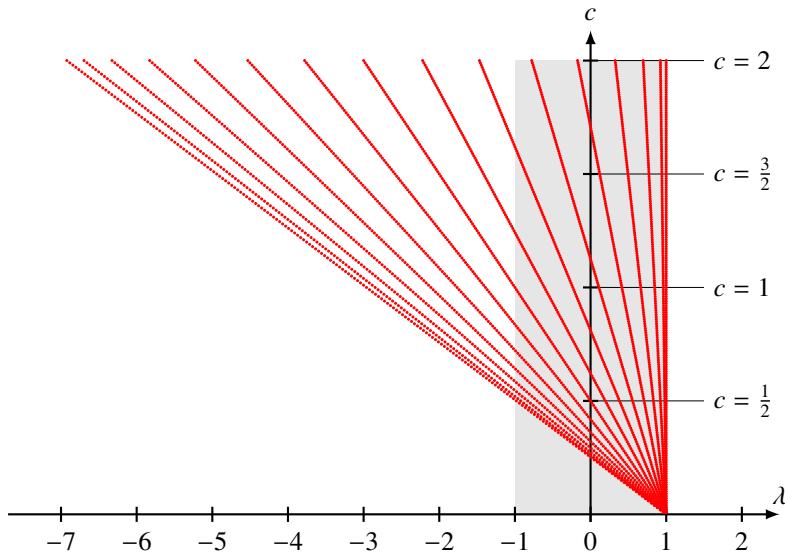


Abbildung 7.15: Eigenwertspektrum des Euler-Verfahrens für Neumann-Randbedingungen. Der Eigenwert 1 ist einfach, für $c < \frac{1}{2}$ ist der Spektralradius 1.

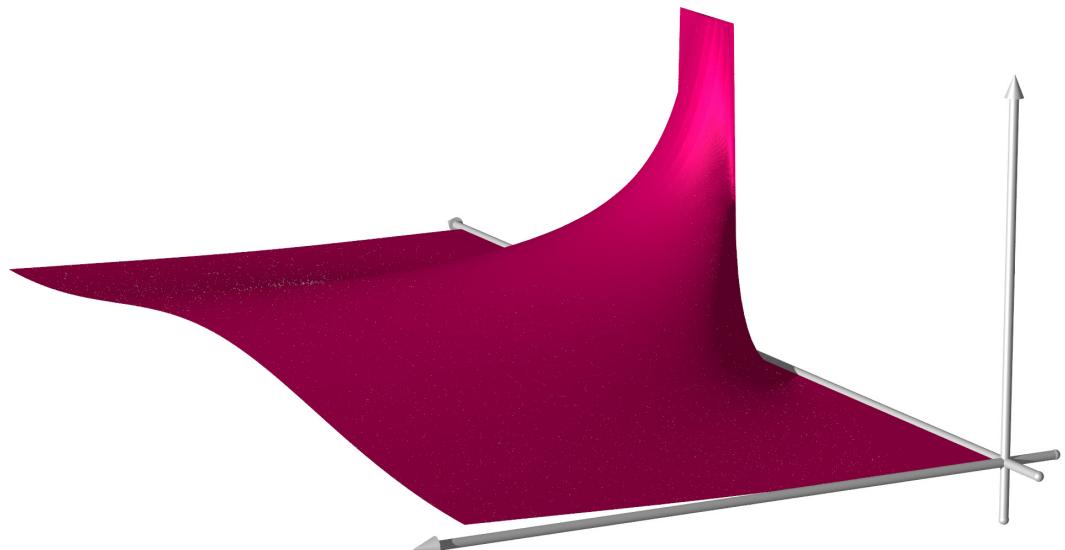


Abbildung 7.16: Lösung der Wärmeleitungsgleichung mit dem Euler-Verfahren.

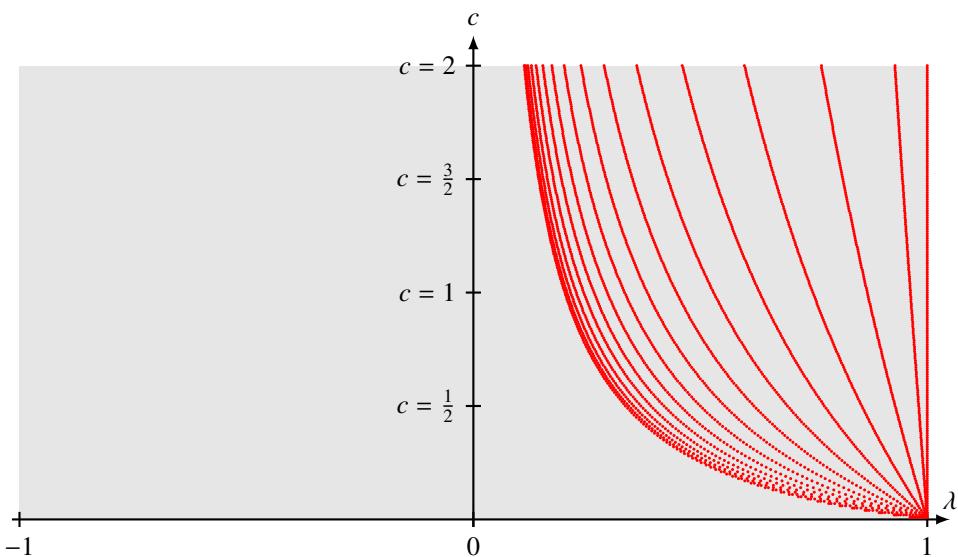


Abbildung 7.17: Das Eigenwertspektrum für das Rückwärtsverfahren für Neumann-Randbedingungen enthält eine einzelnen Eigenwert 1, alle anderen Eigenwerte haben Betrag < 1 .

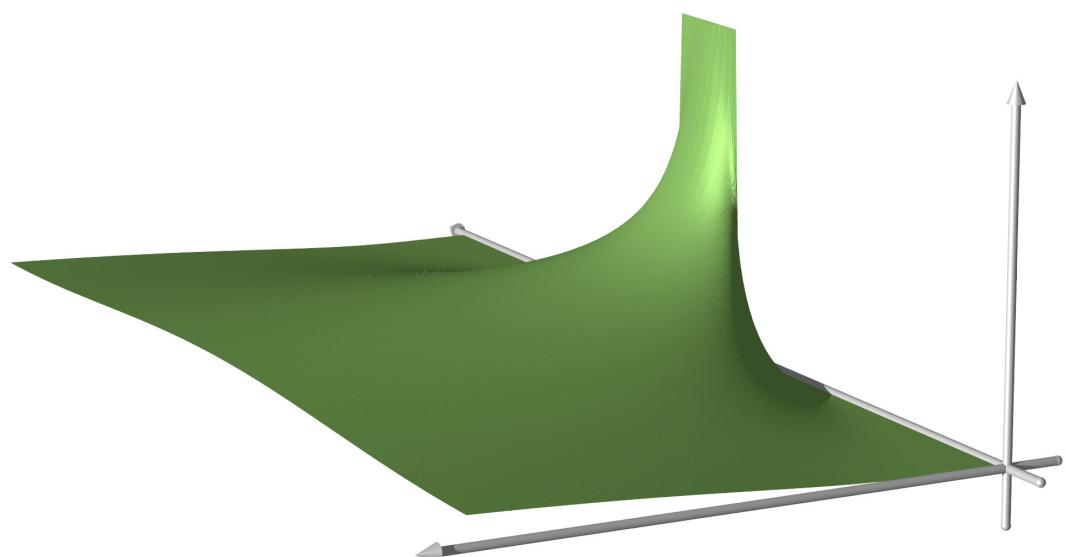


Abbildung 7.18: Lösung der Wärmeleitungsgleichung mit dem Rückwärtsverfahren.

Crank-Nicholson-Verfahren

Für das Crank-Nicholson-Verfahren sind die Gleichungen am Rand des Intervalls

$$\begin{aligned} -cu_{0,k+1} + 2(1+c)u_{1,k+1} - cu_{2,k+1} &= cu_{0k} + 2(1-c)u_{1k} + cu_{2,k} \\ (2+c)u_{1,k+1} - cu_{2,k+1} &= (2-c)u_{1k} + cu_{2,k}, \end{aligned}$$

so dass die Matrixgleichung

$$\begin{aligned} Cu_{k+1} &= \begin{pmatrix} (2+c) & -c & & & & \\ -c & 2(1+c) & -c & & & \\ & -c & 2(1+c) & -c & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & -c & 2(1+c) & -c \\ & & & & & -c & 2+c \end{pmatrix} u_{k+1} \\ &= \begin{pmatrix} 2-c & c & & & & \\ c & 2(1-c) & c & & & \\ & c & 2(1-c) & c & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & c & 2(1-c) & c \\ & & & & & c & 2-c \end{pmatrix} u_k = Du_k \end{aligned}$$

wird. Daraus erhält man als Iterationsverfahren:

$$u_{k+1} = C^{-1}Du_k.$$

Das Eigenwertspektrum (Abbildung 7.19) hat wieder einen einfachen Eigenwert 1, alle anderen Eigenwerte haben Betrag < 1, das Verfahren ist konvergent obwohl der Spektralradius exakt 1 ist.

In Abbildung 7.21 sind die drei Lösungen im gleichen Graphen dargestellt. Die Lösung des Euler-Verfahrens weicht stark von den anderen Verfahren ab. Die Wärmeleitungsgleichung hat Lösungen, bei denen sich Wärme mit unendlicher Geschwindigkeit ausbreitet. Im Euler-Verfahren kann sich ein Temperatursprung nur um h_x in jeder Iteration ausbreiten, also maximal mit der Geschwindigkeit h_x/h_t . Daher verhält sich das Euler-Verfahren so, wie wenn die Wärmeleitfähigkeit reduziert wäre, was zu dem ausgeprägteren ‘‘Buckel’’ der Euler-Lösung führt.

7.2.5 Stabilität und computational mode

In der Diskussion des Wärmeleitungsproblems haben wir für die Zeitableitung nicht versucht, symmetrische Differenzen zu verwenden. Tatsächlich gibt es einen wichtigen Grund dafür, der in diesem Abschnitt untersucht werden soll. Es wird sich zeigen, dass symmetrische Differenzen zusätzliche instabile Lösungen erzeugen können, die nichts mit realen Lösungen der Differentialgleichung zu tun haben. Dieser sogenannte *computational mode* führt auf nutzlose Resultate und dürfte mit ein Grund für die Misserfolge von Lewis Fry Richardson in seinen Pionierversuchen zur numerischen Wettervorhersage gewesen sein [3].

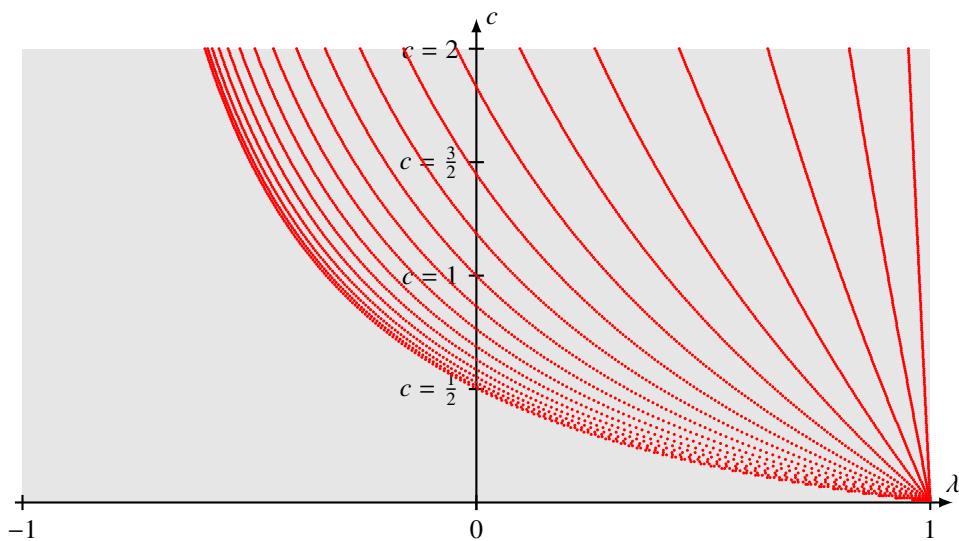


Abbildung 7.19: Das Eigenwertspektrum des Crank-Nicholson-Verfahrens zeigt einen einzelnen Eigenwert 1

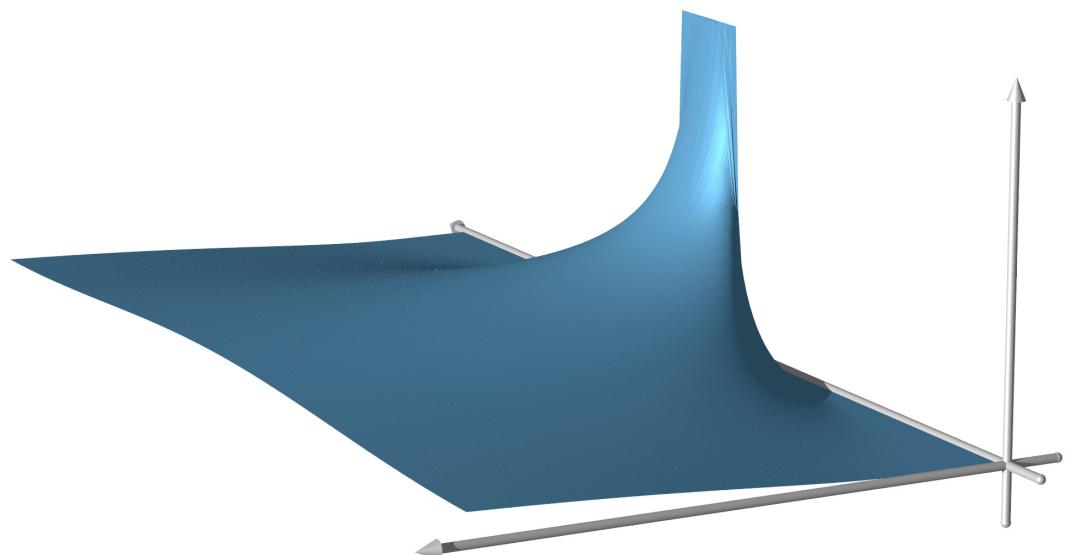


Abbildung 7.20: Lösung der Wärmeleitungsgleichung mit dem Crank-Nicholson-Verfahren

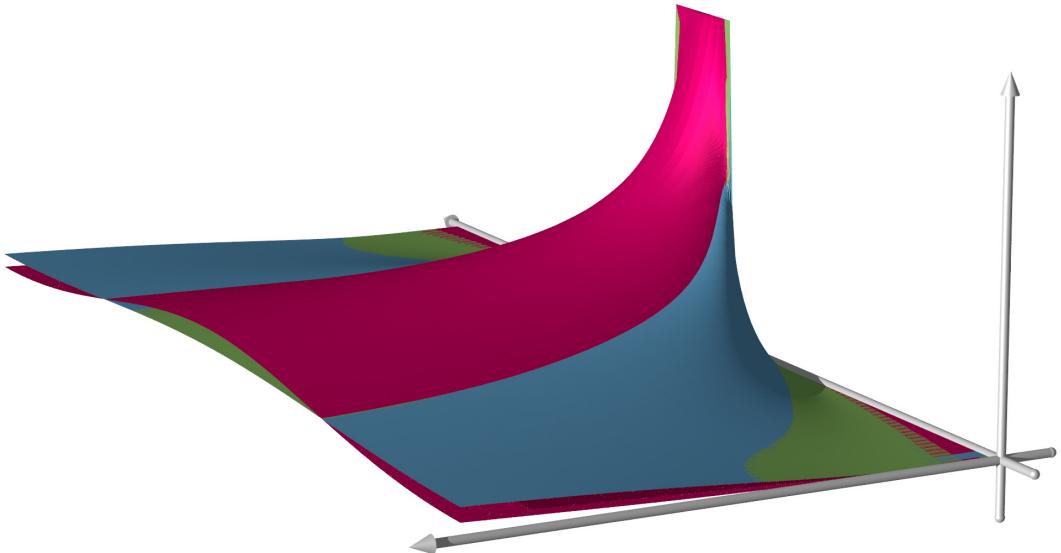


Abbildung 7.21: Alle drei Lösungsverfahren liefern ähnliche Lösungen, doch das Euler-Verfahren (rot) weicht stark von den beiden anderen Verfahren ab. Das Rückwärts-Verfahren und das Crank-Nicholson-Verfahren stimmen weitgehend überein.

Vorwärtssubstitution

Zur Illustration des Problems verwenden wir die einfache Differentialgleichung

$$y' = -y, \quad y(0) = y_0$$

welche die wohlbekannte Lösung $y(x) = y_0 e^{-x}$ hat. Zur Diskretisation verwenden wir die Gitterpunkte $x_i = ih$, $i \in \mathbb{Z}$ und versuchen die Werte $y_i = y(x_i)$ zu berechnen. Diskretisation mit Vorwärtssubstitution führt auf

$$y'(x_i) \approx \frac{y_{i+1} - y_i}{h} = -y(x_i) = -y_i \quad \Rightarrow \quad y_{i+1} = y_i - hy_i = (1 - h)y_i.$$

Damit kann man die Lösung als

$$y_i = y_0(1 - h)^i$$

schreiben. Wählt man $h = x/n$, dann liefert diese Approximation

$$y(x) = y_0 \left(1 - \frac{x}{n}\right)^n \rightarrow y_0 e^{-x}$$

für $n \rightarrow \infty$. Dieses Verfahren konvergiert also, wenn auch nicht besonders schnell, da das Euler-Verfahren nur ein Verfahren erster Ordnung ist.

Symmetrische Differenzen

Eine Schwierigkeit der Verwendung von Vorwärtssubstitution ist, dass die Vorwärtssubstitution eher eine Approximation für $y'(x_i + h/2)$ ist, nicht für $y'(x_i)$. Wie früher gezeigt wurde, können symmetrische Differenzen die Ableitung im Punkt x_i besser darstellen. Die zugehörige Differenzengleichung

wird jetzt

$$y'(x_i) = \frac{y_{i+1} - y_{i-1}}{2h} = -y(x_i) = -y_i \quad \Rightarrow \quad y_{i+1} + 2hy_i - y_{i-1} = 0.$$

Ein Potenzansatz $x_i = \lambda^i$ liefert die charakteristische Gleichung

$$\lambda^{i+1} + 2h\lambda^i - \lambda^{i-1} = \lambda^{i-1}(\lambda^2 + 2h\lambda - 1) = 0$$

mit den Lösungen

$$\lambda_{\pm} = -h \pm \sqrt{h^2 + 1}.$$

Die Quadratwurzel kann in die Taylor-Reihe

$$\sqrt{1 + h^2} \approx 1 + \frac{h^2}{2} + O(h^4)$$

entwickelt werden. Es folgt, dass λ -Wert für das positive Vorzeichen

$$\lambda_+ = -h + \sqrt{h^2 + 1} \approx 1 - h + \frac{h^2}{2} + \dots < 1$$

kleiner als 1 ist, insbesondere ist die Folge λ_+^i monoton fallend.

Verwenden wir wieder die Schrittweite $h = x/n$, dann ist

$$y_i = y_0 \lambda_+^i = y_0 \left(1 - h + \frac{h^2}{2}\right)^n = y_0 \left(1 - \frac{x}{n} + o(h)\right)^n \rightarrow y_0 e^{-x}$$

für $n \rightarrow \infty$, die Lösung für positives Vorzeichen liefert also genau die gleiche Lösung, die wir bereits mit Vorwärtendifferenzen gefunden haben.

Die zweite Lösung der charakteristischen Gleichung mit dem negativen Vorzeichen ist

$$\lambda_- = -\frac{h}{2} - \sqrt{\frac{h^2}{4} + 1} < -1.$$

Die Lösungsfolge λ_-^i hat alternierende Vorzeichen, aber die Beträge $|\lambda_-^i|$ wachsen exponentiell schnell an. Die Verwendung symmetrischer Differenzen führt also dazu, dass die Differenzengleichung eine zweite Lösung bekommt, die exponentiell schnell anwächst. Diese zweite Lösung heisst der *computational mode*.

Numerische Anregung des computational mode

Man muss sich fragen, warum man sich über die zweite Lösung überhaupt Gedanken machen soll, immerhin wird ja für die Lösung der Differentialgleichung nur die Lösung λ_+ benötigt. Der Grund ist, dass bei der Berechnung der Lösung mit Hilfe der Rekursionsformel

$$x_{i+1} = x_{i-1} - 2hx_i$$

die zweite Lösung trotzdem auftritt. Zu zwei aufeinanderfolgenden Werten y_0 und y_1 kann man mit der Rekursionsformel alle weiteren Werte berechnen. Aber selbst wenn man $y_1 = \lambda_+ + y_0$ setzt, wird die numerische Rechnung Rundungsfehler einführen, so dass der Wert y_1 , der für die Rekursion verwendet wird, nicht mit dem exakten Wert $\lambda_+ + y_0$ übereinstimmt. Wir nehmen an, dass dieser Rundungsfehler δ der einzige Fehler ist, der im Laufe der Berechnung eingeführt wird. Wir müssen also

die exakte Lösung für die Startwerte y_0 und $\lambda_+y_0 + \delta$ bestimmen. Es sind also Koeffizienten a_{\pm} zu finden mit

$$\begin{aligned} a_+ + a_- &= y_0 \\ a_+\lambda_+ + a_-\lambda_- &= y_0\lambda_+ + \delta \end{aligned}$$

Subtrahiert man das λ_+ -fache der ersten Gleichung von der zweiten Gleichung, erhält man

$$a_-(\lambda_-\lambda_+) = \delta \quad \Rightarrow \quad a_- = \frac{\delta}{\lambda_- - \lambda_+} = -\frac{\delta}{h}.$$

Es folgt, dass selbst ein einziger minimaler Fehler δ bei der Berechnung von y_1 die Lösung die zweite Lösung sichtbar macht.

Man kann auch ausrechnen, wie gross i werden muss, bis der computational mode von der gleichen Größenordnung wie die gesuchte Lösung geworden ist. Dieser Fall tritt ein wenn

$$|y_0\lambda_+^i| < |\delta\lambda_-^i| \quad \Rightarrow \quad \left| \frac{\lambda_-}{\lambda_+} \right|^i > \left| \frac{y_0}{\delta} \right| \quad \Rightarrow \quad i > \frac{\log |y_0/\delta|}{\log |\lambda_-/\lambda_+|}.$$

Für kleine Werte von h ist

$$\log |\lambda_{\pm}| = \log(1 \mp h + o(h)) \approx \mp h.$$

Die Bedingung an i kann man damit mit

$$i \gtrsim \frac{\log |y_0/\delta|}{2h}$$

approximieren. Der x -Wert, bei dem der computational mode überhand nimmt, ist daher

$$x \gtrsim h \frac{\log |y_0/\delta|}{2h} = \frac{1}{2}(\log |y_0| - \log |\delta|)$$

Für den `double`-Typ ist $|\delta| \approx 10^{-15}$, das Verfahren ist also grundsätzlich nicht in der Lage, vernünftige Resultate für $x > 7.5 \log 10 \approx 17.269$ zu liefern.

Die Annahme, der einzige Fehler trete bei der Rundung von y_1 auf, ist natürlich viel zu optimistisch. In Wahrheit tritt in jedem Schritt ein Fehler auf, so dass der computational mode schon viel früher angeregt wird. In Abbildung 7.22 ist der Betrag der Lösung für verschiedene Schrittweiten h gezeigt. Ganz zu Beginn, für $x < 1$, scheint die Lösung einigermassen genau dem theoretisch zu erwartenden Verlauf zu entsprechen. Dann beginnt jedoch der computational mode überhand zu nehmen, die Lösung wächst exponentiell schnell an und wird unbrauchbar.

Mehr Information zum computational mode insbesondere auch zu einer Filtertechnik, mit der der computational mode auch wieder gedämpft werden kann, ist im Kapitel 20 zu finden.

7.3 Finite Volumina

Die Diskretisierung mit Hilfe eines Gitters hat auf Variablen u_{ik} geführt, die Werte der Funktion u an den Gitterpunkten waren. Die Werte der Funktion zwischen diesen Punkten haben für die diskretisierten Gleichungen keine Rolle gespielt. Ausserdem hat sich gezeigt, dass Gleichungen erster Ordnung nur mit Kompromissen auf diese Weise diskretisiert werden.

Am Beispiel der zweidimensionalen Kontinuitätsgleichung soll in diesem Abschnitt illustriert werden, wie die gleiche Information, die in der Differentialgleichung steckt, auch in einer Integralform dargestellt werden kann. Dies führt auf eine alternative Diskretisation, in der wir nicht auf Differenzenquotienten angewiesen sind.

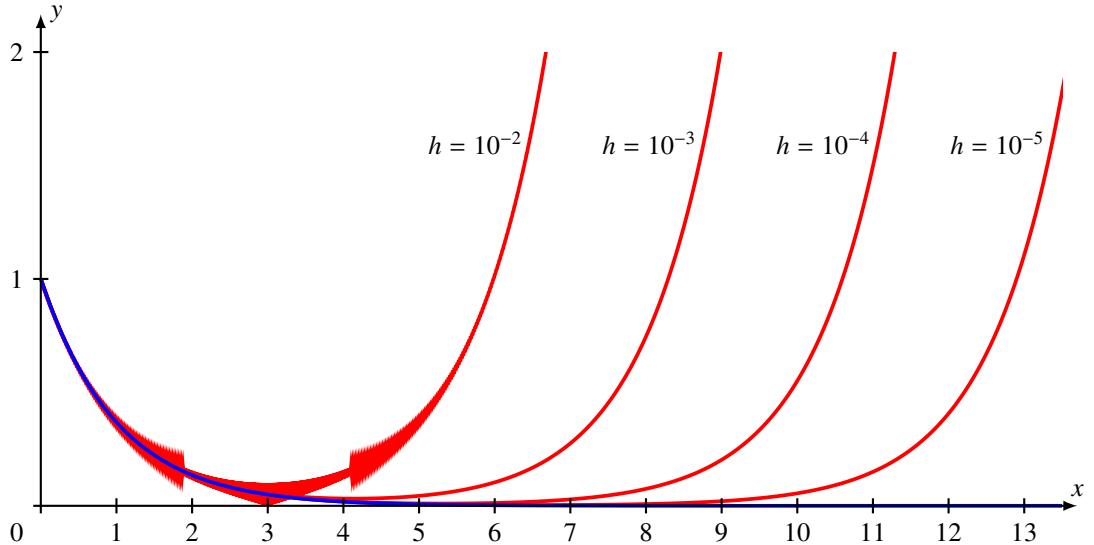


Abbildung 7.22: Betrag der Lösung der Differentialgleichung $y' = -y$ berechnet mit Hilfe symmetrischer Differenzen.

7.3.1 Die Kontinuitätsgleichung

Die Kontinuitätsgleichung beschreibt die Tatsache, dass in einem strömenden Fluid Materie nicht einfach aus dem Nichts entstehen kann oder verschwinden kann. Sie stellt eine Beziehung her zwischen der Dichte ϱ und der Strömungsgeschwindigkeit v . Der Einfachheit halber untersuchen wir das Problem nur in einer Dimension. Die Dichte $\varrho(x, t)$ wie auch die Geschwindigkeit $v(x, t)$ ist eine Funktion von Ort und Zeit.

Zur Herleitung der Kontinuitätsgleichung betrachten wir ein Intervall $[x, x + \Delta x]$. Zur Zeit t enthält es ungefähr die Masse $\varrho(x, t) \cdot \Delta x$. Die Masse kann sich während eines Zeitintervalls Δt dadurch ändern, dass Materie durch das linke und rechte Intervallende strömt. Die Menge, die durch das rechte Ende strömt ist $\varrho(x + \Delta x, t) \cdot v(x + \Delta x, t) \cdot \Delta t$, durch das linke Ende strömt $\varrho(x, t) \cdot v(x, t) \cdot \Delta t$. Die Masseänderung ist die Differenz, also

$$\varrho(x + \Delta x, t) \cdot v(x + \Delta x, t) \cdot \Delta t - \varrho(x, t) \cdot v(x, t) \cdot \Delta t = \varrho(x, t + \Delta t) \Delta x \varrho(x, t) \Delta x.$$

Nach Division durch $\Delta t \Delta x$ bleibt die Gleichung

$$\frac{\varrho(x + \Delta x, t)v(x + \Delta x, t) - \varrho(x, t)v(x, t)}{\Delta x} = \frac{\varrho(x, t + \Delta t) - \varrho(x, t)}{\Delta t}.$$

Im Grenzwert $\Delta t \rightarrow 0$ und $\Delta x \rightarrow 0$ entsteht die partielle Differentialgleichung

$$\frac{\partial(\varrho v)}{\partial x} = \frac{\partial \varrho}{\partial t} \quad \Leftrightarrow \quad \frac{\partial \varrho}{\partial t} - \frac{\partial j}{\partial x} = 0, \quad (7.20)$$

wobei wir in der letzten Umformung $j = \varrho v$ für den Massefluss geschrieben haben. Die Gleichung (7.20) heißt die *Kontinuitätsgleichung*.

Die Kontinuitätsgleichung kann ganz analog auch in höheren Dimensionen hergeleitet werden. Für den Stromvektor $\vec{j} = \varrho \vec{v}$ gilt

$$\frac{\partial \varrho}{\partial t} - \operatorname{div}(\varrho \vec{v}) = \frac{\partial \varrho}{\partial t} - \operatorname{div} \vec{j} = 0,$$

wobei die *Divergenz* eines Vektorfeldes \vec{a} durch

$$\operatorname{div} \vec{a} = \frac{\partial a_x}{\partial x} + \frac{\partial a_y}{\partial y} + \frac{\partial a_z}{\partial z}$$

gegeben ist.

7.3.2 Integralform

Die Herleitung der Kontinuitätsgleichung basiert auf dem Vergleich von Funktionswerten von ϱ und j in den Eckpunkten eines Rechtecks mit Kantenlänge Δx und Δt in der x - t -Ebene. Alternativ hätten wir aber auch Integrale entlang der Kanten verwenden können, um die Veränderung der Masse im Intervall $[x, x + \Delta x]$ zu berechnen.

Die Masse im Intervall $[x, x + \Delta x]$ ist gegeben durch das Integral

$$m(t) = \int_x^{x+\Delta x} \varrho(\xi, t) d\xi$$

der Dichte über das Interval. Die Änderung der Masse zwischen den Zeiten t und $t + \Delta t$ entsteht durch den Fluss durch die Endpunkte. Durch den linken und rechten Endpunkt des Intervalls fliessst im Zeitintervall $[t, t + \Delta t]$ die Masse

$$\int_t^{t+\Delta t} j(x, \tau) d\tau \quad \text{bzw.} \quad \int_t^{t+\Delta t} j(x + \Delta x, \tau) d\tau.$$

Die Änderung der Masse zwischen den Zeitpunkten t und $t + \Delta t$ ist die Differenz, also

$$\begin{aligned} m(t + \Delta t) - m(t) &= \int_t^{t+\Delta t} j(x + \Delta x, \tau) d\tau - \int_t^{t+\Delta t} j(x, \tau) d\tau \\ \Rightarrow \int_x^{x+\Delta x} \varrho(\xi, t + \Delta t) d\xi - \int_x^{x+\Delta x} \varrho(\xi, t) d\xi &= \int_t^{t+\Delta t} j(x + \Delta x, \tau) d\tau - \int_t^{t+\Delta t} j(x, \tau) d\tau \end{aligned} \quad (7.21)$$

Diese Gleichung gilt für beliebig grosse Schritte Δx und Δt , sie ist also nicht nur eine Approximation wie die Differenzenquotienten, die für die Herleitung der Kontinuitätsgleichung verwendet wurden. Die Differenzenquotienten ergaben erst im Grenzwert eine exakte Gleichung. Wir nennen (7.21) die *Integralform der Kontinuitätsgleichung*.

Die Herleitung der Integralform lässt sich auch in der mehrdimensionalen Situation durchführen. Dazu muss man einerseits die Masse $m(t)$ in einem Volumen V berechnen, was mit dem Dreifachintegral

$$m(t) = \iiint_V \varrho(x, y, z, t) dV$$

geschehen kann. Andererseits muss man den Materiefloss durch die Oberfläche von V berechnen können, was das Flussintegral

$$\oint_{\partial V} \vec{j}(x, y, z, t) d\vec{n}$$

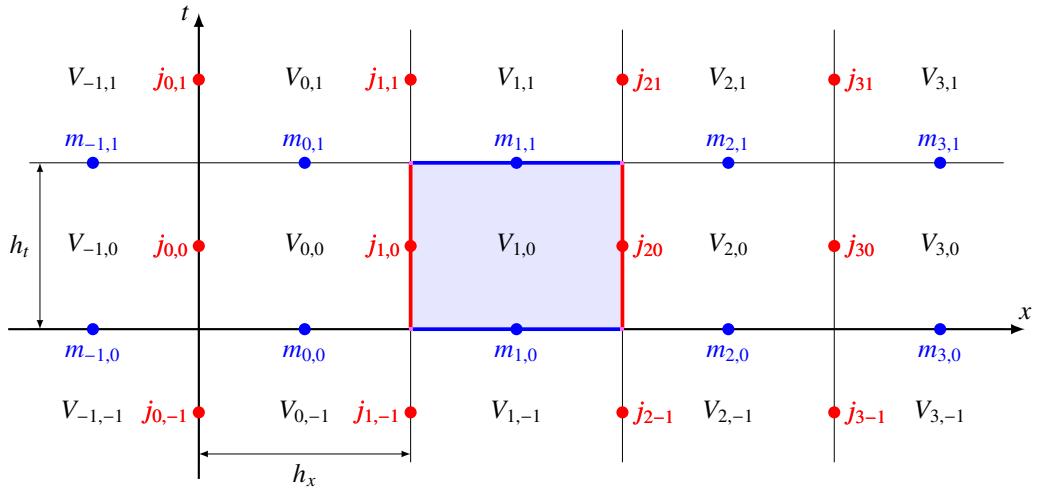


Abbildung 7.23: Diskretisation der Kontinuitätsgleichung (7.22) mit Hilfe diskreter Volumina. Die Variablen m_{ik} stehen für die im Intervall $[ih_x, (i+1)h_x]$ enthaltene Masse, j_{ik} steht für die im Zeitintervall $[kh_t, (k+1)h_t]$ durch das Intervallende ih_x fliessende Masse.

tut. Die Integralform der Kontinuitätsgleichung wird damit

$$\iiint_V \varrho(x, y, z, t + \Delta t) dV - \iiint_V \varrho(x, y, z, t) dV = \oint_{\partial V} \vec{j}(x, y, z, t + \Delta t) d\vec{n} - \oint_{\partial V} \vec{j}(x, y, z, t) d\vec{n}.$$

Die Integralsätze von Green, Gauss und Stokes ermöglichen ganz allgemein, viele der Differentialgleichungen, die Naturphänomene beschreiben, in eine Integralform zu bringen.

7.3.3 Diskretisation

Die Integralform (7.21) der Kontinuitätsgleichung suggeriert, dass die Integrale in dieser Gleichung bessere Variablen für eine Diskretisation sein könnten.

Eine Raumdimension

Wir verwenden wieder ein Gitter in der x - t -Ebene, aber als gesuchte Variablen verwenden wir die Integrale

$$m_{ik} = \int_{ih_x}^{(i+1)h_x} \varrho(\xi, kh_t) d\xi$$

$$j_{ik} = \int_{kh_t}^{(k+1)h_t} j(ih_x, \tau) d\tau$$

der Funktionen über Kanten im Gitter, wie in Abbildung 7.23 dargestellt. Die Kontinuitätsgleichung wird damit zu

$$m_{i,k+1} - m_{ik} = j_{i+1,k} - j_{ik}. \quad (7.22)$$

Man beachte auch hier wieder, dass dies keine Approximation ist, sondern dass diese Gleichungen exakt gelten.

Höhere Dimension

Um diese Idee auf höhere Dimensionen zu verallgemeinern zerlegt man das Gebiet in kleine Volumina V_i . Die Variablen

$$m_{ik} = \int_{V_i} \varrho(x, kh_t) dV$$

berechnen die Masse, die im Volumen V_i enthalten sind. Die Masseänderung in einem Zeitintervall setzt sich aus den Massenflüssen durch die verschiedenen Seitenflächen V_i zusammen. Seien σ_{il} die Seitenflächen des Volumens V_i . Daher verwenden wir

$$j_{ilk} = \int_{\sigma_{il}} j(\xi, kh_t) d\vec{n}.$$

Für jedes Volumen nimmt die Kontinuitätsgleichung die Form

$$m_{i,k+1} - m_{ik} = \sum_{l \text{ Seitenfläche von } V_i} (j_{il,k+1} - j_{il,k}). \quad (7.23)$$

an. Wieder sind lineare Gleichungen entstanden. Haben zwei Volumina V_{i_1} und V_{i_2} die Seite $\sigma_{i_1,j_1} = \sigma_{i_2,j_2}$ gemeinsam, dann sind die zugehörigen Flüsse entgegengesetzt:

$$j_{i_1 l_1 k} = j_{i_2 l_2 k} \quad \forall k \in \mathbb{Z},$$

denn was das Volumen V_{i_1} durch die Seite $\sigma_{i_1 l_1}$ an Masse verliert gewinnt das Volumen V_{i_2} durch die Seite $\sigma_{i_2 l_2}$.

Die Gleichungen (7.22) und (7.23) für sich alleine reichen nicht, weitere Gleichungen wie die Navier-Stokes-Gleichung werden zusätzlich benötigt, um ein Strömungsfeld vollständig zu beschreiben. Es ist allerdings nicht das Ziel dieses Abschnitts, die so begründete Methode der *finiten Volumina* in voller Allgemeinheit zu entwickeln.

7.4 Finite Elemente

Die Technik der finiten Volumina basierte darauf, dass die gesuchte Funktion durch Integrale über Volumina oder Flächenstücke ersetzt werden konnte, zwischen denen lineare Gleichungen gelten. Dies ist jedoch nicht die einzige denkbare Vorgehensweise. In diesem Abschnitt zeigen wir, wie gewisse partielle Differentialgleichungen in ein äquivalentes Minimalprinzip umgewandelt werden können. Approximiert man die gesuchte Funktion anschliessend durch geeignete Interpolationspolynome, wird das Minimalproblem zu einem quadratischen Minimalproblem für die Koeffizienten der Interpolationspolynome, welches mit Methoden der linearen Algebra gelöst werden kann.

7.4.1 Das äquivalente Minimalproblem

Zur Illustration des Prinzips soll in diesem Abschnitt das Eigenwertproblem für eine elliptische Differentialgleichung zweiter Ordnung betrachtet werden.

Ein eindimensionales Problem

Wir betrachten die Differentialgleichung

$$u''(x) = \lambda u(x) \quad (7.24)$$

auf dem Intervall $[a, b]$ mit Randwerten $u(a) = u(b) = 0$ und möchten zeigen, dass eine Lösung gleichzeitig ein stationärer Punkt des Integrals

$$I(u) = \int_a^b u'(x)^2 + \lambda u(x)^2 dx \quad (7.25)$$

ist.

Satz 7.9. Eine Funktion $u(x)$ ist genau dann eine Lösung der Differentialgleichung (7.24), wenn sie das Funktional $I(u)$ von (7.25) minimiert.

Beweis. Ein Minimum der Funktion $I(u)$ erfüllt die Bedingung, dass jede Variation $u(x) + \varepsilon h(x)$ von u , die den Randbedingungen genügt, einen grösseren Wert von I liefert. Die Ableitung nach ε verschwindet also an der Stelle $\varepsilon = 0$. Wir setzen

$$I(\varepsilon) = I(u(x) + \varepsilon h(x))$$

mit beliebigen Funktionen $h(x)$, die am Rand verschwinden: $h(a) = h(b) = 0$.

$$\begin{aligned} 0 &= \frac{dI(\varepsilon)}{d\varepsilon} \Big|_{\varepsilon=0} = \frac{d}{d\varepsilon} \int_a^b (u'(x) + \varepsilon h'(x))^2 + (u(x) + \varepsilon h(x))^2 dx \Big|_{\varepsilon=0} \\ &= \int_a^b 2u'(x)h'(x) + 2\varepsilon h'(x)^2 + 2\lambda u(x)h(x) + 2\lambda\varepsilon h(x)^2 dx \Big|_{\varepsilon=0} \\ &= 2 \int_a^b u'(x)h'(x) + \lambda u(x)h(x) dx \end{aligned}$$

Um das Integralprinzip von Lemma 3.18 anwenden zu können, darf nur $h(x)$ vorkommen. Wir können die Ableitung $h'(x)$ mit Hilfe von partieller Ableitung zum Verschwinden bringen.

$$= 2 \left[u'(x)h(x) \right]_a^b - 2 \int_a^b u''(x)h(x) dx + 2 \int_a^b \lambda u(x)h(x) dx.$$

Der erste Term verschwindet, da $h(x)$ an den Intervallenden verschwindet:

$$= 2 \int_a^b (-u''(x) + \lambda u(x)) h(x) dx. \quad (7.26)$$

Jetzt kann Lemma 3.18 angewendet werden: das Integral (7.26) kann nur dann für alle Funktionen $h(x)$ verschwinden, wenn

$$u''(x) = -\lambda u(x)$$

gilt. □

Partielle Differentialgleichung auf einem Rechteck

Das gleiche Prinzip ist auch anwendbar für das Eigenwertproblem

$$\Delta u(x) = \lambda u(x), \quad \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (7.27)$$

auf einem Rechteck $\Omega = (a, b) \times (c, d)$, es ist nur nicht ganz so klar, wie das Problem formuliert werden muss. Der zentrale Schnitt im Beweis von Satz 7.9 war partielle Integration und die Ausnutzung der Randwerte von $h(x)$. Indem wir dieses Beispiel auf ein Rechteck verallgemeinern, können wir die richtige Verallgemeinerung von Satz 7.9 finden.

Da die Funktion von zwei Variablen abhängt, gibt es jetzt nicht nur eine erste Ableitung sondern deren zwei. Statt dem Quadrat der ersten Ableitung $u'(x)^2$ werden daher einen analogen Terme für beide ersten Ableitungen benötigen. Die Quadratsumme der Ableitungen

$$(\nabla u)^2 = \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2$$

liegt auf der Hand. In Analogie zum eindimensionalen Problem verwenden wir daher als Minimalproblem das Funktional

$$I(u) = \int_a^b \int_c^d \nabla u(x)^2 + \lambda u(x)^2 dy dx$$

und schreiben wieder

$$I(\varepsilon) = I(u + \varepsilon h)$$

für eine Funktion $h: \Omega \rightarrow \mathbb{R}$, die auf dem Rand verschwindet: also

$$u(a, y) = u(b, y) = u(x, c) = u(x, d) = 0$$

für beliebige $x \in [a, b]$ und $y \in [c, d]$.

Die Ableitung nach ε an der Stelle $\varepsilon = 0$ ist

$$\begin{aligned} \frac{dI(\varepsilon)}{d\varepsilon} \Big|_{\varepsilon=0} &= \frac{d}{d\varepsilon} \int_a^b \int_c^d (\nabla u(x) + \varepsilon \nabla h(x))^2 + \lambda(u(x) + \varepsilon h(x))^2 dy dx \Big|_{\varepsilon=0} \\ &= 2 \int_a^b \int_c^d \nabla u(x) \cdot \nabla h(x) + \varepsilon \nabla h(x)^2 + \lambda u(x)h(x) + \lambda \varepsilon h(x)^2 dy dx \Big|_{\varepsilon=0} \\ &= 2 \int_a^b \int_c^d \nabla u(x) \cdot \nabla h(x) + \lambda u(x)h(x) dy dx. \end{aligned}$$

Das erste Term im Integranden enthält wieder Ableitungen der Funktion h , die wir loswerden müssen.

Lemma 7.10. Ist $v(x, y)$ eine beliebige Funktion $v: \Omega \rightarrow \mathbb{R}^2$ und $h: \Omega \rightarrow \mathbb{R}$. Dann gilt für das Integral von $v \cdot \nabla u$ die partielle Integrationsformel

$$\begin{aligned} \int_a^b \int_c^d v(x, y) \cdot \nabla h(x, y) dy dx &= \int_c^d v_x(b, y)h(b, y) - v_x(a, y)h(a, y) dy \\ &\quad + \int_a^b v_y(x, d)h(x, d) - v_y(x, c)h(x, c) dx \\ &\quad - \int_a^b \int_c^d \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} dy dx \end{aligned} \tag{7.28}$$

Beweis. Wir teilen das Integral mit Hilfe von

$$v(x, y) \cdot \nabla h(x, y) = v_x(x, y) \frac{\partial h}{\partial x}(x, y) + v_y(x, y) \frac{\partial h}{\partial y}(x, y)$$

in zwei Summanden

$$\int_a^b \int_c^d v \cdot \nabla h \, dy \, dx = \int_a^b \int_c^d v_x \frac{\partial h}{\partial x} \, dy \, dx + \int_a^b \int_c^d v_y \frac{\partial h}{\partial y} \, dy \, dx = I_1 + I_2$$

auf, die wir separat berechnen können. Für den ersten Summanden erhalten wir

$$\begin{aligned} I_1 &= \int_a^b \int_c^d v_x(x, y) \frac{\partial h}{\partial x}(x, y) \, dy \, dx \\ &= \int_c^d \int_a^b v_x(x, y) \frac{\partial h}{\partial x}(x, y) \, dx \, dy \\ &= \int_c^d \left[v_x(x, y) h(x, y) \right]_a^b - \int_a^b \frac{\partial v_x}{\partial x}(x, y) h(x, y) \, dx \, dy \\ &= \int_c^d v_x(b, y) h(b, y) - v_x(a, y) h(a, y) - \int_a^b \frac{\partial v_x}{\partial x}(x, y) h(x, y) \, dx \, dy \\ &= \int_c^d \color{red}{v_x(b, y) h(b, y)} - \color{blue}{v_x(a, y) h(a, y)} \, dy - \int_a^b \int_c^d \frac{\partial v_x}{\partial x}(x, y) h(x, y) \, dy \, dx. \end{aligned} \quad (7.29)$$

Die zweite Summe ist noch einfacher, weil es gar nicht erst notwendig ist, die Integrationsreihenfolge zu ändern:

$$\begin{aligned} I_2 &= \int_a^b \int_c^d v_y(x, y) \frac{\partial h}{\partial y} \, dy \, dx \\ &= \int_a^b \left[v_y(x, y) h(x, y) \right]_c^d - \int_c^d \frac{\partial v_y}{\partial y} h(x, y) \, dy \, dx \\ &= \int_a^b \color{green}{v_y(x, d) h(x, d)} - \color{orange}{v_y(x, c) h(x, c)} \, dx - \int_a^b \int_c^d \frac{\partial v_y}{\partial y} h(x, y) \, dy \, dx \end{aligned} \quad (7.30)$$

Die beiden Terme zusammen geben genau die im Lemma behauptete Formel. \square

Der erste Integral auf der rechten Seite von Lemma 7.10 kombiniert Integrale von $v_x(x, y)h(x, y)$ über die beiden vertikalen Kanten des Rechtecks, während das zweite Integral die Integrale von $v_y(x, y)h(x, y)$ über die horizontalen Kanten kombiniert (siehe auch Abbildung 7.24). Schreibt man $\vec{n}(x, y)$ für die nach aussen zeigende Normale auf den Rand des Rechtecks, dann können diese Integrale alle einheitlich geschrieben werden:

Kante	Integrand
untere Kante $y = c$	$-v_x(x, c)h(x, c) = h(x, c) \vec{v}(x, c) \cdot \vec{n}(x, c)$
obere Kante $y = d$	$v_x(x, d)h(x, d) = h(x, d) \vec{v}(x, d) \cdot \vec{n}(x, d)$
linke Kante $x = a$	$-v_y(a, y)h(a, y) = h(a, y) \vec{v}(a, y) \cdot \vec{n}(a, y)$
rechte Kante $x = b$	$v_y(b, y)h(b, y) = h(b, y) \vec{v}(b, y) \cdot \vec{n}(b, y)$

Es folgt also, dass die einfachen Integrale in Lemma 7.10 das Integral

$$\int_{\partial\Omega} \vec{v}(x, y)h(x, y) \cdot d\vec{n}$$

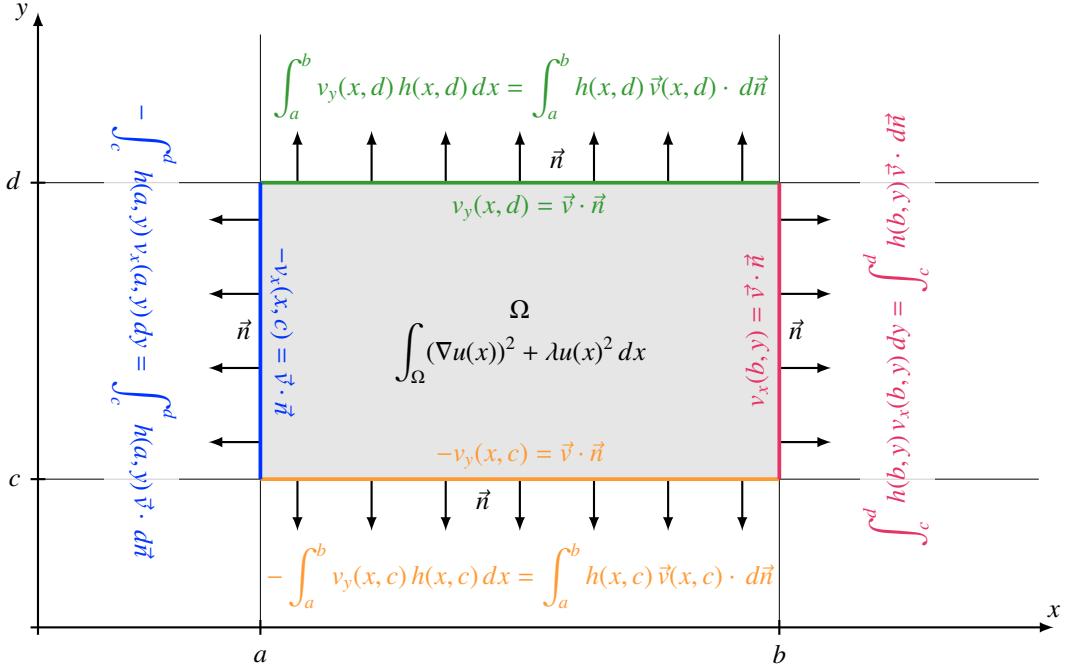


Abbildung 7.24: Die Randterme des Integrals (7.28) können als ein Flussintegral über den Rand des Rechtecks geschrieben werden. Dazu müssen die Integrale über die einzelnen Kanten des Rechtecks einzeln als Flussintegrale über die Kante geschrieben werden. Die Terme werden auch in den Gleichungen (7.29) und (7.30) mit den gleichen Farben hervorgehoben.

ist.

Nach diesen Vorarbeiten können wir jetzt das zur Differentialgleichung (7.27) gehörige äquivalente Minimalprinzip formulieren.

Satz 7.11. *Die Funktion $u: \Omega \rightarrow \mathbb{R}$ ist genau dann eine Lösung der Differentialgleichung*

$$\Delta u = \lambda u$$

wenn sie das Funktional

$$I(u) = \int_{\Omega} \nabla u(x, y)^2 + \lambda u(x, y)^2 dx dy$$

minimiert.

Beweis. Die Ableitung von $I(\varepsilon)$ an der Stelle $\varepsilon = 0$ wurde früher schon berechnet, das Integral (7.28) muss jetzt mit Hilfe der Formel für die partielle Integration von Lemma 7.10 umgeformt werden. Dazu setzen wir $\vec{v} = \nabla u$ und erhalten für

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \Delta u.$$

Es folgt

$$0 = \int_{\partial\Omega} \nabla u(x, y) h(x, y) \cdot d\vec{n} - \int_{\Omega} \Delta u(x, y) h(x, y) dx dy + \int_{\Omega} \lambda u(x, y) h(x, y) dx dy$$

$$= - \int_{\Omega} (\Delta u(x, y) - \lambda u(x, y)) h(x, y) dx dy.$$

Dies gilt genau dann für jede Funktion h , wenn

$$\Delta u = \lambda u,$$

wenn also u eine Lösung des Eigenwertproblems ist. \square

7.4.2 Approximation

Das äquivalente Minimalproblem zu einer partiellen Differentialgleichung hat das Problem etwas vereinfacht, die Ordnung der Ableitung ist reduziert worden, aber ist immer noch ein Problem, in dem eine Funktion bestimmt werden muss, also ein unendlichdimensionales Problem. In dieser Form ist es daher immer noch nicht einer effizienten numerischen Lösung zugänglich.

Beispielproblem

Zur Illustration soll in diesem Abschnitt das folgende Problem gelöst werden. Auf dem Intervall $\Omega = [0, \pi]$ ist eine Funktion $f(x)$ gegeben. Gesucht ist eine Funktion $u: [0, \pi] \rightarrow \mathbb{R}$ derart, dass

$$\begin{aligned} u''(x) &= f(x) \\ u(0) &= u_0 \quad u(\pi) = u_1. \end{aligned} \tag{7.31}$$

Als erstes müssen wir das äquivalente Minimalproblem finden.

Problem 7.12. Sei $F(x)$ eine Stammfunktion von $f(x)$. Eine Lösung des Differentialgleichungsproblems (7.31) minimiert

$$I(u) = \int_0^\pi (u'(x) - F(x))^2 dx.$$

Beweis. Die Richtungsableitung von $I(u)$ ist

$$\begin{aligned} \frac{dI}{d\varepsilon}(u + \varepsilon h) \Big|_{\varepsilon=0} &= \int_0^\pi 2(u'(x) + \varepsilon h'(x) - F(x))h'(x) dx \\ &= 2 \int_0^\pi (u'(x) - F(x))h'(x) dx \\ &= 2 \left[(u'(x) - F(x))h(x) \right]_0^\pi - 2 \int_0^\pi (u''(x) - F'(x))h(x) dx. \end{aligned}$$

Der erste Term verschwindet wegen $h(0) = h(\pi) = 0$ und es bleibt

$$= 2 \int_0^\pi (u''(x) - f(x))h(x) dx = 0.$$

Dies muss für alle $h(x)$ gelten, so dass folgt $u''(x) - f(x) = 0$ oder $u''(x) = f(x)$. \square

Die Stammfunktion $F(x)$ ist nicht eindeutig bestimmt, vielmehr ist jede $F(x) + C$ ebenfalls eine Stammfunktion. Nach obiger Rechnung führt sie jedoch auf die gleichen Minima.

Approximation mit Polynomen

Wir können aber davon ausgehen, dass die Lösungsfunktionen einigermassen glatt sind, also sich gut durch ein Polynom approximieren lässt. Wir approximieren jetzt $u(x)$ als Polynom und schreiben

$$u(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

Das Minimalprinzip für $u(x)$ führt auf ein Minimalprinzip für die Koeffizienten a_k . Zunächst brauchen wir die Ableitung von $u(x)$:

$$u'(x) = \sum_{k=1}^n k a_k x^{k-1}.$$

Jetzt müssen wir das Integral von

$$(u'(x) - F(x))^2 = u'(x)^2 - 2u'(x)F(x) + F(x)^2 \quad (7.32)$$

durch die Koeffizienten a_k ausdrücken. Für die drei Terme rechts in (7.32) müssen die Integrale

$$\begin{aligned} \int_0^\pi u'(x)^2 dx &= \int_0^\pi \sum_{i,j=1}^n i j a_i a_j x^{i+j-2} dx = \sum_{i,j=1}^n i j \int_0^\pi x^{i+j-2} dx a_i a_j = \sum_{i,j=1}^n i j \underbrace{\frac{\pi^{i+j-1}}{i+j-1}}_{b_{ij}} a_i a_j = a^t B a \\ \int_0^\pi u(x)' F(x) dx &= \int_0^\pi \sum_{i=1}^n i a_i x^{i-1} F(x) dx = \sum_{i=1}^n a_i \underbrace{\int_0^\pi i x^{i-1} F(x) dx}_{= b_i} = b^t a \\ \int_0^\pi F(x)^2 dx &=: c \end{aligned}$$

ausgewertet werden können. Gesucht ist jetzt also ein Koeffizientenvektor a , der den Ausdruck

$$Q(a) = a^t B a + b^t a + c$$

minimiert. Der Summand c kann natürlich weggelassen werden.

Die Randbedingungen müssen natürlich auch erfüllt sein, auch diese können wir durch die Polynomkoeffizienten ausdrücken:

$$\begin{aligned} u_0 &= u(0) = a_0 \\ u_1 &= u(\pi) = a_0 + a_1 \pi + a_2 \pi^2 + \dots + a_n \pi^n. \end{aligned}$$

Dies lässt sich auch in Matrixform mit der Matrix

$$A = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & \pi & \pi^2 & \dots & \pi^n \end{pmatrix}$$

als

$$Aa = \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}$$

schreiben.

Wir machen das Problem noch etwas konkreter und verlangen $f(x) = 1$ und $u_0 = u_1 = 0$. Dann ist $F(x) = x$ eine mögliche Stammfunktion und damit lässt sich der Vektor b berechnen als

$$b_i = \int_0^\pi iF(x)x^{i-1} dx = \int_0^\pi ix^i dx = \frac{i}{i+1}\pi^{i+1}.$$

Die Matrix B ist

$$B = \begin{pmatrix} \pi & \pi^2 & \pi^3 \\ \pi^2 & \frac{4}{3}\pi^3 & \frac{3}{2}\pi^4 \\ \pi^3 & \frac{3}{2}\pi^4 & \frac{9}{5}\pi^5 \end{pmatrix}.$$

Aus den Randbedingung folgt $a_0 = 0$ und

$$a_1\pi + a_2\pi^2 = 0 \quad \Rightarrow \quad a_1 = \pi a_2.$$

Die gesuchte Lösung muss sich also den Ausdruck

$$a_2 \begin{pmatrix} 1 \\ \pi \end{pmatrix}^t \begin{pmatrix} \frac{4}{3} & \frac{3}{2}\pi \\ \frac{3}{2}\pi & \frac{9}{5}\pi^2 \end{pmatrix} \begin{pmatrix} 1 \\ \pi \end{pmatrix} a_2 - \begin{pmatrix} \frac{2}{3}\pi^3 \\ \frac{3}{4}\pi^4 \end{pmatrix} \begin{pmatrix} 1 \\ \pi \end{pmatrix} a_2 = \left(\frac{4}{3}\pi^3 + 3\pi^5 + \frac{9}{5}\pi^7 \right) a_2^2 - \left(\frac{2}{3}\pi^3 + \frac{3}{4}\pi^5 \right) a_2^2.$$

Gesucht wird also nur das Minimum eines quadratischen Ausdrucks in a_2 und dieses wird bei $a_2 = \frac{1}{2}$ angenommen. Tatsächlich kann man nachprüfen, dass

$$u(x) = \frac{1}{2} \left(x - \frac{\pi}{2} \right)^2 - \frac{\pi^2}{8} = \frac{1}{2}x^2 - \frac{1}{2}x\pi + \frac{\pi^2}{8} - \frac{\pi^2}{8} = \frac{1}{2}x \left(x - \frac{\pi}{2} \right)$$

eine Lösung des Problems ist.

Allgemeine Formulierung

Etwas allgemeiner kann man das Problem wie folgt formulieren. Gesucht sei die Lösung der Differentialgleichung $u''(x) = f(x)$ auf dem Intervall $[a, b]$ mit Randbedingungen $u(a) = u_0$ und $u(b) = u_1$. Das zugehörige Minimalprinzip verlangt, dass

$$I(u) = \int_a^b (u'(x) - F(x))^2 dx, \quad \text{mit} \quad F(x) = \int_a^x f(\xi) d\xi$$

minimiert wird mit der Nebenbedingungen $u(a) = u_0$ und $u(b) = u_1$. Gesucht ist die Lösung $u(x)$ als Linearkombination einer linear unabhängigen Menge von Funktionen $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$. Zu bestimmen sind die Koeffizienten $a_k \in \mathbb{R}$ der Linearkombination

$$u(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \dots + a_n\varphi_n(x) = \sum_{k=0}^n a_k\varphi_k(x).$$

Das Minimalprinzip kann durch die Koeffizienten a_k als

$$\int_a^b (u'(x) - F(x))^2 dx = \int_a^b \sum_{i,j=0}^n a_i a_j \varphi'_i(x) \varphi'_j(x) - 2F(x) \sum_{i=0}^n \varphi'_i(x) + F(x)^2 dx$$

$$= \sum_{i,j=0}^n a_i a_j \underbrace{\int_a^b \varphi'_i(x) \varphi'_j(x) dx}_{= b_{ij}} + \sum_{i=0}^n a_k \underbrace{\int_a^b \varphi'_i(x) F(x) dx}_{= c_i} + \int_a^b F(x)^2 dx$$

ausgedrückt werden. Der letzte Summand hat keinen Einfluss auf das Minimum und kann daher weggelassen werden. Die Randbedingungen können ebenfalls vektoriell geschrieben werden:

$$\left. \begin{array}{l} u_0 = u(a) = \sum_{i=0}^n a_i \varphi_i(a) \\ u_1 = u(b) = \sum_{i=0}^n a_i \varphi_i(b) \end{array} \right\} \Rightarrow \underbrace{\begin{pmatrix} \varphi_0(a) & \varphi_1(a) & \dots & \varphi_n(a) \\ \varphi_0(b) & \varphi_1(b) & \dots & \varphi_n(b) \end{pmatrix}}_{= A} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = b$$

Das Problem ist damit auf das quadratische Minimalproblem

$$\begin{aligned} \text{minimiere} \quad & Q(a) = a^t B a + c^t a \\ \text{Nebenbedingung:} \quad & A a = b \end{aligned}$$

mit der Matrix

$$B = (b_{ij}) \quad \text{mit} \quad b_{ij} = \int_a^b \varphi'_i(x) \varphi'_j(x) dx$$

und dem Vektor

$$c = (c_i) \quad \text{mit} \quad c_i = \int_a^b \varphi'_i(x) F(x) dx$$

zurückgeführt.

Höhere Dimensionen

Das Beispielproblem der vorangegangenen Abschnitte war eindimensional, was erlaubt hat, bekannte Formeln wie die partielle Integration aus der Analysis zu verwenden. Ziel dieses Abschnitts ist, ein paar Eigenheiten der Verallgemeinerung auf ein mehrdimensionales Problem zu diskutieren.

Sei $\Omega \subset \mathbb{R}^2$ ein zweidimensionales Gebiet mit glattem Rand $\partial\Omega$, zum Beispiel ein Rechteck wie in Abbildung 7.24. Ausserdem ist $f: \Omega \rightarrow \mathbb{R}$ und $g: \partial\Omega \rightarrow \mathbb{R}$ geben. Das Differentialgleichungsproblem sucht eine Funktion $u: \bar{\Omega} \rightarrow \mathbb{R}$ mit

$$\begin{aligned} \Delta u &= f \quad \text{in } \Omega \\ u &= g \quad \text{auf } \partial\Omega. \end{aligned} \tag{7.33}$$

Die Rechnung zum eindimensionalen Problem suggeriert, dass das äquivalente Minimalproblem

$$I(u) = \int_{\Omega} (\nabla u(x, y) - F(x, y))^2 dx dy \tag{7.34}$$

ist, wobei $F(x, y)$ eine beliebige vektorwertige Funktion mit $\nabla \cdot F(x, y) = f(x, y)$ ist.

In den meisten Fällen ist es einfach, eine solche Funktion zu finden. Besonders einfach ist es, wenn die Funktion $f: \Omega \rightarrow \mathbb{R}$ stetig zu einer Funktion $\tilde{f}: \mathbb{R}^2 \rightarrow \mathbb{R}$ ausgedehnt werden kann. Dies ist nicht immer möglich, wie die Funktion φ des Polarwinkels des Punktes (x, y) auf dem Gebiet von Abbildung 7.25 zeigt. Gäbe es eine stetige Funktion $\tilde{\varphi}$, die φ erweitert, dann müssten die Grenzwerte

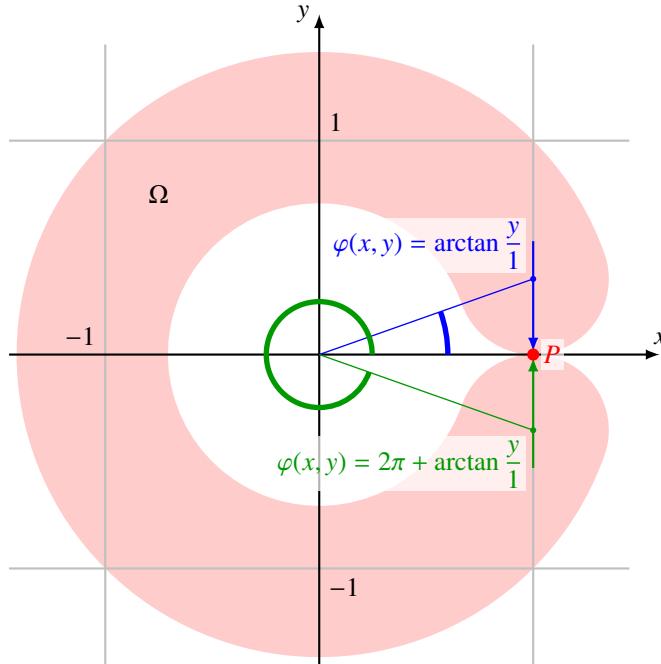


Abbildung 7.25: Die auf dem Gebiet Ω definierte Funktion, φ , die einem Punkt den Polarwinkel φ in Polarkoordinaten zuordnet, kann nicht stetig auf ganz \mathbb{R}^2 ausgedehnt werden. Der Grund dafür sind die unterschiedlichen Grenzwerte im Punkt P .

von $\bar{\varphi}$ im Punkt P von ‘‘oben’’ und von ‘‘unten’’ übereinstimmen. Der Grenzwert im Punkt P hängt aber von der Richtung ab, es ist

$$\begin{aligned}\lim_{y \rightarrow 0+} \varphi(1, y) &= \lim_{y \rightarrow 0+} \arctan \frac{y}{1} = 0 \\ \lim_{y \rightarrow 0-} \varphi(1, y) &= 2\pi + \lim_{y \rightarrow 0-} \arctan \frac{-y}{1} = 2\pi.\end{aligned}$$

Die Grenzwerte sind also verschieden.

Nehmen wir jetzt also an, dass es eine stetige Funktion $\tilde{f}: \mathbb{R}^2 \rightarrow \mathbb{R}$ gibt mit $\tilde{f}(x, y) = f(x, y)$ für $(x, y) \in \Omega$. Wir betrachten die Vektorfunktion $F(x, y)$ mit Komponenten

$$\left. \begin{array}{l} F_x(x, y) = \int_0^x \tilde{f}(\xi, y) d\xi \\ F_y(x, y) = 0 \end{array} \right\} \Rightarrow F(x, y) = \begin{pmatrix} F_x(x, y) \\ F_y(x, y) \end{pmatrix} = \begin{pmatrix} F_x(x, y) \\ 0 \end{pmatrix}. \quad (7.35)$$

Die Divergenz von $F(x, y)$ ist

$$\nabla \cdot F(x, y) = \frac{\partial F_x}{\partial x}(x, y) + \frac{\partial F_y}{\partial y}(x, y) = \frac{\partial}{\partial x} \int_0^x \tilde{f}(\xi, y) d\xi = f(x, y).$$

Die Funktion $F(x, y)$ definiert in (7.35) ist also eine Funktion der gesuchten Art. Die Erweiterbarkeit der Funktion $f(x, y)$ auf $\tilde{f}(x, y)$ stellt sicher, dass das Integral und damit $F(x, y)$ eine stetige Funktion von x und y wird, so dass das Minimalproblem wohldefiniert ist.

Für das vorgeschlagene Minimalprinzip können wir jetzt wieder die Richtungsableitung für eine Änderung von u um eine Funktion h mit verschwindenden Randwerten berechnen. Wie früher finden wir

$$\begin{aligned} \frac{dI(u + \varepsilon h)}{d\varepsilon} \Big|_{\varepsilon=0} &= \int_{\Omega} 2(\nabla u(x, y) - F(x, y)) \cdot \nabla h(x, y) dx dy \\ &= \int_{\partial\Omega} (\nabla u(x, y) - F(x, y)) h(x, y) \cdot d\vec{n} - \int_{\Omega} (\Delta u(x, y) - \nabla F(x, y)) h(x, y) dx dy. \end{aligned}$$

Darin verschwindet der erste Term, da $h(x, y) = 0$ ist auf dem Rand. Es bleibt

$$= - \int_{\Omega} (\Delta u(x, y) - f(x, y)) h(x, y) dx dy.$$

Da dieser Ausdruck für jede Funktion h mit $h(x, y) = 0$ mit $(x, y) \in \partial\Omega$ verschwinden muss, folgt wieder, dass

$$\Delta u = f$$

in Ω gelten muss. Das Minimalprinzip (7.34) gehört also tatsächlich zur Differentialgleichung (7.33).

7.4.3 Quadratische Minimalprobleme

In den vorangegangenen Beispielen wurde gezeigt, wie sich viele partielle Differentialgleichungen auf äquivalente Minimalprobleme zurückführen lassen. Indem man die Funktionen dann als Linearkombinationen approximiert, kann man daraus quadratische Minimalprobleme für die Koeffizienten dieser Linearkombinationen gewinnen. In diesem Abschnitt soll gezeigt werden, wie solche quadratischen Minimalprobleme algebraisch gelöst werden können.

Least-Squares-Problem

Das folgende Problem ist ein wohlbekanntes Beispiel für eine einfaches quadratisches Minimalproblem. Es soll hier als Einstieg dienen, Ziel ist eine allgemeinere Form von Minimalproblem zu formulieren und die Analysis hinzuzuziehen, um das verallgemeinerte Problem zu lösen. Natürlich springt dabei auch ein neuer Beweis für die bekannte Lösung des Least-Squares-Problems heraus.

Problem 7.13. Sei A eine $n \times m$ -Matrix mit $m < n$ und $\text{Rang } A = m$ und b ein n -dimensionaler Spaltenvektor. Finde einen m -dimensionalen Vektor x derart, dass $|Ax - b|$ minimal ist.

Die geometrische Intuition zur Lösung des Problems ist, dass Ax die Punkte eines Unterraums sind, der von den Spalten von A aufgespannt wird. Gesucht wird jetzt derjenige Punkt, der möglichst nahe am Punkt b liegt, der möglicherweise außerhalb des Unterraums ist. Für diesen Punkt Ax muss die Differenz $Ax - b$ auf dem Unterraum senkrecht stehen. Da dieser von den Spalten von A aufgespannt wird, muss $Ax - b$ auf allen Spalten von A senkrecht stehen. Das Skalarprodukt aller Spalten von A mit $Ax - b$ muss daher verschwinden, also

$$A^t(Ax - b) = 0 \quad \Rightarrow \quad A^t Ax = A^t b \quad \Rightarrow \quad x = (A^t A)^{-1} A^t b.$$

Diese Argumentation verwendet die geometrischen Eigenschaften des Skalarproduktes. Und funktioniert daher nur für den euklidischen Abstand, der von dem sehr speziellen quadratischen Ausdruck $x_1^2 + \dots + x_n^2$ berechnet wird. Im Folgenden soll das Problem für beliebige quadratische Ausdrücke verallgemeinert werden.

Problemstellung

Das Least Squares Problem 7.13 ist ein Spezialfall eines quadratischen Minimalproblems.

Problem 7.14. *Sei A eine Matrix und b ein Vektor wie in Problem 7.13. Finde einen Vektor x derart, dass die quadratische Funktion*

$$Q(x) = |Ax - b|^2 = (Ax - b)^t(Ax - b) = x^t A^t Ax - x^t A^t b - b^t A x + b^t b \quad (7.36)$$

minimiert wird.

Während es in der ursprünglichen Formulierung des Least-Squares-Problems um die Minimierung des Abstands ging, ist jetzt ein Problem entstanden, in der ein allgemeinerer quadratischer Ausdruck der Form $x^t B x$ minimiert werden muss, wobei $B = A^t A$ eine symmetrische Matrix ist. Allerdings ist immer noch die Beschreibung des Unterraums der zulässigen Punkte Ax vermischt mit dem quadratischen Ausdruck, der minimiert werden soll, in dem ebenfalls die Matrix A vorkommt. Um diese beiden Komponenten klarer zu trennen, formulieren wird das Problem wie folgt:

Problem 7.15. *Sei B eine positiv definite, symmetrische $n \times n$ -Matrix und A eine $m \times n$ -Matrix mit $m < n$ und $\text{Rang } A = m$. Sei weiter b ein m -dimensionaler Vektor. Finde einen n -dimensionalen Vektor x der $Ax = b$ erfüllt und $Q(x) = x^t b x$ minimiert.*

Nichtlineare Minimalprobleme

In der Theorie der Funktionen mehrere Variablen lernt man die folgende Methode zur Lösung nichtlinearer Minimalprobleme. Sie wird oft auch die Methode der Lagrange-Multiplikatoren genannt, die Zahlen λ_i heißen Lagrange-Multiplikatoren.

Lemma 7.16. *Gegeben ist eine Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}$ und m -Funktionen $g_i: \mathbb{R}^n \rightarrow \mathbb{R}$ mit $m < n$. Ist x ein Punkt, in dem $f(x)$ minimiert wird und gleichzeitig $g_i(x) = 0$ gilt, dann gibt es Zahlen λ_i derart dass*

$$\nabla f(x) = \lambda_1 \nabla g_1(x) + \cdots + \lambda_m g_m(x)$$

gilt.

Das Lemma 7.16 besagt, dass x und die Zahlen λ_i als Lösung der Gleichungen

$$\begin{aligned} g_i(x) &= 0 & i = 1, \dots, m \\ \nabla f(x) - \sum_{i=1}^m \nabla g_i(x) &= 0 \end{aligned} \quad (7.37)$$

gefunden werden kann. Die letzte Gleichung ist eine Vektorgleichung mit n -Komponenten, das System (7.37) ist also ein Gleichungssystem mit $m + n$ Gleichungen für die $n + m$ Unbekannten x und $\lambda = (\lambda_1, \dots, \lambda_m)$, ein Zeilenvektor.

Die Funktionen $g_i(x) = 0$ können in einen m -dimensionalen Vektor $g(x)$ zusammengefasst werden, für die wir auch den Gradienten

$$\nabla g(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_1} \\ \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_2} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial x_n} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}$$

definieren können. Mit diesen Notation wird das Gleichungssystem (7.37) schreiben als

$$\begin{aligned} g(x) &= 0 \\ \nabla f(x) - \lambda \nabla g(x) &= 0. \end{aligned} \tag{7.38}$$

In dieser Form versuchen wir das Problem zu lösen.

Gradienten von linearen und quadratischen Formen

Für die Lösung eines quadratischen Minimalproblems mit Hilfe der Gleichungen (7.38) muss der Gradient von quadratischen und linearen Funktionen berechnet werden können. In diesem Abschnitt tragen wir die benötigten Formeln zusammen.

Lemma 7.17. *Sei v ein n -dimensionaler Spaltenvektor und w ein n -dimensionaler Zeilenvektor. Der Gradient der Funktionen $f(x) = x^t v$ und $g(x) = wx$ ist*

$$\nabla f = v \quad \text{und} \quad \nabla g = w^t. \tag{7.39}$$

Beweis. Die Funktionen f und g sind etwas ausführlicher geschrieben

$$f(x) = x^t v = \sum_{i=1}^n x_i v_i \quad \text{und} \quad g(x) = wx = \sum_{i=1}^n w_i x_i.$$

Die partiellen Ableitungen von f und g sind

$$\begin{aligned} \frac{\partial f}{\partial x_k} &= \sum_{i=1}^n \frac{\partial x_i}{\partial x_k} v_i = \sum_{i=1}^n \frac{\partial x_i}{\partial x_k} v_i = \sum_{i=1}^n \delta_{ik} v_i = v_k \\ \frac{\partial f}{\partial x_k} &= \sum_{i=1}^n w_i \frac{\partial x_i}{\partial x_k} = \sum_{i=1}^n w_i \frac{\partial x_i}{\partial x_k} = \sum_{i=1}^n w_i \delta_{ik} = w_k \end{aligned}$$

also $\nabla f = v$ und $\nabla g = w$. □

Lemma 7.18. *Ist B eine $n \times n$ -Matrix, dann ist der Gradient der quadratischen Form $q(x) = x^t Bx$*

$$\nabla q(x) = (B + B^t)x. \tag{7.40}$$

Falls B symmetrisch ist, ist $\nabla q(x) = 2Bx$.

Beweis. Die Funktion $q(x)$ ist ausführlicher geschrieben

$$q(x) = x^t Bx = \sum_{i,j=1}^n x_i b_{ij} x_j.$$

Die partiellen Ableitungen sind

$$\begin{aligned} \frac{\partial q}{\partial x_k} &= \sum_{i,j=1}^n \frac{\partial}{\partial x_k} x_i b_{ij} x_j = \sum_{i,j=1}^n \frac{\partial x_i}{\partial x_k} b_{ij} x_j + \sum_{i,j=1}^n x_i b_{ij} \frac{\partial x_j}{\partial x_k} = \sum_{i,j=1}^n \delta_{ik} b_{ij} x_j + \sum_{i,j=1}^n x_i b_{ij} \delta_{jk} \\ &= \sum_{j=1}^n b_{kj} x_j + \sum_{i=1}^n x_i b_{ik}. \end{aligned}$$

Die beiden Terme sind die k -Komponente von Bx und die k -Komponente von $B^t x$, es folgt $\nabla q(x) = (B + B^t)x$. □

Invertierung von Blockmatrizen

Eine 2×2 -Matrix ist sehr leicht zu invertieren, es ist

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \Rightarrow \quad A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}. \quad (7.41)$$

Dies lässt sich zum Beispiel mit dem Gauss-Algorithmus sofort beweisen. Auf die gleiche Weise kann man aber auch eine Formel für die Inverse einer Blockmatrix herleiten.

Lemma 7.19. Gegeben ist die reguläre Matrix $(n+m) \times (n+m)$ -Matrix

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, \quad \text{und} \quad \left\{ \begin{array}{l} A \text{ eine } n \times n\text{-Matrix} \\ B \text{ eine } n \times m\text{-Matrix} \\ C \text{ eine } m \times n\text{-Matrix} \\ D \text{ eine } m \times m\text{-Matrix.} \end{array} \right.$$

Falls A regulär ist, ist auch $D - CA^{-1}B$ regulär und die Inverse von M ist

$$M^{-1} = \begin{pmatrix} A^{-1} - A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ (D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{pmatrix}. \quad (7.42)$$

Beweis. Wir schreiben E_n für die $n \times n$ -Einheitsmatrizen und führen den Gauss-Algorithmus auf einem Tableau von Blockmatrizen durch. Der Leser ist aufgefordert, sich zu überlegen, dass die Operationen des Gauss-Algorithmus auch funktionieren, wenn man sie in einer Algebra von Matrizen durchführt, man muss nur sorgfältig darauf achten, die Reihenfolge der Faktoren nicht zu verändern.

Der erste Schritt im Gauss-Algorithmus ist die Pivot-Division durch A , was in der Matrizenalgebra die Multiplikation von links mit A^{-1} wird. Nach Voraussetzung existiert A^{-1} , so dass die Zeilenoperation mit Pivot A durchgeführt werden kann:

$$\begin{array}{c|cc|cc} A & B & E_n & 0 \\ C & D & 0 & E_m \end{array} \xrightarrow{\quad} \begin{array}{c|cc|cc} E_n & A^{-1}B & A^{-1} & 0 \\ C & D & 0 & E_m \end{array} \\ \xrightarrow{\quad} \begin{array}{c|cc|cc} E_n & A^{-1}B & A^{-1} & 0 \\ 0 & D - CA^{-1}B & -CA^{-1} & E_m \end{array}.$$

Jetzt muss die Pivotdivision durch $D - CA^{-1}B$ durchgeführt werden, was aber nur möglich ist, wenn $D - CA^{-1}B$ regulär ist. Wäre $D - CA^{-1}B$ nicht regulär, dann könnte auch M nicht regulär sein. Somit kann auch die Pivotdivision durch $D - CA^{-1}B$ und das Rückwärtseinsetzen durchgeführt werden, was auf

$$\begin{array}{c|cc|cc} & E_n & A^{-1}B & A^{-1} & 0 \\ \xrightarrow{\quad} & 0 & E_m & -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{array} \\ \xrightarrow{\quad} \begin{array}{c|cc|cc} E_n & 0 & A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ 0 & E_m & -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{array}$$

führt. Daraus kann man die inverse Matrix von M ablesen und findet (7.42) \square

Beispiel. Im Fall $n = m = 1$ sind die Blöcke von M gewöhnliche reelle Zahlen und es kommt nicht mehr auf die Reihenfolge der Faktoren an, damit bekommt man für die Inverse der Matrix

$$\begin{aligned} A &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} & A^{-1} &= \begin{pmatrix} \frac{1}{a} + \frac{b}{a} \left(d - \frac{cb}{a} \right)^{-1} & \frac{b}{a} \left(d - \frac{cb}{a} \right)^{-1} \\ -\frac{c}{a} \left(d - \frac{cb}{a} \right)^{-1} & \left(d - \frac{bc}{a} \right)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{a} + \frac{bc}{ad-bc} & \frac{-b}{ad-bc} \\ \frac{-c}{ad-bc} & \frac{a}{ad-bc} \end{pmatrix} \\ &= \frac{1}{ad-bc} \begin{pmatrix} (ad-bc)-ad & -b \\ -c & a \end{pmatrix} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}, \end{aligned}$$

die Formel (7.41) für die Inverse einer 2×2 -Matrix. \circ

Korollar 7.20. Ist B eine symmetrische, positiv definite $n \times n$ -Matrix und und A eine $m \times n$ -Matrix mit $m \leq n$ und $\text{Rang } A = m$. Dann ist

$$\begin{pmatrix} 2B & -A^t \\ -A & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{2}B^{-1} & \frac{1}{2}B^{-1}A^t(AB^{-1}A^t)^{-1}AB^{-1} & -B^{-1}A(AB^{-1}A^t)^{-1} \\ -(AB^{-1}A^t)^{-1}AB^{-1} & \frac{1}{2}(AB^{-1}A^t)^{-1} \end{pmatrix} \quad (7.43)$$

Beweis. Dies ist der Fall $A = 2B$, $B = -A^t$, $C = -A$ und $D = 0$ des Lemmas 7.19. \square

Die Lösung eines quadratischen Minimalproblems

Das quadratische Minimalproblem 7.15 sucht einen Vektor x derart, dass $f(x) = x^t B x$ minimiert wird unter der Nebenbedingung $Ax = b$. Die Funktion $g(x)$ für dieses nichtlineare Extremalproblem ist $g(x) = Ax - b$. Nach dem Verfahren der Lagrange-Multiplikatoren 7.16 sind Vektoren x und λ zu finden derart, dass

$$\begin{aligned} g(x) &= 0 \\ \nabla f(x) - \lambda \nabla g(x) &= 0 \end{aligned}$$

gilt. Wir verwenden die Lemmata 7.17 und 7.18 zur Berechnung der Gradienten von $f(x)$ und $g(x)$:

$$\begin{aligned} \nabla f(x) &= (B + B^t)x \\ \nabla g(x) &= A^t. \end{aligned}$$

So entsteht das lineare Gleichungssystem

$$\begin{array}{l} 2Bx - A^t\lambda = 0 \\ Ax - b = 0 \end{array} \Rightarrow \begin{array}{l} Ax = b \end{array} \quad \left\{ \quad \text{oder} \quad \begin{pmatrix} 2B & -A^t \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \right.$$

in Matrixform.

Die Matrix hat die in Korollar 7.20 untersuchte Form. Mit der Formel (7.43) für die Inverse können wir jetzt die Lösung angeben:

$$x = B^{-1}A(AB^{-1}A^t)^{-1}b$$

$$\lambda = \frac{1}{2}(AB^{-1}A^t)^{-1}b.$$

Insbesondere ist damit die Lösung des quadratischen Minimalproblems vollständig auf Operationen mit Matrizenoperationen zurückgeführt.

Least Squares mit Hilfe des Gradienten

Zur weiteren Illustration der Rechentechnik mit Gradienten von Matrixfunktionen lösen wir hier auch noch das Least-Squares-Problem 7.14 mit dieser Methode. In diesem Fall haben wir keine Nebenbedingungen. Wir müssen nur das Minimum des Ausdrucks (7.36) bestimmen. Wir tun dies, indem wir Nullstellen der Ableitung suchen. Der Gradient von $Q(x)$ ist

$$\nabla Q(x) = 2A^tAx - 2A^tb = 0 \quad \Rightarrow \quad A^tAx = A^tb.$$

Daraus leitet man die bekannte Lösung

$$x = (A^tA)^{-1}A^tb$$

ab.

Literatur

- [1] Jean Dieudonné. *Foundations of Modern Analysis*. Dieudonné, Jean: Treatise on analysis Vol. 1. Academic Press, 1960.
- [2] *Kahan summation algorithm*. 29. Feb. 2020. URL: https://en.wikipedia.org/wiki/Kahan_summation_algorithm.
- [3] Peter Lynch. *The emergence of numerical weather prediction: Richardson's dream*. Cambridge University Press, 2006. ISBN: 978-0-52-185729-1.
- [4] Andreas Müller. *Source Code Repository*. 2020. URL: <https://github.com/AndreasFMueller/SeminarNumerik.git>.
- [5] David S. Watkins. *Fundamentals of Matrix Computations*. 3. Aufl. John Wiley und Sons, Inc., 2010.

Teil II

Anwendungen und weiterführende Themen

Übersicht

Im zweiten Teil kommen die Teilnehmer des Seminars selbst zu Wort. Die im ersten Teil dargelegten mathematischen Methoden und grundlegenden Modelle werden dabei verfeinert, verallgemeinert und auch numerisch überprüft.

Das Iterationsproblem wurde bereits in Kapitel 1 als zentral für das Verständnis der numerischen Berechnungsprozesse etabliert. *Michael Schneeberger* vertieft die Analyse in Kapitel 8 und zeigt das faszinierende Verhalten der Iterationsfolgen bei der Variation des Parameters in der logistischen Gleichung auf. Die Periodenverdoppelungen sind ein universelle Eigenschaft chaotischer Systeme.

Das Beispiel der Legendre-Polynome zeigt, dass selbst die Berechnung von Polynomen mit naheliegenden Formeln manchmal nicht einfach so funktioniert. *Patrick Elsener* analysiert in Kapitel 9 die Gründe dafür, warum diese Berechnung instabil werden kann und erklärt damit die Fehler populärer Websites wie Wolfram Alpha.

Integrale können natürlich auch mit Verfahren zur Lösung von gewöhnlichen Differentialgleichungen bestimmt werden. Mit besonders wenigen Auswertungen des Integranden kommt jedoch ein ganzlich anderes Verfahren aus, welches sich aus der Interpolationstheorie von Kapitel 3 entwickeln lässt. *Mike Schmid* stellt in Kapitel 10 die Gauss-Quadratur vor.

Ein mathematisches Modell für einen natürlichen Vorgang zu haben garantiert noch nicht, dass man diesen Vorgang exakt vorhersagen kann. *Manuel Cattaneo* und *Niccolò Galliani* untersuchen in Kapitel 11 die Van der Pol-Differentialgleichung, die einen nichtlinearen Oszillator beschreibt. Sie heben besonders hervor, wie für gewisse Parameterwerte chaotisches Verhalten auftritt. Dies bedeutet, dass die Rundungsfehler selbst so viel Unsicherheit in die Problemlösung injizieren, dass keine zuverlässigen Langzeitvorhersagen mehr möglich sind.

Das Beispiel der Van der Pol-Gleichung zeigt, dass es sich lohnt, etwas Aufwand in die Entwicklung guter numerischer Verfahren zu investieren. *Fabio Marti* zeigt in Kapitel 12, wie man auf der Basis der Taylor-Reihe im Prinzip ein Lösungsverfahren beliebig hoher Ordnung entwickeln kann. Die Wahl der Schrittweite hat einen entscheidenden Einfluss auf die Genauigkeit des Resultats. In Kapitel 13 zeigt *Reto Fritsche*, wie ein Lösungsalgorithmus die Schrittweite anpassen und damit die Genauigkeit optimieren kann. Eine beliebte analytische Technik zur Lösung gewöhnlicher Differentialgleichung verwendet die Laplace-Transformation, die jedoch oft schwierig zu invertieren ist. *Severin Weiss* erklärt in Kapitel 14, wie das Verfahren von Talbot die Laplace-Inverse numerisch berechnen kann.

Der Aufwand für die Lösung einer Differentialgleichung, die alle messbaren Einflüsse beinhaltet, ist manchmal nur mit grossem Aufwand durchführbar. Die Bahn eines Satelliten um die Erde wird von Sonne, Mond, den grossen Planeten und vielen weiteren Kräften beeinflusst. Eine CPU mit beschränkter Leistung kann diese Aufgabe nicht bewältigen. Die Störungstheorie ist in der Lage, alle diese Einflüsse in ein einfacheres Modell zusammenzufassen. *Daniel Bucher* und *Thomas Kistler* stellen diese interessante Technik an einem einfachen Beispiel in Kapitel 15 vor. Die Störungsidee

lässt sich auch auf andere Probleme anwenden. Von besonderer Bedeutung ist das Eigenwertproblem für lineare Operatoren, welches in der Quantenmechanik die Spektren von Atomen und Molekülen zu berechnen gestattet. Die vielfältigen Einflüsse sind jedoch in der Wellengleichung kaum alle zu berücksichtigen. Die von *Nicolas Tobler* in Kapitel 16 ausgeführte Störungstheorie für das Eigenwertproblem löst dieses Problem mit spektakulärem Erfolg, so dass wir heute auch die Feinheiten der Spektrallinien zum Beispiel des Wasserstoffatoms verstehen.

Polynome sind in manchen Fällen keine guten Approximationen, Brüche können in solchen Fällen gute Näherungsbrüche liefern. Kettenbrüche, von *Benjamin Bouhafs-Keller* in Kapitel 17 vorgestellt, liefern auf effiziente Weise Näherungsbrüche für irrationale Zahlen, die sogar eine Optimalitätseigenschaft besitzen. Die Padé-Approximation versucht Funktionen durch rationale Funktionen, d. h. Brüche von Polynomen, anzunähern. *Cédric Renda* zeigt in Kapitel 18, wie das funktioniert. Es zeigt sich, dass Kettenbrüche auf Padé-Approximationen führen. Kapitel 19 rechnet nach, dass die Näherungsbrüche der in Kapitel 17 vorgestellt Kettenbruchentwicklung der $\arctan x$ -Funktion Padé-Approximanten sind.

Die Gleichung von Burgers ist eine besonders einfache nichtlineare partielle Differentialgleichung, die viele numerische Probleme illustrieren kann. *Michael Schmid* zeigt einige mögliche Ansätze in Kapitel 20. In Abschnitt 7.4 wurden die wichtigsten Ideen der Methode der finiten Elemente an eindimensionalen Beispielen eingeführt. *Joël Rechsteiner* zeigt in Kapitel 21, was sich in zwei Dimensionen ändert.

Die lineare Algebra ist ein besonders wichtiges Betätigungsgebiet für Numeriker. Grundaufgaben der linearen Algebra wie Lösung eines Gleichungssystems, Matrixzerlegungen oder das Eigenwertproblem werden in grossen Simulationen immer wieder verwendet. Entsprechend sind effiziente Algorithmen wichtig. Das Kapitel 24 von *Raphael Unterer* erklärt ein besonderes erfolgreiches iteratives Lösungsverfahren für Gleichungssysteme mit symmetrischer, positiv definiter Koeffizientenmatrix. Der QR-Algorithmus löst zum Beispiel Least-Squares-Optimierungsprobleme, *Manuel Tischhauser* zeigt in Kapitel 22, wie man mit Givens-Drehungen eine effiziente Zerlegung finden kann. Für das Eigenwertproblem hat John Francis in den 1960er Jahren einen erfolgreichen Algorithmus formuliert, den *Tobias Grab* in Kapitel 23 vorstellt.

Zum Trainieren von neuronalen Netzwerken mit Gradientabstieg muss der Gradient der Zielfunktion berechnet werden können. *Martin Stypinsky* versucht in Kapitel 25, statt der üblichen Backpropagation die Zielfunktion als Black-Box zu behandeln und mit numerischen Ableitungsmethoden den Gradienten zu schätzen. Alternativ zu dem aus der Taylor-Reihe abgeleitete Differentiationsverfahren von Kapitel 25 kann man auch versuchen, ähnliche Verfahren aus der Interpolationstheorie abzuleiten, dies ist in Kapitel 26 durchgeführt.

8

Iteration der logistischen Gleichung

Michael Arthur Schneeberger

8.1 Modellierung von Populationen

Die logistische Gleichung ist ein beliebtes mathematisches Objekt, das zeigt, wie aus einer scheinbar einfach Gleichung ein sehr komplexes und chaotisches Verhalten entstehen kann. Ihren Ursprung hat die logistische Gleichung beim Modellieren vom zeitlichen Verlauf von Populationen. Wie man dabei ganz organisch auf die logistische Gleichung kommt, wird in diesem Abschnitt kurz demonstriert.

Als Erstes nehmen wir eine Population an, die ungehindert wachsen kann. Anfänglich hat diese die Tendenz, exponentiell zu wachsen. Sei x_n die Population zu einem bestimmten Zeitpunkt, beispielsweise am Anfang vom Jahr n . Nun ist x_{n+1} die Population im darauf folgenden Jahr, welche eine direkte Funktion der Population im vorherigen Jahr ist. Ein einfaches mathematisches Modell für dieses Verhalten könnte wie folgt aussehen:

$$x_{n+1} = \lambda x_n. \quad (8.1)$$

Der Faktor λ gibt in diesem Modell an, wie schnell die Population von Jahr zu Jahr ansteigt. Es ist schnell erkennbar, dass ein $\lambda > 1$ ein Wachstum und ein $\lambda < 1$ einen Zerfall der Population im folgenden Jahr zur Folge hat. Abbildung 8.1 zeigt die jährlichen Entwicklungen von Populationen für verschiedene Werte von λ mit diesem Modell. Auf der linken Grafik mit $\lambda = 0.75$ ist deutlich ein exponentieller Zerfall und auf der rechten Grafik mit $\lambda = 1.25$ ein exponentielles Wachstum ersichtlich.

In der Realität können Populationen natürlich nicht endlos exponentiell weiterwachsen, da früher oder später Platz, Nahrung, usw. ausgehen. Um dieses Verhalten in das Modell zu implementieren, fügen wir noch einen zusätzlichen Term hinzu, womit wir auch schon bei der logistischen Gleichung

$$x_{n+1} = \lambda x_n(1 - x_n) \quad (8.2)$$

angekommen sind. Dieser neue Term sorgt dafür, dass das Wachstum der Population zurückgeht, wenn sie grösser wird. Der Wert von x_n ist so limitiert auf $0 \leq x_n \leq 1$ und kann als die relative

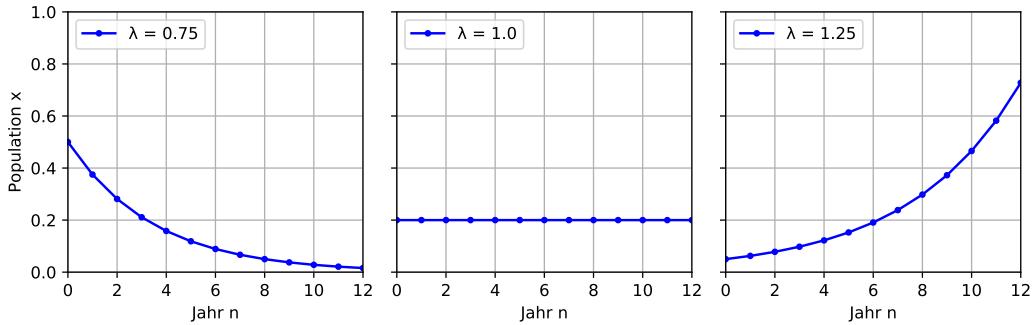


Abbildung 8.1: Exponentielle Populationsverläufe nach Gleichung (8.1) und verschiedenen Werten von λ .

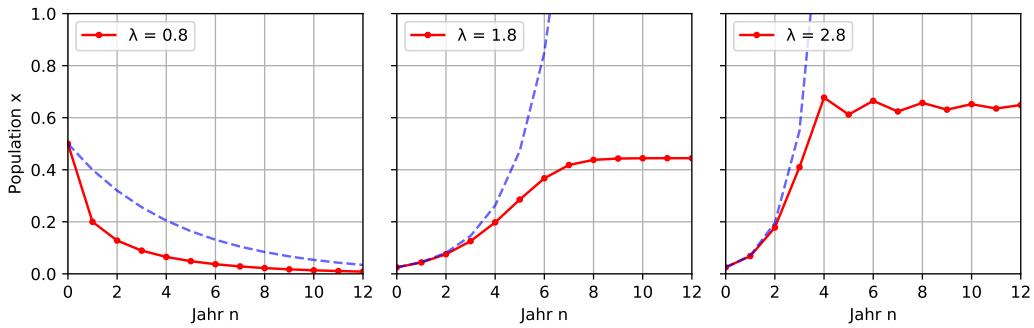


Abbildung 8.2: In rot: logistische Populationsverläufe nach Gleichung (8.2) und verschiedenen Werten von λ . In blau, zum Vergleich: exponentielle Populationsverläufe nach Gleichung (8.1) mit gleichem λ .

Population mit einem theoretischen Maximum von 1 interpretiert werden. Abbildung 8.2 zeigt nun für einige Werte von λ die jährlichen Entwicklungen von Populationen, welche mit der logistischen Gleichung modelliert werden. Dazu ist zum Vergleich das bereits bekannte exponentielle Modell mit gleichem λ als gestrichelte Linie abgebildet. Auf der linken Grafik ist auch hier wieder erkennbar, dass ein $\lambda < 1$ zur Folge hat, dass die Population rasch ausstirbt. Auf der mittleren Grafik mit $\lambda = 1.8$ sieht die Kurve zuerst annähernd exponentiell wachsend aus, doch der neu hinzugefügte Term sorgt jetzt dafür, dass das Wachstum bei grösser werdender Population zurückgeht und sich schliesslich auf ≈ 0.42 einpendelt. Sehr interessant ist die rechten Grafik mit $\lambda = 2.8$, wo die Population zuerst ein wenig „überschwingt“, sich aber schliesslich auf ≈ 0.62 einpendelt.

8.2 Iteration der logistischen Gleichung

Im Abschnitt 1.3 haben wir uns bereits mit dem Iterieren von Funktionen befasst. Genau dieses Prinzip der Iteration finden wir auch bei der logistischen Gleichung wieder. Jede Berechnung der Population im Folgejahr ist eine Iteration der logistische Gleichung. Die logistische Gleichung (8.2)

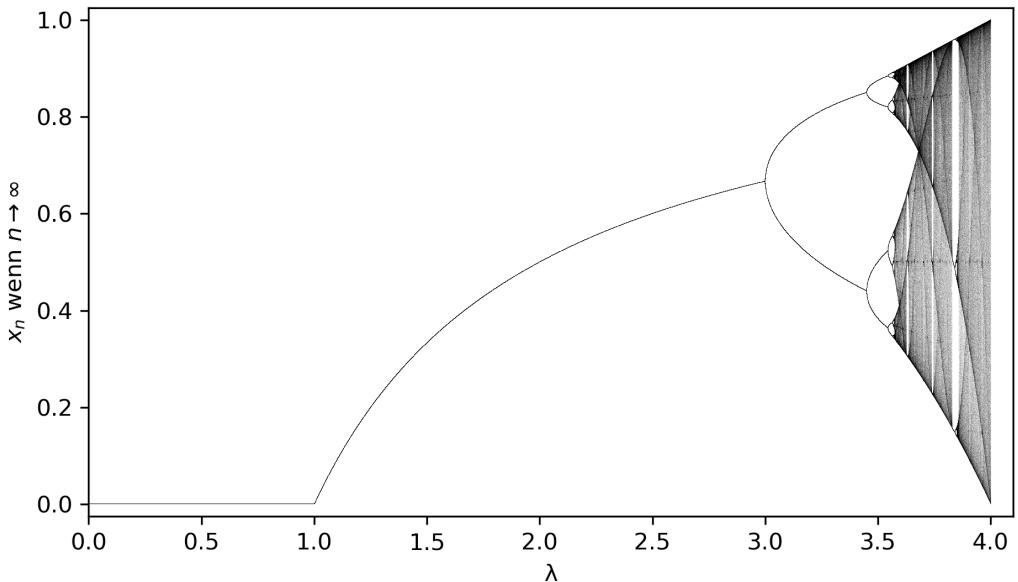


Abbildung 8.3: Bifurkationsdiagramm der logistischen Gleichung (8.2). Das Diagramm zeigt für jeden Wert von λ welche Werte x_n annimmt wenn $n \rightarrow \infty$.

könnte man auch als Funktion $f(x) = \lambda x(1-x)$ schreiben. Jede Iteration ist nun eine Verschachtelung von $f(x)$, somit lässt sich jeder beliebige Wert x_n auch wie folgt berechnen:

- $x_1 = f(x_0)$
- $x_2 = f(x_1) = f(f(x_0))$
- $x_3 = f(x_2) = f(f(f(x_0)))$
- $x_4 = f(x_3) = f(f(f(f(x_0))))$
- $x_5 = \dots$

Wir haben im Abschnitt 8.1 bereits beobachtet, dass die logistische Gleichung beim Iterieren für verschiedene Werte von λ gegen verschiedene Endwerte konvergiert. Der Anfangswert x_0 scheint dabei keine Rolle zu spielen solange er sich im Bereich $0 < x < 1$ befindet, er beeinflusst lediglich die Form der Kurve, bevor sie den Konvergenzwert erreicht. Das heisst, wir können x_0 einfach auf einen bestimmten Wert, zum Beispiel 0.1, setzen. Nun können wir λ ebenfalls auf einen bestimmten Wert setzen und die logistische Gleichung endlos iterieren, um zu sehen, gegen welchen Wert x_n schliesslich konvergiert. Mit dem Computer können natürlich nicht unendlich viele Iterationen durchgeführt werden, aber wenn n gross genug ist, kommt man doch sehr nahe an den tatsächlichen Konvergenzwert. Genau dieser Prozess ist nun in Abbildung 8.3 ersichtlich. Sie zeigt einen Plot, auf dem die horizontale Achse die verschiedenen Werte von λ im Bereich für $0 \leq \lambda \leq 4$ annimmt und die vertikale Achse anzeigt, auf welchem Wert x_n konvergiert, wenn n gegen unendlich läuft. Dabei werden einige Eigenschaften von der Iteration der logistischen Gleichung ersichtlich:

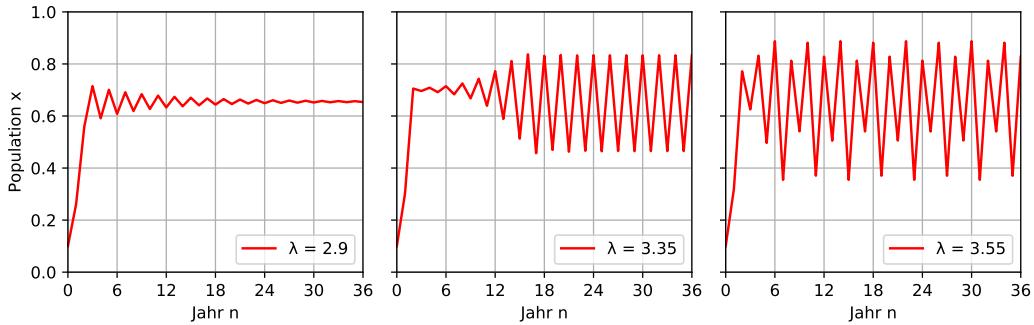


Abbildung 8.4: Iteration der logistischen Gleichung. Für $\lambda = 2.9$ konvergiert x_n gegen ≈ 0.62 . Im Kontrast dazu oszilliert x_n für $\lambda = 3.35$ und zwischen zwei und für $\lambda = 3.55$ sogar zwischen vier Werten.

- für $0 \leq \lambda \leq 1$ konvergiert x_n gegen 0
- für $1 \leq \lambda \leq 3$ konvergiert x_n gegen einen fixen Wert
- für $3 \leq \lambda \leq 4$ scheint x_n nicht mehr gegen einen fixen Wert zu konvergieren. Stattdessen gibt es diese Verzweigungen, die darauf hindeuten, dass x_n zuerst bis $\lambda \approx 3.4$ zwischen zwei Werten hin und her oszilliert, dann bis $\lambda \approx 3.6$ zwischen vier Werten, dann 8, 16, usw. Dieses Phänomen wird auch “Periodenverdoppelung” genannt.
- für $\lambda > 4$ gibt es scheinbar nichts mehr. Das kommt daher, dass x_n ab da nur noch divergiert.

Das doch eher spezielle Verhalten von $3 < \lambda < 4$ wird auch in Abbildung 8.4 ersichtlich, wenn man die Werte der einzelnen Iteration plottet. Im ersten Plot mit $\lambda = 2.9$ sieht man, dass x_n zuerst oszilliert, aber schliesslich gegen einen fixen Wert konvergiert. Auf dem zweiten Plot mit $\lambda = 3.35$ scheint x_n nach einigen Iterationen nur noch zwischen zwei Werten zu oszillieren und beim dritten Plot mit $\lambda = 3.55$ sogar zwischen vier Werten. Dieses Oszillieren zeigt sich in Abbildung 8.3 durch die Verzweigungen oder auch “Bifurkationen”. Darum wird sie auch “Bifurkationsdiagramm” genannt.

Auf dem Bifurkationsdiagramm sind noch mehr interessante Eigenschaften der logistischen Gleichung erkennbar. Wenn wir, wie in Abbildung 8.5, in die Bereiche hineinzoomen, wo die Bifurkationen immer dichter werden, finden wir Gebilde, welche wieder fast genau gleich aussehen. Tatsächlich könnte man endlos immer weiter in diese Bereiche hineinzoomen und würde immer wieder auf solche ähnlichen Muster stossen. Diese Selbstähnlichkeit ist eine typische Eigenschaft von Fraktalen. Das Bifurkationsdiagramm der logistischen Gleichung ist also ein Fraktal. Es gibt sogar einen Zusammenhang zwischen der logistischen Gleichung und einem anderen berühmten Fraktal, der Mandelbrotmenge. Mehr dazu später in Abschnitt 8.4.

Des weiteren ist auf dem Bifurkationsdiagramm zu sehen, dass für jede weitere Periodenverdoppelung eine geringere Erhöhung von λ notwendig ist. Diese rapide zunehmende Dichte der Periodenverdoppelungen führt dazu, dass wenn λ den Wert ≈ 3.57 überschreitet, die Verdoppelungen aufhören und x_n nicht mehr oszilliert. Stattdessen nimmt x_n nur noch scheinbar willkürliche Werte in bestimmten Bereichen an, man spricht von Chaos. Interessanterweise gibt es aber auch für Werte von $\lambda \gtrsim 3.57$ wieder gewissen Bereiche, wo wieder periodisches Verhalten auftritt. Ein Beispiel für

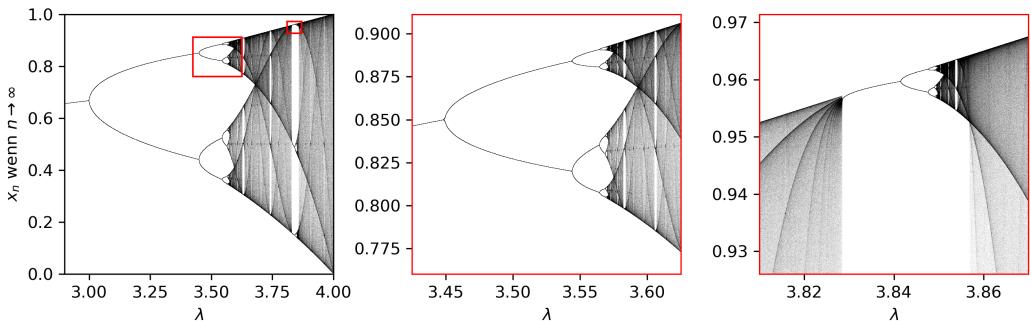


Abbildung 8.5: Das Bifurkationsdiagramm ist ein Fraktal. Die mittlere Grafik verdeutlicht mit einem Zoom bei $\lambda = 3.525$ die Selbstähnlichkeit. Auf der rechten Grafik sieht man mit einem Zoom bei $\lambda = 3.84$, dass sich immer wieder das selbe Muster finden lässt.

einen solchen Bereich sieht man gut in Abbildung 8.5 bei $\lambda \approx 3.83$. Dort gibt es zuerst eine Dreier-Periode, dann wieder die Periodenverdoppelungen, wobei die Dichte der Verdoppelungen wieder rapide zunimmt. Schliesslich bricht wieder das Chaos aus bei $\lambda \approx 3.85$.

8.3 Stabilität der logistischen Gleichung

In den bisherigen Grafiken ist leider nicht ersichtlich, warum die logistische Gleichung dieses Iterationsverhalten aufweist. Glücklicherweise kennen wir aber aus dem Kapitel 1.5 bereits Werkzeuge zur Stabilitätsbetrachtung für das Iterieren von Funktionen. Diese können wir jetzt einsetzen, damit wir dieses Verhalten besser verstehen können.

Kurze Auffrischung: Beim Iterieren einer Funktion gibt es manchmal Fixpunkte. Jeder Fixpunkt muss die Bedingung $f(x^*) = x^*$ erfüllen, das heisst wenn man die Funktion mit diesem Wert iteriert, passiert gar nichts, weil immer wieder der gleiche Wert herauskommt. Grafisch sind diese Punkte die Schnittpunkte vom Funktionsplot mit der 45° -Geraden durch den Ursprung. Zusätzlich gibt es die Stabilitätsbedingung $|f'(x^*)| < 1$. Wenn diese erfüllt ist, handelt es sich um einen stabilen Fixpunkt. Er hat eine “anziehende” Wirkung. Grafisch sieht man das am Winkel der Tangente in den Fixpunkten von $f(x)$, welcher zwischen $+45^\circ$ und -45° sein muss.

Wir können nun auf das uns bereits bekannte Spinnwebdiagramm zurückgreifen, um grafisch diese Fixpunkte zu finden. Beim Betrachten der logistischen Gleichung $f(x) = \lambda x(1 - x)$ sehen wir, dass es sich dabei eigentlich nur um eine Parabel mit den Nullstellen 0 und 1 handelt. Der Faktor λ bestimmt die Öffnung der Parabel.

Abbildung 8.6 zeigt nun das Spinnwebdiagramm, welches wir beim Iterieren der logistischen Gleichung erhalten. Auf der linken Grafik mit $\lambda = 0.75$ sehen wir, dass x_n gegen den Fixpunkt bei 0 konvergiert. Dies war bereits in Abbildung 8.2 ersichtlich, hier sehen wir aber zusätzlich noch, dass x_n eben gegen diesen Punkt konvergiert, weil es sich dabei um einen stabilen Fixpunkt handelt. Auf der mittleren Grafik mit $\lambda = 1.8$ ist nun für den Fixpunkt im Ursprung die Stabilitätsbedingung nicht mehr erfüllt. Dafür ist sie jetzt für den zweiten Fixpunkt erfüllt, was dazu führt, dass die Funktion beim Iterieren gegen diesen Wert konvergiert. Auf der rechten Grafik mit $\lambda = 2.8$ ist wieder der zweite Fixpunkt stabil, hier “umkreist” die Linie aber den Fixpunkt und konvergiert langsam gegen ihn. Auf Abbildung 8.2 hat sich dieses Verhalten als “Einschwingen” geäussert.

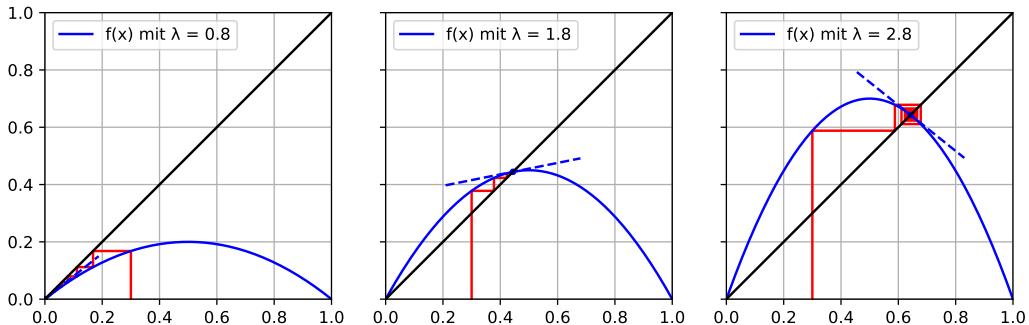


Abbildung 8.6: Spinnwebplot ohne Oszillation. Für alle drei Werte von λ gibt es einen stabilen Fixpunkt. Auf der rechten Grafik mit $\lambda = 2.8$ ist sichtbar wie x_n zuerst über- und dann langsam einschwingt.

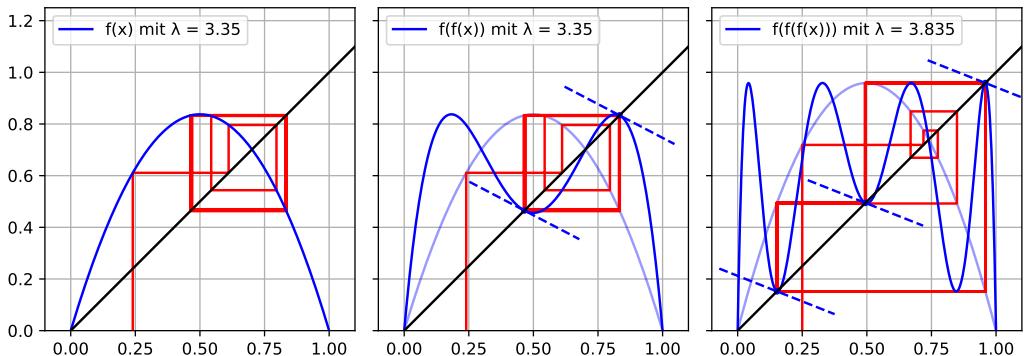


Abbildung 8.7: Spinnwebplot mit oszillierenden x_n . Die mittlere Grafik zeigt, wie die Verschachtelung $f(f(x))$ mit $\lambda = 3.35$ zwei stabile Fixpunkte hervorbringt und damit die Oszillation zwischen den zwei Punkten in der linken Grafik erklärt. In der rechten Grafik zeigt eine weitere Verschachtelung zu $f(f(f(x)))$ mit $\lambda = 3.835$ drei stabile Fixpunkte, was die entsprechende Dreier-Periode aus dem Bifurkationsdiagramm (Abbildung 8.3) erklärt.

Nun haben wir eine gute Erklärung für das Iterationsverhalten der logistischen Gleichung für $0 < \lambda < 3$. Aber was passiert nun, wenn $\lambda > 3$ wird, wo es beginnt zwischen mehreren Punkten zu oszillieren? Wenn wir die Stabilitätsbedingung $|f'(x^*)| < 1$ für Fixpunkte von $f(x)$ mit $\lambda > 3$ anwenden sehen wir, dass für eine einzelne Iteration der logistischen Gleichung kein stabiler Fixpunkt mehr existiert. Wenn wir jetzt aber zwei Iterationen der logistischen Gleichung machen, also $f(f(x))$ wobei $f(x) = \lambda x(1-x)$, dann bekommen wir ein Polynom vierten Grades:

$$f(f(x)) = \lambda[\lambda x(1-x)][1 - [\lambda x(1-x)]] = \lambda^2 x(1-x)(\lambda x^2 - \lambda x + 1). \quad (8.3)$$

Diese neue Funktion bringt mit der 45° -Geraden neue Schnittpunkte und damit auch neue Fixpunkte mit sich. Tatsächlich finden wir unter diesen neuen Fixpunkten auch solche, welche die Stabilitätsbedingung erfüllen für Werte von $\lambda > 3$.

Auch das wollen wir uns nochmal grafisch mit Abbildung 8.7 veranschaulichen. Auf der linken Grafik ist das Spinnwebdiagramm der logistischen Gleichung mit $\lambda = 3.35$ und $f(x)$ abgebildet. Offenbar gibt es eine Oszillation zwischen zwei Punkten. Jedoch ist nicht sichtbar, warum das so sein sollte, es gibt da keinen stabilen Fixpunkt zu sehen. Auf der mittleren Grafik ist nun nochmal das gleiche Spinnwebdiagramm, jetzt ist aber zusätzlich noch die verschachtelte Funktion $f(f(x))$ sichtbar. Darauf wird deutlich, warum die Oszillation eben genau zwischen diesen beiden Punkten entsteht. Es handelt sich dabei um zwei Fixpunkte, welche erst mit der Verschachtelung von $f(x)$ sichtbar werden. Würde man λ nun weiter erhöhen, würden auch diese beiden Fixpunkte wieder instabil werden, dafür vier neue stabile Fixpunkte bei einer erneuten Verschachtelung erscheinen, also bei $f(f(f(f(x))))$. Das geht so weiter bis zum Anfang des Chaos, wo es keine stabilen Fixpunkte mehr gibt. Wenn wir aber weiter durch das Chaos gehen, wie in Kapitel 8.2 zu sehen war, immer wieder diese kleinen Fenster, wo x_n wieder periodisch wird. Da gibt es zum Beispiel bei $\lambda \approx 3.835$ eine Dreier-Periode, was vermuten lässt, dass für diesen Wert von λ durch die dreifache Verschachtelung von $f(x)$ wieder drei stabile Fixpunkte entstehen. Tatsächlich gibt es, wie auf der rechten Grafik in Abbildung 8.7 zu sehen ist, durch $f(f(f(x)))$ und $\lambda = 3.835$ drei stabile Fixpunkte.

8.4 Beispiele von verwandten Funktionen

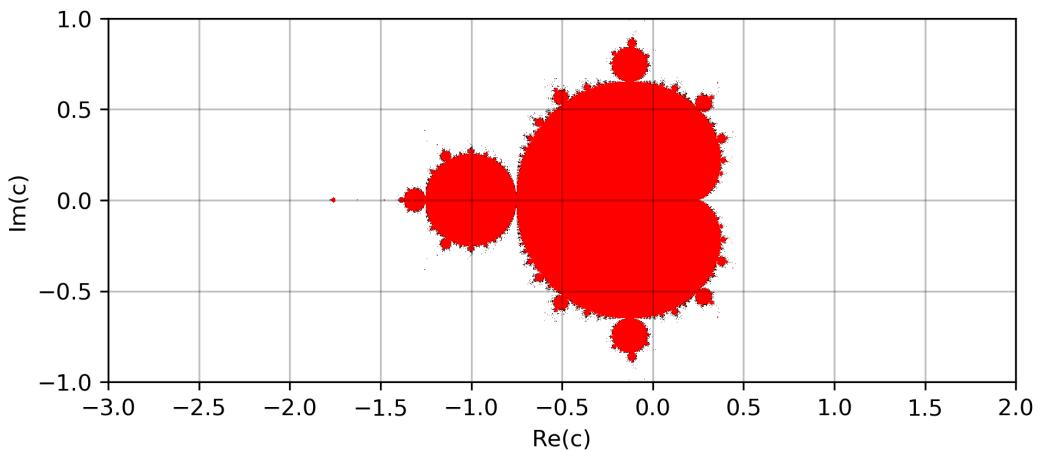


Abbildung 8.8: Mandelbrotmenge

8.4.1 Mandelbrotmenge

In Kapitel 8.2 haben wir bereits gesehen, dass das Bifurkationsdiagramm der logistischen Gleichung ein Fraktal ist. Eines der bekanntesten Fraktale ist die Mandelbrotmenge, deren Gleichung der logistischen Gleichung sehr ähnlich ist. Die Iterationsgleichung der Mandelbrotmenge lautet

$$z_{n+1} = z_n^2 + c, \quad (8.4)$$

wobei z_n und c komplexe Zahlen sind und der Einfachheit halber $z_0 = 0$ gesetzt werden kann. Zum Vergleich, die logistische Gleichung hat ausmultipliziert die Form $x_{n+1} = -\lambda x_n^2 + \lambda x_n$. Wie bei der lo-

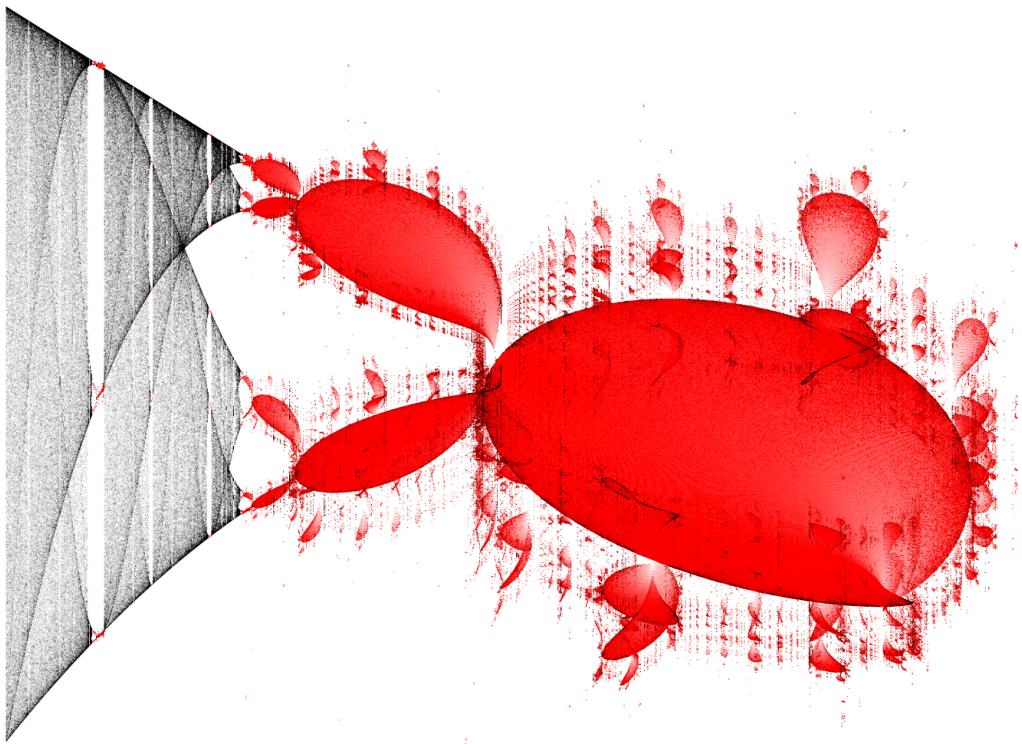


Abbildung 8.9: Dreidimensionales Bifurkationsdiagramm der Mandelbrotmenge. Das Bild wird aus einem flachen Winkel leicht von oben angeschaut. Die horizontale Ebene repräsentiert die Werte von c , siehe Abbildung 8.8 zum Vergleich. Die vertikale Achse zeigt die Werte, die $\Re(z_n)$ annimmt wenn $n \rightarrow \infty$. Rote Farbe bedeutet, dass $\Re(z_n)$ eine endliche Periode hat, schwarze Farbe bedeutet chaotisches Verhalten. Besonders interessant ist das Gebilde auf der linken Seite, das dem Bifurkationsdiagramm der logistischen Gleichung sehr ähnlich sieht.

gistischen Gleichung gibt es auch bei der Gleichung der Mandelbrotmenge wieder bestimmte Werte von c , bei der z_n entweder divergiert, konvergiert, oszilliert oder in chaotisches Verhalten ausbricht. Jeder Wert von c , für den z_n nicht divergiert, ist Teil der Mandelbrotmenge. Nun können wir, ähnlich wie schon beim Bifurkationsdiagramm der logistischen Gleichung, auf der komplexen Ebene für jeden Wert von c das Verhalten der Gleichung der Mandelbrotmenge darstellen. Dazu färben wir den Bereich, in dem z_n nicht divergiert, rot ein. Damit sehen wir in Abbildung 8.8 die Mandelbrotmenge. Auf dieser zweidimensionalen Darstellung ist jedoch nicht zu sehen, welche Werte z_n annimmt. Darum nehmen wir jetzt die dritte Dimension zur Hilfe um, wie schon beim Bifurkationsdiagramm der logistischen Gleichung, auf der vertikalen Achse darzustellen, auf welchem Wert sich der Wert von z_n schlussendlich eingependelt. Oder eben auch nicht, wenn es oszilliert oder sogar chaotisch wird. Da wir nur drei Dimensionen zur Verfügung haben, beschränken uns dabei auf den Realteil von z_n . Das Ergebnis davon ist in Abbildung 8.9 zu sehen. Die Farben geben Auskunft über die Periodizität des jeweiligen Punktes. Rote Farbe bedeutet, dass $\Re(z_n)$ für den jeweiligen Wert von c eine endliche Periode hat. Schwarz bedeutet chaotisches Verhalten oder möglicherweise auch, dass die

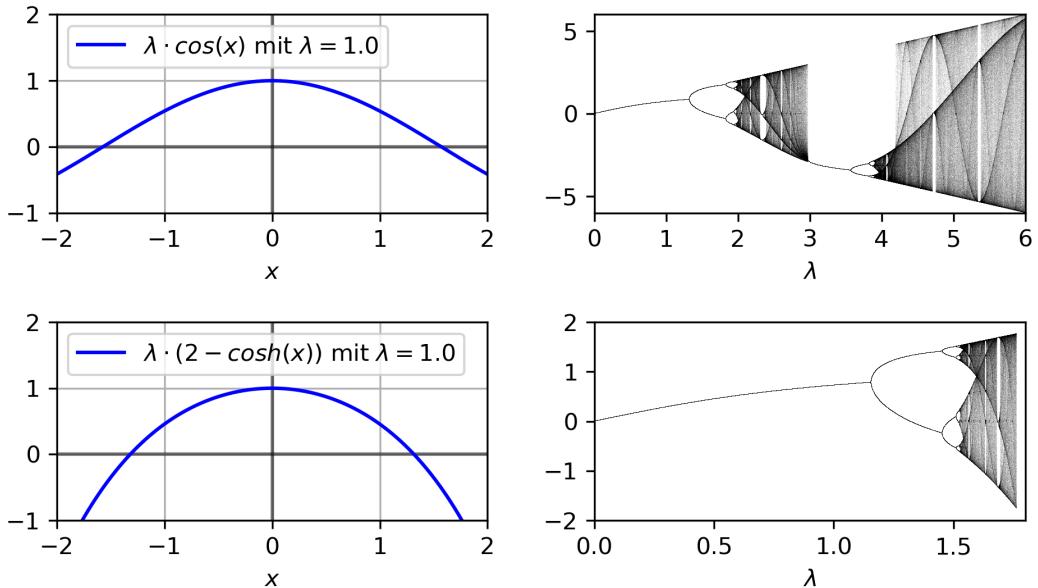


Abbildung 8.10: Bifurkationsdiagramme von $\lambda \cdot \cos(x)$ und $\lambda \cdot (2 - \cosh(x))$. Links ist jeweils der Funktionsplot und rechts das dazugehörige Bifurkationsdiagramm.

Periode so gross ist, dass der Computer sie bei der Berechnung nicht als endlich erkannt hat. Auf der reellen Achse ist deutlich ein Gebilde zu sehen, welches dem Bifurkationsdiagramm der logistischen Gleichung sehr ähnlich sieht. Ebenfalls zu sehen ist, dass die kreisförmigen “Plattformen” sich regelmäßig verdoppeln, weil $\Re(z_n)$ zwischen immer mehr Werten hin und her oszilliert.

8.4.2 Universelle Eigenschaft

Dieses Verhalten mit den Periodenverdoppelungen bis es schliesslich chaotisch wird und dann immer wieder kurzen oszillierenden Fenstern im Chaos ist keineswegs eine Eigenschaft, die nur die logistische Gleichung besitzt. Man findet dieses Verhalten auch beim Iterieren von vielen anderen nichtlinearen Funktionen. Ein typisches Merkmal, an dem man Funktionen erkennt die diese Eigenschaften aufweisen, sind “Hügel” im Funktionsplot. Als Beispiel dazu sind in Abbildung 8.10 links die beiden Funktionen $\lambda \cdot \cos(x)$ und $\lambda \cdot (2 - \cosh(x))$ abgebildet. Beide Funktionen haben, wie auch schon die Funktion der logistischen Gleichung, einen Hügel. Rechts sind die Bifurkationsdiagramme dieser beiden Funktionen abgebildet. Die Ähnlichkeiten zum Bifurkationsdiagramm der logistischen Gleichung sind deutlich sichtbar. Besonders interessant wird es, wenn man bei diesen Funktionen im Bifurkationsdiagramm die Periodenverdoppelungen genauer anschaut. Denn wenn man das Verhältnis der Breite einer Periode zur Breite der vorherigen Periode im Limit anschaut, also $\lim_{n \rightarrow \infty} \left(\frac{\lambda_{n-1} - \lambda_{n-2}}{\lambda_n - \lambda_{n-1}} \right)$ wobei λ_n der Wert von λ bei der n -ten Periodenverdoppelung ist, dann kommt man auf die Zahl $\delta \approx 4.6692$. Diese Zahl hat den Namen *Feigenbaum-Konstante*. Es ist irrelevant, welche Kaskade von Periodenverdoppelungen bei diesen Funktionen gewählt wird, im Limit kommt schlussendlich immer diese Zahl heraus. Die Feigenbaum-Konstante findet man auf die genau gleiche Art auch in der logistischen Gleichung und in vielen anderen Funktionen, die einen einzelnen

“Hügel” haben, wieder. Dadurch haben alle diese Funktionen durch die Feigenbaum-Konstante eine universale Eigenschaft. Es ist überhaupt nicht offensichtlich warum das so sein sollte und es hat Mitchell Feigenbaum, den Entdecker dieser Konstante, und andere Mathematiker viel Arbeit gekostet, diese Eigenschaft zu erforschen.

Abschliessend können wir so aus diesen Erkenntnissen direkt noch etwas für die weitere Numerik mitnehmen. Offenbar können sich Funktionen beim Iterieren sehr unerwartet verhalten. Gerade in der Numerik bauen viele Verfahren auf iterativen Algorithmen auf. Deswegen schadet es sicherlich nicht, wenn man sich bewusst ist, dass beim Iterieren von scheinbar einfachen Funktionen so ein “wildes” Verhalten entstehen kann.

9

Stabile Berechnung von Legendre-Polynomen

Patrick Elsener

9.1 Einleitung

Die zugeordneten Legendre-Polynome (engl. *associated Legendre polynomials*) P_l^m besitzen zwei Parameter, m und l . Mittels diesen beiden Parameter lassen sich einige Rekursionsbeziehungen (engl. *recurrence relation*) aufstellen. Zur Berechnung der Werte von zugeordneten Legendre-Polynomen wird normalerweise auf eine solche Rekursionsbeziehung zurückgegriffen. Die Rekursionsbeziehungen werden bevorzugt, da deren Alternativen Nachteile mit sich bringen. So ist beispielsweise die Verwendung der geschlossenen Form

$$P_l^m(x) = (-1)^m \cdot 2^l \cdot (1 - x^2)^{m/2} \cdot \sum_{k=m}^l \frac{k!}{(k-m)!} \cdot x^{k-m} \cdot \binom{l}{m} \binom{\frac{l+k-1}{2}}{l} \quad (9.1)$$

für ein zugeordnetes Legendre-Polynom P_l^m rechnerisch zu aufwendig. Andererseits ist es möglich, die Polynomfunktionen für die verschiedenen Werte von l und m aufzustellen. Dies ist jedoch für höhere zugeordnete Legendrepolygone nicht praktikabel.

Von den Rekursionsbeziehungen gibt es eine ganze Liste voll und es ist gut möglich, dass nicht alle davon numerisch stabil sind. Es ist daher möglich, dass bei einer unglücklichen Wahl einer solchen Rekursionsbeziehung numerische Probleme auftreten können, die zu völlig falschen Resultaten führen. Dass ein solches numerisches Problem auftreten kann, wird leider oft vergessen. Zusätzlich ist es eine anspruchsvolle Aufgabe, solche numerischen Instabilitäten vorzeitig zu erkennen. Aus diesen Gründen ist es nicht verwunderlich, dass sogar namhafte Bibliotheken numerisch instabile Implementationen enthalten. So ist beispielsweise die Implementation der zugeordneten Legendre-Polynome auf Wolfram Alpha [6] numerisch nicht stabil, wie gut in der Abbildung 9.1 zu sehen ist. Aus diesen Gründen befasst sich dieses Kapitel mit der Stabilität oder eben Instabilität der Rekursionsbeziehungen für zugeordnete Legendre-Polynome.

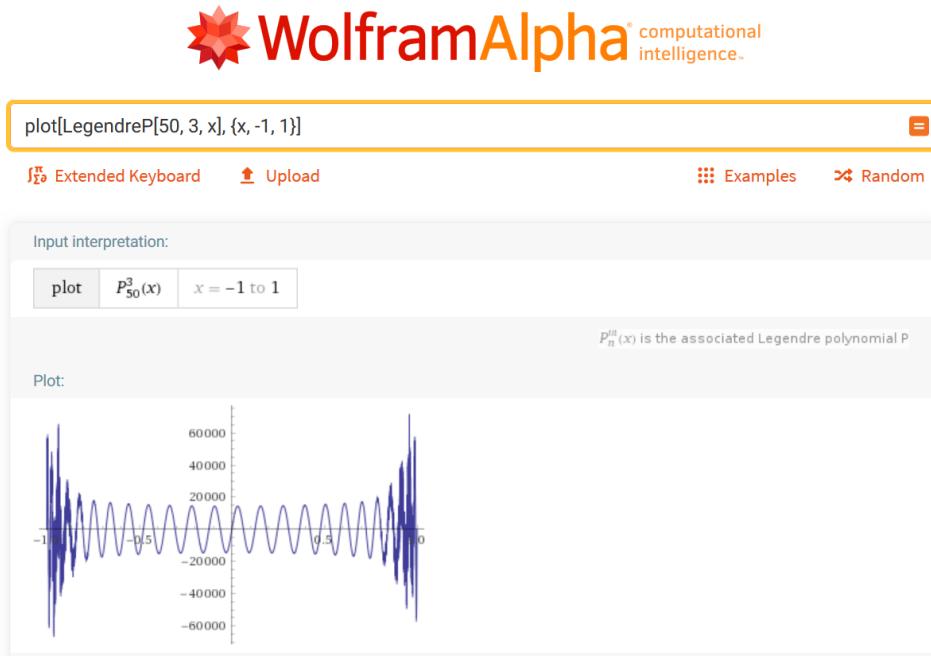


Abbildung 9.1: Von Wolfram Alpha [6] generierter Graph des zugeordneten Legendre-Polynoms mit $l = 50$ und $m = 3$. Deutliche numerische Instabilitäten nahe den Randbereichen.

9.2 Problemstellung

Die zugeordneten Legendre-Polynome sind die Lösungen der allgemeinen *Legendre-Gleichung*

$$(1-x^2) \frac{d^2y}{dx^2} - 2x \frac{dy}{dx} + \left[l(l+1) - \frac{m^2}{1-x^2} \right] y = 0 \quad (9.2)$$

[2] [1]. Die Lösung dieser Gleichung lässt sich mittels Polynomansatz

$$y(x) = a_0 + a_1 x + \dots + a_n x^n \quad (9.3)$$

finden und führt auf die geschlossene Form (9.1) aus Abschnitt 9.1. Die zugeordneten Legendre-Polynome sind zudem nur auf dem Intervall $[-1, 1]$ definiert. Verwendung finden diese Polynome vor allem als Teil der Kugelflächenfunktionen (engl. *spherical harmonics*) [5].

Ein zugeordnetes Legendre-Polynom wird mit P_l^m bezeichnet und besitzt zwei Parameter, l und m . Dabei gilt für die beiden Parameter, dass $l \geq 0$ und $m = 0, \dots, l$. Der Parameter l ist für den Grad des zugeordneten Legendre-Polynoms verantwortlich, m hingegen wird mancherorts, für Polynome eher ungewöhnlich, als *Ordnung* des Polynoms bezeichnet. Aus den Bedingungen für die beiden Parameter l und m ergibt sich eine Struktur möglicher zugeordneter Legendre-Polynome. Der Anfang dieser Struktur ist in Abbildung 9.2 dargestellt.

Wie im Abschnitt 9.1 bereits erwähnt, gibt es Rekursionsbeziehungen für die zugeordneten Legendre-Polynome. Der englische Wikipedia-Artikel [3] zu den zugeordneten Legendre-Polynomen führt eine Liste solcher Rekursionsformeln. Die ersten beiden Rekursionsformeln des Wikipedia-

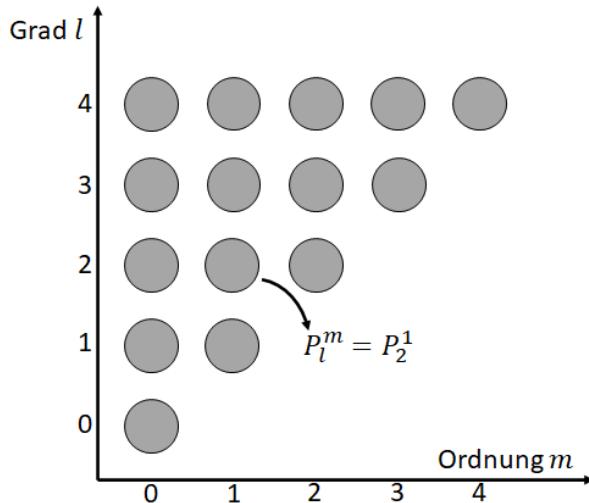


Abbildung 9.2: Ausschnitt aus der Struktur der möglichen zugeordneten Legendre-Polynome.

Artikels sind wohl die gängigsten. Die erste Rekursionsformel

$$(l - m + 1)P_{l+1}^m(x) = (2l + 1)xP_l^m(x) - (l + m)P_{l-1}^m(x) \quad (9.4)$$

verläuft in der Richtung des Parameters l , also in der Richtung des Grades des Legendre-Polynoms. Diese Rekursionsbeziehung nennen wir ab jetzt l -Rekursion. Die zweite Rekursionsformel

$$2mxP_l^m(x) = -\sqrt{1 - x^2} [P_l^{m+1}(x) + (l + m)(l - m + 1)P_l^{m-1}(x)] \quad (9.5)$$

hingegen verläuft in der Richtung des Parameters m . Analog zur vorherigen Rekursionsbeziehung nennen wir diese m -Rekursion. Zu erwähnen ist noch, dass die m -Rekursion in negativer m -Richtung verläuft, also in Richtung abnehmender m -Werte. Dies führt dazu, dass die m -Rekursion nicht wie in Gleichung (9.5) dargestellt verwendet werden kann. Sie muss daher nach $P_l^{m-1}(x)$ aufgelöst werden. Darauf wird später noch etwas genauer eingegangen.

Beim Erstellen der Graphen (siehe Abbildung 9.3 und Abbildung 9.4) mittels diesen beiden Rekursionsformeln fällt auf, dass keine numerischen Instabilitäten ersichtlich sind, wie dies beim Graphen von Wolfram Alpha der Fall ist (vergleiche Abbildung 9.1). Die beiden Graphen (Abbildungen 9.3 und 9.4) zeigen, dass die Rekursionsformeln nicht die alleinige Ursache für die numerischen Instabilitäten des Legendre-Polynom-Graphen von Wolfram Alpha (Abbildung 9.1) sein können. Trotzdem lohnt es sich die beiden Rekursionsformeln etwas genauer zu untersuchen. Sie können nämlich mitverantwortlich sein für numerische Instabilitäten, indem sie numerische Fehler aus einer anderen Quelle begünstigen.

9.3 Lösung

9.3.1 Anfangswerte

Jede Rekursionsbeziehung braucht ihre Anfangswerte. Sowohl für die l -Rekursion (9.4) als auch für die m -Rekursion (9.5) werden je zwei Anfangswerte benötigt. Für die l -Rekursion (9.4) werden die

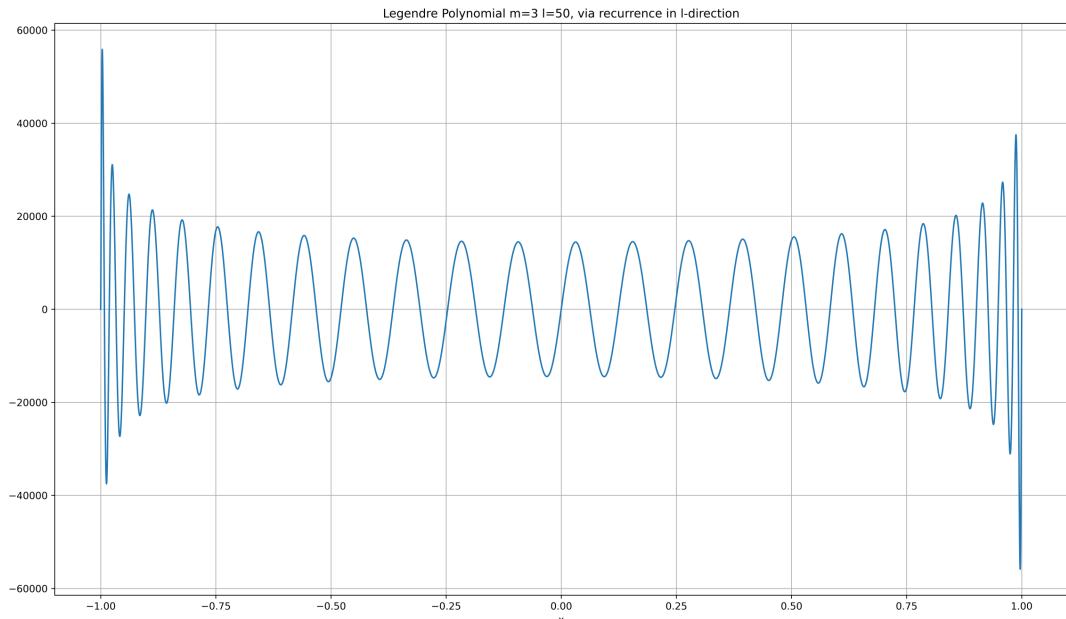


Abbildung 9.3: Zugeordnetes Legendre-Polynom mit $l = 50$ und $m = 3$, berechnet mit der Rekursionsformel in l -Richtung.

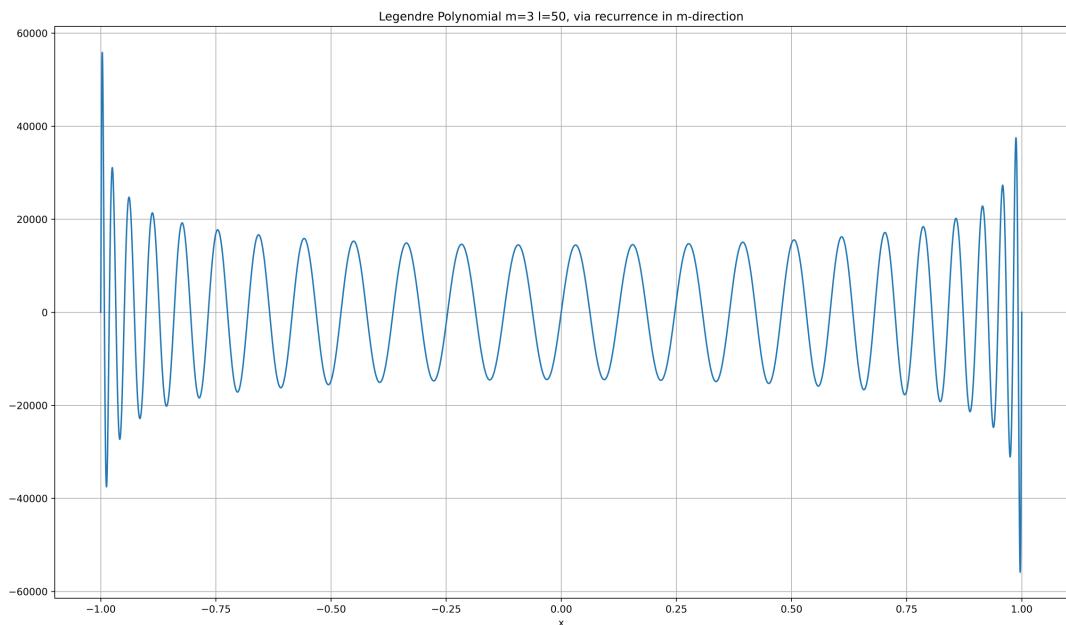


Abbildung 9.4: Zugeordnetes Legendre-Polynom mit $l = 50$ und $m = 3$, berechnet mit der Rekursionsformel in m -Richtung.

Anfangswerte $P_l^l(x)$ und $P_{l+1}^l(x)$ benötigt. Für die m -Rekursion (9.5) werden aus dem Anfangswert $P_l^l(x)$ die beiden weiteren Anfangswerte $P_{l+1}^l(x)$ und $P_{l+1}^{l+1}(x)$ berechnet. Diese Anfangswerte lassen sich mittels der Formeln

$$\begin{aligned} P_l^l(x) &= (-1)(2l-1)!!(1-x^2)^{l/2}, \\ P_{l+1}^l(x) &= x(2l+1)P_l^l(x), \\ P_{l+1}^{l+1}(x) &= -(2l+1)\sqrt{1-x^2}P_l^l(x) \end{aligned} \quad (9.6)$$

berechnen, die sich aus der geschlossenen Form (9.1) herleiten lassen. Bei der Berechnung des Anfangswertes $P_l^l(x)$ gilt zu beachten, dass $!!$ für die Doppelfakultät steht und nicht für die Fakultät der Fakultät. Im Gegensatz zur einfachen Fakultät werden bei der Doppelfakultät nur die geraden, respektive ungeraden Faktoren zwischen 1 und der Zahl selber berücksichtigt. Berechnen lässt sich die Doppelfakultät wie folgt:

$$k!! = \begin{cases} 1 \cdot 3 \cdot \dots \cdot k & \text{falls } k \text{ ungerade} \\ 2 \cdot 4 \cdot \dots \cdot k & \text{falls } k \text{ gerade.} \end{cases} \quad (9.7)$$

In der Einleitung 9.1 wurde bereits erwähnt, dass die Rekursionsformeln weniger Rechenaufwand benötigen als die Formel für die geschlossene Form (9.1). Dies gilt auch noch unter Einbezug dieser Anfangswerte.

9.3.2 m -Rekursion

Für die Abbildung 9.4 wurde das zugeordnete Legendre-Polynom mit Grad 50 und Ordnung 3 mittels der m -Rekursion (9.5) berechnet. Die Anfangswerte $P_{l+1}^l(x)$ und $P_{l+1}^{l+1}(x)$ geben vor, dass bei Verwendung dieser Rekursionsbeziehung diese in negativer m -Richtung verläuft. Dazu muss die Formel für die m -Rekursion (9.5) nach $P_l^{m-1}(x)$ umgeformt werden. Daraus entsteht die neue Formel für die m -Rekursion:

$$P_l^{m-1}(x) = \left[\frac{2mxP_l^m(x)}{-\sqrt{1-x^2}} - P_l^{m+1} \right] \frac{1}{(l+m)(l-m+1)}. \quad (9.8)$$

Wird die umgeformte Rekursionsgleichung (9.8) auf numerische Instabilität untersucht, fällt sofort der Term $\sqrt{1-x^2}$ auf. Dieser Term deutet darauf hin, dass nahe an den Intervallsgrenzen ($x \rightarrow \pm 1$) Auslöschung auftreten kann. Dies alleine ist noch nicht allzu schlimm. Jedoch ist es so, dass die zugeordneten Legendre-Polynome für $m > 0$ an den Intervallsgrenzen Null ergeben. Das heisst, dass die Differenz in der eckigen Klammer nahe den Intervallsgrenzen in etwa Null ergibt. Der Term $\sqrt{1-x^2}$, welcher von Auslöschung betroffen ist, wird nahe den Intervallsgrenzen sehr klein und bläst somit den Term $2mxP_l^m(x)$ auf, welcher dadurch selbst fehlerhaft wird. Die Differenz in der eckigen Klammer unterliegt somit einer quasi-Auslöschung. In der eckigen Klammer werden zwei fast gleich grosse Terme voneinander subtrahiert, wobei der erste Term durch die Auslöschung und das Aufblasen durch $\sqrt{1-x^2}$ selbst stark fehleranfällig ist.

Es stellt sich nun die Frage, warum diese numerische Instabilität nicht im Graphen 9.4 ersichtlich ist, wie beispielsweise in dem Graphen 9.1 den Wolfram Alpha erstellt. Eine Antwort darauf liefert der Term $\frac{1}{(l+m)(l-m+1)}$, welcher einen positiven Einfluss auf numerischer Fehler hat. Dadurch, dass die Rekursionsgleichung zwei Anfangswerte hat, ist $l \geq m+2$ gegeben. Damit lässt sich der Nenner dieses Terms auf ≥ 6 abschätzen, was wiederum bedeutet, dass $\frac{1}{(l+m)(l-m+1)} \leq \frac{1}{6}$ ist. Dadurch verkleinert sich der Wert der Differenz in der eckigen Klammer der Rekursionsgleichung und ein Teil des Fehlers wird somit eliminiert. Einige fehlerhaften Stellen hinter dem Komma verschwinden

somit. Diese Rekursionsgleichung bleibt jedoch fehleranfällig nahe den Intervallsgrenzen, falls numerische Fehler aus einer anderen Quelle einfließen, wie beispielsweise fehlhafte oder ungenaue Anfangswerte.

9.3.3 l -Rekursion

Für die Abbildung 9.3 wurde ebenfalls das zugeordnete Legendre-Polynom mit Grad 50 und Ordnung 3 berechnet, jedoch mittels l -Rekursion (9.4). Die Anfangswerte $P_l^l(x)$ und $P_{l+1}^l(x)$ zeigen, dass die Rekursion in positiver l -Richtung verläuft. Daher muss bei der Gleichung (9.4) für die l -Rekursion nur noch der Faktor $(l - m + 1)$ auf die rechte Seite gebracht werden. Dies führt zur neuen Formel für die l -Rekursion:

$$P_{l+1}^m(x) = \frac{(2l + 1)xP_l^m(x) - (l + m)P_{l-1}^m(x)}{(l - m + 1)}. \quad (9.9)$$

Im Gegensatz zur m -Rekursion ist der Term $\sqrt{1 - x^2}$ in der Formel für die l -Rekursion (9.9) nicht vorhanden. Auch sonst ist kein Term vorhanden, der nahe den Intervallsgrenzen eine numerische Instabilität hervorrufen könnte. Jedoch gilt es noch die Faktoren vor den zugeordneten Legendre-Polynomen zu untersuchen. Diese beiden Faktoren sind $\frac{(2l+1)}{(l-m+1)}$ und $\frac{(l+1)}{(l-m+1)}$. Gegeben durch die Anfangswerte ist mit dem gleichen Argument wie im vorherigen Abschnitt 9.3.2 wieder gegeben, dass $l \geq m + 2$ ist. Daraus lässt sich ableiten, dass $\frac{(2l+1)}{(l-m+1)} \geq 1$ für $m > 0$ ist und dass $\frac{5}{3} \leq \frac{(l+1)}{(l-m+1)} < 2$. Da die beiden Faktoren jeweils ≥ 1 sind, bedeutet dies, dass die Fehler in den Polynomen P_l^m und P_{l-1}^m vergrößert werden. Daraus lässt sich schließen, dass diese Rekursionsbeziehung fehleranfällig ist, falls numerische Fehler aus einer anderen Quelle einfließen, wie beispielsweise fehlhafte oder ungenaue Anfangswerte.

9.3.4 Welche Rekursionsformel soll nun verwendet werden?

Es stellt sich die Frage, welche der beiden Rekursionsformeln verwendet werden soll? Die Formel für die l -Rekursion (9.9)? Oder doch die Formel für die m -Rekursion (9.8)? Denn obwohl die beiden Graphen, welche mittels l -Rekursion (Abbildung 9.3) respektive mittels m -Rekursion (Abbildung 9.4) erstellt wurden, optisch gleich aussehen, gibt es Unterschiede. Diese Unterschiede sind in der Abbildung 9.5 ersichtlich. Der Graph in dieser Abbildung zeigt die Differenz der berechneten Werte für das zugeordnete Legendre-Polynom mit $l = 50$ und $m = 3$ zwischen der Berechnung mittels l -Rekursion und der Berechnung mittels m -Rekursion. Es ist deutlich zu sehen, dass sich die beiden Rekursionsformeln nahe den Intervallsgrenzen unterscheiden. Die Analyse der m -Rekursion aus Abschnitt 9.3.2 lässt vermuten, dass diese Rekursionsbeziehung dafür verantwortlich ist. Dies spricht dafür, die l -Rekursion zu verwenden. Ein weiterer Punkt, der für die l -Rekursion spricht, ist, dass die GNU Scientific Library [4] diese Implementation, mit den gleichen Anfangswerten wie in Abschnitt 9.3.1 beschrieben, verwendet. Die Implementation der GNU Scientific Library wird allgemein als numerisch stabil angesehen.

9.4 Folgerungen

In diesem Kapitel wurde aufgezeigt, dass sich nicht jede numerische Instabilität nur auf die Verwendung einer bestimmten Formel zurückführen lässt, dass jedoch eine Formel numerische Fehler aus einer anderen Quelle begünstigen kann. Beide untersuchten Rekursionsbeziehungen offenbarten

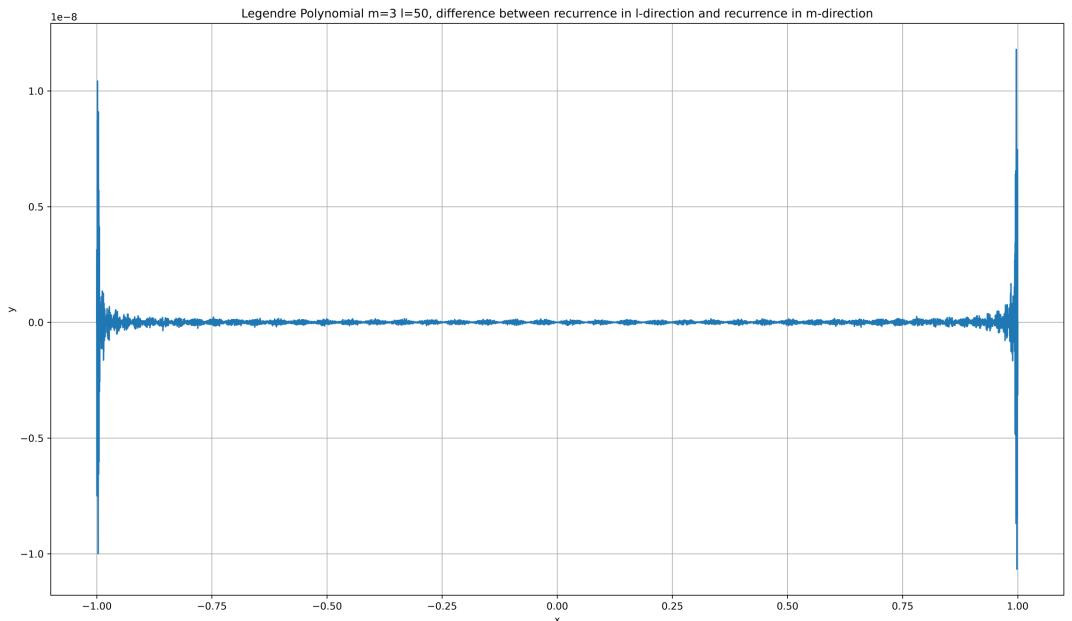


Abbildung 9.5: Differenz zweier zugeordneten Legendre-Polynome mit $l = 50$ und $m = 3$, einmal berechnet mittels l -Rekursion und einmal berechnet mittels m -Rekursion.

ihre Schwächen. Es gibt jedoch Hinweise, dass die Formel für die l -Rekursion die bessere Wahl gegenüber der Formel für die m -Rekursion darstellt. Daher ist zu empfehlen, nur Implementationen einer Programmbibliothek zu verwenden, die auf die l -Rekursion zurückgreifen.

Die Untersuchungen auf numerische Instabilität in diesem Kapitel zeigt klar auf, dass numerische Instabilitäten verschiedene Quellen haben können und erst einige Berechnungsschritte später auftauchen können. Eine einfache Auslöschung kann somit eine ganze Kette von Effekten mit sich bringen, die erst ganz am Schluss einer Berechnung ein Resultat verfälscht.

Literatur

- [1] *Associated Legendre Differential Equation – from Wolfram MathWorld*. 10. Mai 2020. URL: <https://mathworld.wolfram.com/AssociatedLegendreDifferentialEquation.html>.
- [2] *Associated Legendre Polynomial – from Wolfram MathWorld*. 10. Mai 2020. URL: <https://mathworld.wolfram.com/AssociatedLegendrePolynomial.html>.
- [3] *Associated Legendre polynomials - Wikipedia*. 10. Mai 2020. URL: https://en.wikipedia.org/wiki/Associated_Legendre_polynomials.
- [4] *GSL - GNU Scientific Library - GNU Project - Free Software Foundation*. 21. Mai 2020. URL: <https://www.gnu.org/software/gsl/>.
- [5] *Spherical Harmonic – from Wolfram MathWorld*. 10. Mai 2020. URL: <https://mathworld.wolfram.com/SphericalHarmonic.html>.

- [6] *Wolfram|Alpha: Computational Intelligence.* 10. Mai 2020. URL: <https://www.wolframalpha.com/>.

10

Gauss-Quadratur

Mike Schmid

10.1 Einleitung

Im Kapitel 4, Integration, wurden die Trapezregel und die Mittelpunktsregel für die numerische Integration, auch Quadratur genannt, erklärt. In diesem Kapitel wird eine weitere Methode, die Gauss-Quadratur, erarbeitet. Die Gauss-Quadratur ist ein Verfahren, welches ein bestimmtes Integral der Form

$$\int_a^b f(x) dx \quad (10.1)$$

mit der approximierten Summenformel

$$I = \sum_{i=0}^n A_i f(x_i) \quad (10.2)$$

annähert, wobei die Stützstellen x_i und die Gewichtung A_i von der gewählten Methode abhängen. Es werden dabei im Vergleich mit ähnlichen Verfahren viel weniger Funktionsauswertungen benötigt.

Methoden für die Quadratur lassen sich in zwei Gruppen unterteilen: *Newton-Cotes-Formeln* und *Gauss-Quadratur*. Die im Kapitel 4.1.2 und 4.1.3 beschriebenen Trapezregel und Mittelpunktsregel gehören zu der ersten Kategorie. Newton-Cotes-Formeln lassen sich dadurch erkennen, dass die Stützstellen auf der x -Achse gleichmäßig verteilt sind und eignen sich besonders für die Integration, wenn sich $f(x)$ effizient in kleinen Intervallen berechnen lässt oder bereits von Computern berechnet worden ist.

10.2 Problemstellung

10.2.1 Überlegung hinter der Gauss-Quadratur

Bei der Integration mit der Trapez-/Mittelpunktregel kann man die Genauigkeit der Annäherung verbessern, indem man mehr Teilintervalle berechnet. Dabei werden mehr Stützstellen auf der x -Achse benötigt und mehr Funktionsauswertungen müssen berechnet werden. Die Wahl der Position

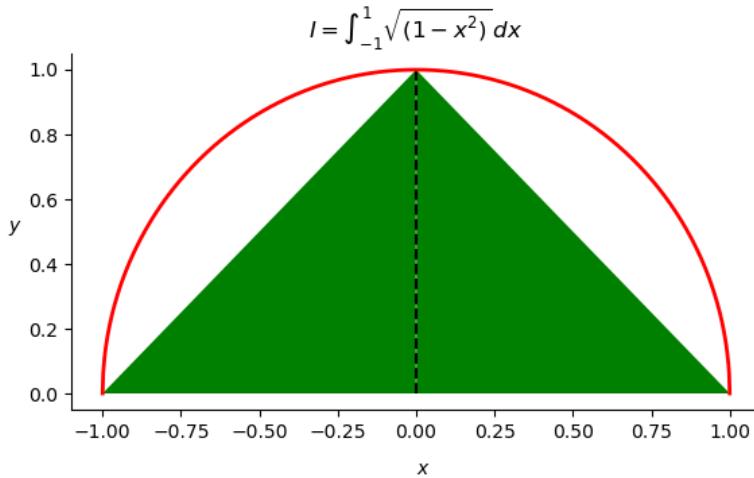


Abbildung 10.1: Integration mit der Trapezregel mit drei Stützstellen

und Anzahl Stützstellen stellt hierbei einer der beiden Freiheitsgrade in der Berechnung des Integrals dar. Der zweite Freiheitsgrad ist die Verwendung einer Gewichtung (die Wahl der Gewichte A_i in (10.2)) für eine spezifische Stützstelle. Das Hinzuziehen einer Gewichtung erlaubt es, in einem Teilintervall, in dem die Annäherung stärkere Abweichungen zum Graphen in diesem Intervall hat, diese Abweichung zu minimieren.

Beispiel: Integration einer Funktion mit der Trapezregel

Das folgende Beispiel illustriert die Integration für eine Funktion mit der Trapezregel und zeigt die Verbesserung der Genauigkeit unter Verwendung von mehr Stützstellen oder einer Gewichtung. In der Grafik 10.1 sieht man die Integration I der Funktion $f(x) = \sqrt{1 - x^2}$ mit der Trapezregel mit drei Stützstellen an den Punkten $x = -1, x = 0, x = 1$. Die Funktion $f(x)$ beschreibt hierbei einen Halbkreis um das Zentrum $x = 0$, dessen Fläche man geometrisch leicht ermitteln kann. Die Fläche eines Halbkreises mit Radius 1 ist $\frac{\pi}{2} \approx 1.570796$. Betrachtet man nun die grün eingefärbte Fläche, sieht man zwei gleichschenklige rechtwinklige Dreiecke mit der Seitenlänge 1, somit lässt sich die grüne Fläche als ein Quadrat mit der Seitenlänge 1 berechnen und erhält die Fläche 1.

Beispiel: Gewichtung der Stützstellen

Wird der Stützstelle $x = 0$ das Gewicht 1.2 gegeben, dann verhält sich die grüne Fläche wie zwei rechtwinklige Dreiecke mit den Seitenlängen $l_x = 1, l_y = 1.2$ und die resultierende Fläche lässt sich mit einem Rechteck mit den Seitenlängen $a = 1, b = 1.2$ berechnen. Man erhält die Fläche 1.2 welche bereits etwas näher am tatsächlichen Resultat $\frac{\pi}{2} \approx 1.570796$ ist.

Beispiel: Erhöhung der Anzahl Stützstellen

In der Grafik 10.2 werden vier Stützstellen verwendet und man erhält zwei Teilintervalle mit rechtwinkligen Dreiecken mit den Seitenlängen $l_x = 0.6666, l_y = 0.94281$ und einem Rechteck mit

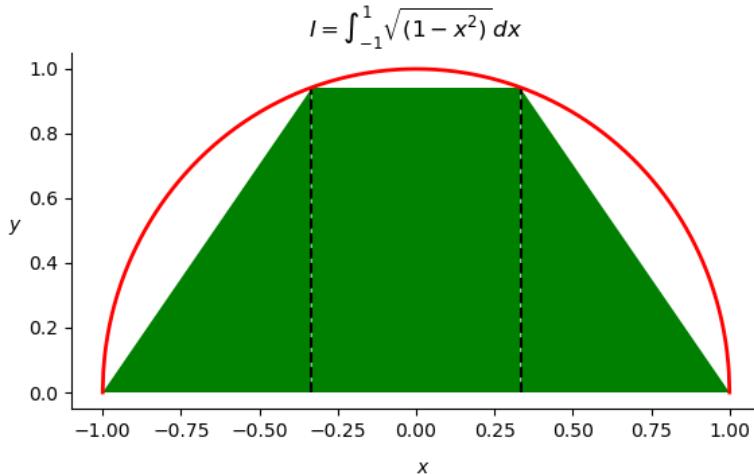


Abbildung 10.2: Integration mit der Trapezregel mit vier Stützstellen

den Seitenlängen $l_x = 0.6666, l_y = 0.94281$. Die resultierende Fläche lässt sich mit der Formel $A = 2 \cdot (l_x \cdot l_y)$ berechnen und man erhält das Resultat $A = 1.2571$. Auch diese Annäherung weicht noch vom eingentlichen Resultat $\frac{\pi}{2} \approx 1.570796$ ab.

Die Gauss-Quadratur basiert auf der Idee, dass die Stützstellen und deren Gewichtung so gewählt werden, dass das Resultat optimiert wird und möglichst genau berechnet werden kann. Dieses Verfahren erlaubt es, das Integral eines Polynoms vom Grad n mit nur $\frac{n+1}{2}$ Funktionsauswertungen exakt zu berechnen. Die Bedeutung dieser Eigenschaft wird im Abschnitt 10.3.2 genauer beschrieben.

10.2.2 Anwendungsbereiche der numerischen Integration

Die numerische Integration kann, wie im Kapitel 4 beschrieben, dann verwendet werden, wenn keine Stammfunktion in analytischer Form für eine Funktion gefunden werden kann. Ein bekanntes Beispiel dafür ist die Verteilungsfunktion

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt \quad (10.3)$$

der Standardnormalverteilung

$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \frac{x^2}{\sigma^2}} \quad (10.4)$$

oder auch Gauss'sche Glockenkurve, wie in der Grafik 10.3 dargestellt. Weitere Beispiele mathematischer Funktionen ohne elementare Stammfunktionen sind:

$$\frac{\sin(x)}{x}, \frac{x}{\sin(x)}, \frac{e^x}{x}, \frac{1}{\ln(x)}, \frac{x}{\ln(x)}, \ln(\sin(x)), e^x \cdot \ln(x).$$

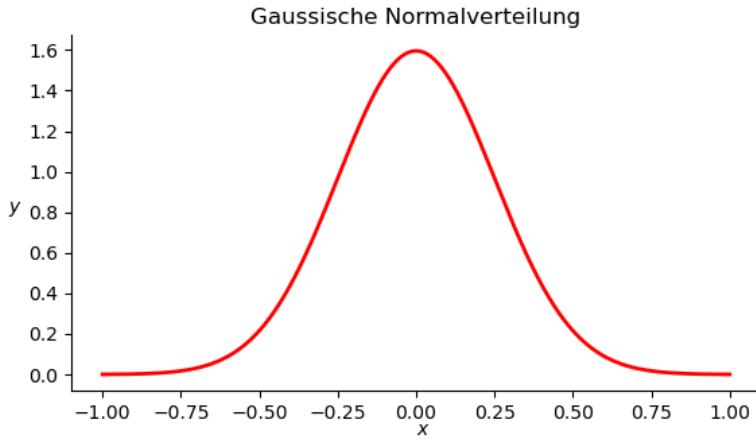


Abbildung 10.3: Darstellung der Normalverteilung mit $\sigma = 0.25$ und $\mu = 0$

Stützstellen x_i	Gewichte A_i
-0.861136	0.347855
-0.339981	0.652145
0.339981	0.652145
0.861136	0.347855

Tabelle 10.1: Werte für vier Stützstellen und deren Gewichte

10.3 Lösung

10.3.1 Anwendung der Gauss-Legendre-Formel

In diesem Unterabschnitt versuchen wir, die Verwendung der Gauss-Legendre-Formel in einem einfachen Beispiel herzuleiten, hierzu wird wieder die Funktion $f(x) = \sqrt{1-x^2}$ verwendet und wir berechnen das Integral mit vier Stützstellen. In der Tabelle 10.1 sind die Werte der Stützstellen und der Gewichte für dieses Beispiel angegeben. Die Berechnung der Stützstellen, der Gewichte und die verschiedenen Formen der Gauss-Quadratur wird in den folgenden Abschnitten hergeleitet. In der Abbildung 10.4 sind die vier Stützstellen ersichtlich. Die Fläche unter dem Graphen lässt sich nun mit der Formel

$$I = \int_{-1}^1 f(x) dx \approx \sum_{i=0}^n A_i f(x_i) \quad (10.5)$$

berechnen, indem man die Stützstellen x_i und die Gewichte A_i einsetzt. Man erhält

$$\begin{aligned} I &\approx 0.347855 \cdot \sqrt{1 - (-0.861136)^2} \\ &+ 0.652145 \cdot \sqrt{1 - (-0.339981)^2} \\ &+ 0.652145 \cdot \sqrt{1 - (0.339981)^2} \\ &+ 0.347855 \cdot \sqrt{1 - (0.861136)^2} \end{aligned}$$

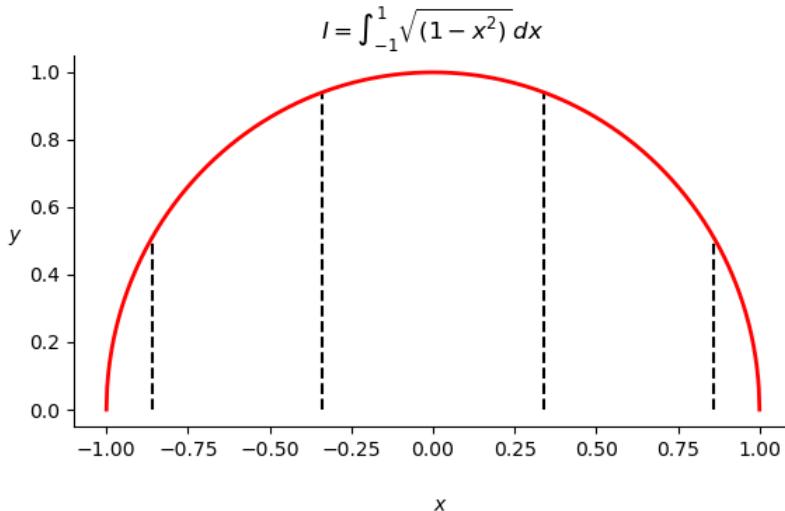


Abbildung 10.4: Position der Stützstellen

$$\begin{aligned}
 &\approx 2 \cdot 0.652145 \cdot \sqrt{1 - (0.339981)^2} + 2 \cdot 0.347855 \cdot \sqrt{1 - (0.861136)^2} \\
 &\approx 1.226596 + 0.386863 \\
 &\approx 1.580278. \tag{10.6}
 \end{aligned}$$

Das Resultat 1.580278 ist eine genauere Annäherung an die tatsächliche Fläche 1.570796 als die Resultate der Trapezformel.

10.3.2 Berechnung der Position der Stützstellen

Um die Position der Stützstellen zu bestimmen, benötigt man zuerst die Anzahl Stützstellen, die für die Quadratur einer Funktion benötigt werden. Wie im Abschnitt 10.2 erwähnt, kann man mit n Stützstellen das Integral eines Polynoms vom Grad $2n - 1$ exakt berechnen. Anders ausgedrückt: Hat man ein Polynom vom Grad g , benötigt man für eine exakte Berechnung des Integrals $\frac{g+1}{2}$ Stützstellen. Hat man eine Funktion, die sich nicht als Polynom darstellen lässt, kann man die Funktion durch ein Polynom annähern, oder die Anzahl benötigter Stützstellen schätzen. Falls man die Anzahl Stützstellen schätzt, ist darauf zu achten, dass eine grössere Anzahl zwar die Genauigkeit der Quadratur erhöht, aber auch der Berechnungsaufwand grösser wird.

Beispiel: Polynom vom Grad 7

Angenommen, man hat die Funktion $f(x) = 5 \cdot x^7 + 2 \cdot x^5 - 8 \cdot x^3 + x + 3$. Der Grad der Funktion lässt sich aus dem höchsten vorkommenden Exponenten von x ableiten, also 7. Für ein Polynom vom Grad 7 werden $\frac{7+1}{2} = 4$ Stützstellen benötigt. In der Abbildung 10.5 ist die Funktion mit den vier Stützstellen ersichtlich.

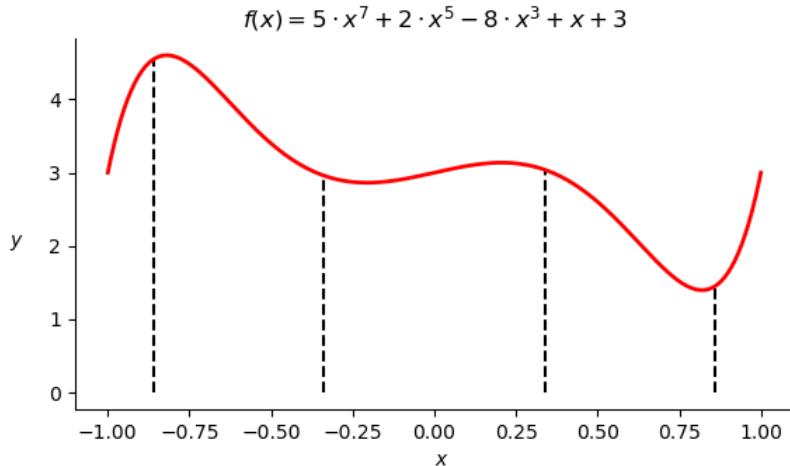


Abbildung 10.5: Funktion des Polynom mit vier Stützstellen

Annäherung von Funktionen durch Polynome

Im Kapitel 3 wurde bereits gezeigt, wie man eine Funktion $f(x)$ durch ein Polynom

$$p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n \quad (10.7)$$

annähern kann. Möchte man nun das Integral $\int_{-1}^1 f(x) dx$ annähern, muss man dazu die Integrale

$$\int_{-1}^1 p(x) dx = a_0 \int_{-1}^1 1 dx + a_1 \int_{-1}^1 x dx + a_2 \int_{-1}^1 x^2 dx + \dots + a_n \int_{-1}^1 x^n dx \quad (10.8)$$

berechnen können. Diese Berechnungen kann durch die allgemeine Formel

$$\int_{-1}^1 x^k dx = \left[\frac{1}{k+1} x^{k+1} \right]_{-1}^1 = \frac{1^{k+1} - (-1)^{k+1}}{k+1} \quad (10.9)$$

dargestellt werden.

Man kann hier erkennen, dass:

$$\int_{-1}^1 x^k dx = \begin{cases} 0 & k \text{ ungerade} \\ \frac{2}{k+1} & k \text{ gerade.} \end{cases} \quad (10.10)$$

Die Erkenntnis, dass ungerade Terme in der Berechnung des Integrals eines Polynoms verschwinden, erklärt, warum für die Quadratur eines Polynoms nur $\frac{n+1}{2}$ Stützstellen benötigt werden. Es müssen nur die geraden Terme evaluiert werden, da:

$$\int_{-1}^1 p(x) dx = a_0 \int_{-1}^1 1 dx + 0 + a_2 \int_{-1}^1 x^2 dx + 0 + \dots + a_n \int_{-1}^1 x^n dx \quad (10.11)$$

$$= \sum_{i=0}^{\frac{n}{2}} a_{2i} \cdot \frac{2}{2i+1}. \quad (10.12)$$

Weiss man, wieviele Stützstellen n man für die Berechnung der Quadratur verwenden möchte, kann man die Position der Stützstellen berechnen. Dafür hat man drei Möglichkeiten, welche nachfolgend erklärt werden.

Substitution der Polynome

Eine Möglichkeit für die Berechnung der Stützstellen (und Gewichte, $w(x)$ ist dabei die Gewichtungsfunktion) ist, in der Formel

$$\int_a^b w(x)P_m(x)dx = \sum_{i=0}^n A_i P_m(x_i), \quad m \leq 2n+1 \quad (10.13)$$

die Polynome P_m durch

$$P_0(x) = 1, \quad P_1(x) = x, \quad \dots, \quad P_{2n+1}(x) = x^{2n+1} \quad (10.14)$$

zu ersetzen und die resultierenden $2n+2$ Gleichungen

$$\int_a^b w(x)x^j dx = \sum_{i=0}^n A_i x_i^j, \quad j = 0, 1, \dots, 2n+1 \quad (10.15)$$

für die Unbekannten x_i und A_i zu lösen.

Nimmt man beispielsweise an, dass $w(x) = x$, $a = -1$, $b = 1$ und $n = 1$, dann ist das resultierende Gleichungssystem:

$$\int_{-1}^1 x dx = A_0 + A_1 \quad (10.16)$$

$$\int_{-1}^1 xx dx = A_0 x_0 + A_1 x_1 \quad (10.17)$$

$$\int_{-1}^1 xx^2 dx = A_0 x_0^2 + A_1 x_1^2 \quad (10.18)$$

$$\int_{-1}^1 xx^3 dx = A_0 x_0^3 + A_1 x_1^3. \quad (10.19)$$

Löst man die Integrale, erhält man

$$A_0 + A_1 = 0 \quad (10.20)$$

$$A_0 x_0 + A_1 x_1 = \frac{2}{3} \approx 0.66667 \quad (10.21)$$

$$A_0 x_0^2 + A_1 x_1^2 = 0 \quad (10.22)$$

$$A_0 x_0^3 + A_1 x_1^3 = \frac{2}{5} = 0.4. \quad (10.23)$$

Daraus ergibt sich die Lösung

$$A_0 = -\frac{\sqrt{\frac{5}{3}}}{3} \quad x_0 = -\sqrt{\frac{3}{5}} \quad (10.24)$$

n	Stützstellen x_i für n
0	0.00000
1	± 0.57735
2	± 0.77460 0.00000
3	± 0.86114 ± 0.33998
4	± 0.90618 ± 0.53847 0.00000
5	± 0.93247 ± 0.66121 ± 0.23862

Tabelle 10.2: Werte für Stützstellen x_i der Gauss-Quadratur für $n \leq 6$

$$A_1 = -\frac{\sqrt{\frac{5}{3}}}{3} \quad x_1 = -\sqrt{\frac{3}{5}}. \quad (10.25)$$

Setzt man diese Werte in die Integrationsformel ein, so wird die Formel

$$\int_{-1}^1 x \cdot f(x) dx \approx \frac{1}{3} \left[\left(\sqrt{\frac{5}{3}} \right) f \left(-\sqrt{\frac{3}{5}} \right) + \left(-\sqrt{\frac{5}{3}} \right) f \left(\sqrt{\frac{3}{5}} \right) \right] \quad (10.26)$$

Man kann sehen, dass diese Methode für die Berechnung der Stützstellen für grosse n wegen der Nichtlinearität der x_i sehr schnell viel Zeit beansprucht. Die anderen Methoden für die Bestimmung der Stützstellen und Gewichte, welche nachfolgend erklärt werden, sind effektiver und in der Anwendung einfacher.

Werte in der Tabelle nachschlagen

Für die gängigsten Formen der Gauss-Quadratur existieren Tabellen mit vorberechneten Werten für die Position der Stützstellen und deren Gewichte. Diese Fälle setzen voraus, dass die Gewichtungsfunktion $w(x) = 1$ ist. Die Tabellen sind sehr präzise und für die ersten hundert n vorgerechnet. Außerdem gibt es Programme, welche für beliebige n die dazugehörigen Stützstellen und Gewichte berechnen. In der Tabelle 10.2 sind die Werte für $n \leq 5$ beschrieben.

Berechnen durch Formel

Die Position der Stützstellen lässt sich mit der Formel

$$\int_{-1}^1 P(x) \cdot x^j dx = 0$$

mit

$$j = 0, 1, 2, \dots, n-1$$

und

$$P(x) = (x - x_1)(x - x_2) \dots (x - x_n) \quad (10.27)$$

berechnen. Um die Formel zu verstehen, werden einige Grundkenntnisse über orthogonale Polynome und deren Beziehung zur Gauss-Quadratur vorausgesetzt. Da aber für die häufigsten Formen der Gauss-Quadratur (Siehe Abschnitt 10.3.5) bereits vorberechnete Tabellen für die Stützstellen und Gewichte existieren, muss man die Theorie für die Anwendung der Formeln nicht verstehen. Für interessierte Leser sind die wichtigsten Eigenschaften von orthogonalen Polynomen im nächsten Abschnitt genauer erläutert.

Beispiel: Berechnung für zwei Stützstellen

Möchte man die Position für zwei Stützstellen berechnen, sieht die Formel (10.27) folgendermassen aus:

$$P(x) = (x - x_1)(x - x_2) \quad (10.28)$$

und man erhält für $j = 0, 1$ die beiden Gleichungen

$$j = 0 : \int_{-1}^1 P(x) dx = \int_{-1}^1 (x - x_1)(x - x_2) dx = 0 \quad (10.29)$$

$$j = 1 : \int_{-1}^1 P(x)x^1 dx = \int_{-1}^1 (x - x_1)(x - x_2)x dx = 0 \quad (10.30)$$

Löst man das Integral, erhält man:

$$j = 0 : \frac{1}{3} + x_1 x_2 = 0 \quad (10.31)$$

$$j = 1 : \frac{2}{3}(x_1 + x_2) = 0. \quad (10.32)$$

Umgeformt erhält man:

$$x_1 x_2 = -\frac{1}{3} \quad \text{und} \quad x_1 + x_2 = 0. \quad (10.33)$$

Somit sind x_1 und x_2 :

$$\begin{aligned} x_1 &= -\frac{1}{\sqrt{3}} \approx -0.55735 \\ x_2 &= \frac{1}{\sqrt{3}} \approx 0.55735. \end{aligned} \quad (10.34)$$

10.3.3 Orthogonale Polynome

Dieser Abschnitt befasst sich mit den für die Gauss-Quadratur wichtigsten Eigenschaften von orthogonalen Polynomen. Aus der linearen Algebra kennt man orthogonale Vektoren als zwei Vektoren, deren Skalarprodukt im Raum \mathbb{R}^2 null ist, also wenn für die Vektoren \vec{v} und \vec{w} gilt:

$$\langle \vec{v}, \vec{w} \rangle = 0 \quad (10.35)$$

Ebenso kann man Funktionenräume als Vektorräume beschreiben und somit die Orthogonalität von zwei Funktionen f und g mit

$$\langle f, g \rangle = 0 \quad (10.36)$$

beschreiben. Auch für Polynome gibt es solche Orthogonalitätseigenschaften, welche nachfolgend erklärt werden. Da das gesamte Thema Orthogonalität aber den Rahmen dieses Kapitels sprengen würde, werden nur einige für die Gauss-Quadratur relevante Eigenschaften angesprochen. Eine gute Übersicht über Orthogonalität findet sich auf [3]. Weitere Informationen zu orthogonalen Polynomen kann man auf [1] oder [2] finden.

Name	Symbol	a	b	$w(x)$
Legendre	$p_{n(x)}$	-1	1	1
Chebyshev	$T_n(x)$	-1	1	$(1 - x^2)^{-1/2}$
Laguerre	$L_n(x)$	0	∞	e^{-x}
Hermite	$H_n(x)$	$-\infty$	∞	e^{-x^2}

Tabelle 10.3: Klassische Orthogonalpolynome

Name	$\phi_0(x)$	$\phi_1(x)$	a_n	b_n	c_n	d_n
Legendre	1	x	$n + 1$	0	$2n + 1$	n
Chebyshev	1	x	1	0	2	1
Laguerre	1	$1 - x$	$n + 1$	$2n + 1$	-1	n
Hermite	1	$2x$	1	0	2	2

Tabelle 10.4: Koeffizienten für die berechnung von Orthogonalpolynomen

Grundlagen orthogonale Polynome

Orthogonale Polynome werden in der mathematischen und numerischen Analysis oft verwendet. In der numerischen Integration entsprechen die Nullstellen der orthogonalen Polynome den Stützstellen der Gauss-Quadratur. Zwei Polynome $\phi_m(x)$ und $\phi_n(x)$ (n und m ist hierbei der Grad des Polynoms) formen ein orthogonales Paar im Intervall (a, b) im Bezug auf die Gewichtungsfunktion $w(x)$ falls:

$$\int_a^b w(x)\phi_m(x)\phi_n(x) dx = 0, \quad m \neq n \quad (10.37)$$

Orthogonalpolynome sind abgesehen von einem konstanten Faktor, der durch die Normierung der Polynome ermittelt wird, über die Wahl der Gewichtungsfunktion und der Grenzen des Integrals a, b eindeutig festgelegt. Klassische Orthogonalpolynome sind nach ihren Entdeckern benannt und in der Tabelle 10.3 mit ihren Eigenschaften beschrieben. Weiterhin haben orthogonale Polynome rekursive Beziehungen zueinander, das heisst, dass mit der Formel

$$a_n\phi_{n+1}(x) = (b_n + c_n x)\phi_n(x) - d_n\phi_{n-1}(x) \quad (10.38)$$

mit den ersten beiden Orthogonalpolynomen weitere Polynome berechnet werden können. Die Koeffizienten für die Rekursionsformel finden sich in der Tabelle 10.4

Für die Gauss-Quadratur relevante Eigenschaften

Ein orthogonales Polynom $\phi_n(x)$ vom Grad n hat n reelle, unterschiedliche Nullstellen im Intervall (a, b) . Diese Eigenschaft ist für die Gauss-Quadratur relevant, da für eine Gauss-Quadraturformel vom Grad n für die exakte Berechnung des Integrals diejenigen Abszissenwerte x_i benötigt werden, welche den Nullstellen des n -ten orthogonalen Polynoms P_n entsprechen. Weiterhin liegen die Nullstellen des Polynoms $\phi_n(x)$ zwischen den Nullstellen des Polynoms $\phi_{n+1}(x)$. Diese Eigenschaft erlaubt, den Grad n des Polynoms zu erhöhen und man erhält dadurch mehr Nullstellen, welche den Stützstellen x_i der Gauss-Quadraturformel entsprechen.

Ein beliebiges Polynom $P_n(x)$ kann in der Form $P_n(x) = \sum_{i=0}^n c_i\phi_i(x)$ ausgedrückt werden. Anders ausgedrückt, lässt sich durch diese Eigenschaft ein beliebiges Polynom als Linearkombination

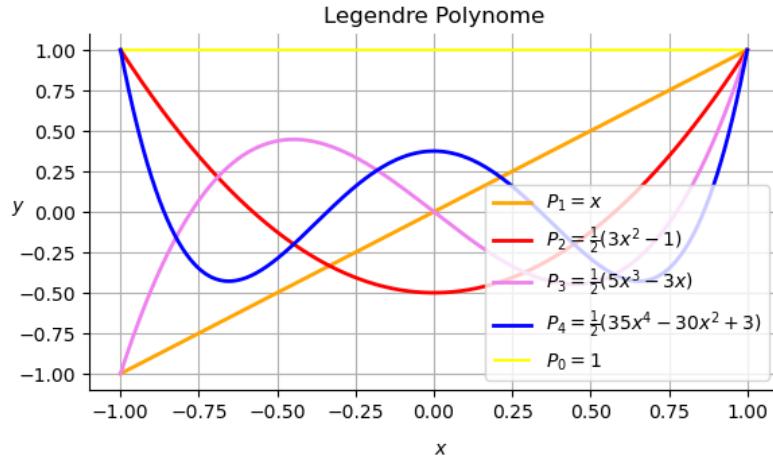


Abbildung 10.6: Die Ersten vier Legendre-Polynome

von Orthogonalpolynomen darstellen. Somit lässt sich das Integral für beliebige Polynome mit der Gauss-Quadratur exakt berechnen.

In der Abbildung 10.6 sind die ersten vier Legendre-Polynome abgebildet. Man kann darin sehr gut sehen, dass die Positionen der Nullstellen den in der Tabelle 10.2 genannten Stützstellen x_i entsprechen und dass die Nullstellen eines Polynoms P_n zwischen denen eines Polynoms P_{n+1} liegen.

10.3.4 Berechnung der Gewichte an den Stützstellen

Auch für die Bestimmung der Gewichte an den berechneten Stützstellen kann man entweder in einer vorberechneten Tabelle die Werte nachschlagen, oder die Gewichte selber berechnen.

Tabelle für die Gewichte

In der Tabelle 10.5 sind für $n \leq 5$ die Gewichte A_i an den Stützstellen x_i angegeben.

Berechnen der Gewichte

Die Gewichte an den Stützstellen lässt sich mit der Formel

$$A_j = \int_{-1}^1 l_j(x) dx, \quad j = 1, 2, \dots, n$$

und

$$l_j(x) := \prod_{0 \leq m \leq k, m \neq j} \frac{x - x_m}{x_j - x_m} \quad (10.39)$$

berechnen.

Die Formel lässt sich mit der im Kapitel 3 genannten Interpolationspolynom herleiten. Für die Funktion $f(x)$ mit Stützstellen x_i ist das Interpolationspolynom

$$p(x) = \sum_{i=0}^n f(x_i) l_i(x) \quad (10.40)$$

n	Stützstellen x_i für n	Gewichte A_i
0	0.00000	2.00000
1	± 0.57735	1.00000
2	± 0.77460 0.00000	0.55556 0.88889
3	± 0.86114 ± 0.33998	0.34785 0.65215
4	± 0.90618 ± 0.53847 0.00000	0.23693 0.47863 0.56889
5	± 0.93247 ± 0.66121 ± 0.23862	0.17132 0.36076 0.46791

Tabelle 10.5: Werte für Stützstellen x_i und Gewichte A_i der Gauss-Quadratur für $n \leq 6$

und das Integral ist

$$\int_{-1}^1 f(x) dx \approx \int_{-1}^1 p(x) dx \quad (10.41)$$

$$= \int_{-1}^1 \sum_{i=0}^n f(x_i) l_i(x) dx \quad (10.42)$$

$$= \sum_{i=0}^n f(x_i) \cdot \int_{-1}^1 l_i(x) dx \quad (10.43)$$

$$= \sum_{i=0}^n A_i f(x_i), \quad (10.44)$$

wobei $\int_{-1}^1 l_i(x) dx = A_i$.

Beispiel: Berechnung der Gewichte für zwei Stützstellen

In der Beispielberechnung der Stützstellen im Abschnitt 10.3.2 sieht man, dass sich die Stützstellen an den Punkten $x_1 = -\frac{1}{\sqrt{3}}$ und $x_2 = \frac{1}{\sqrt{3}}$ befinden.

Die Formeln für die Berechnung der Gewichte A_1 und A_2 sind:

$$A_1 = \int_{-1}^1 \frac{x - x_2}{x_1 - x_2} dx$$

und

$$A_2 = \int_{-1}^1 \frac{x - x_1}{x_2 - x_1} dx. \quad (10.45)$$

setzt man die Werte für x_1 und x_2 in die Formeln ein, erhält man:

$$A_1 = \int_{-1}^1 \frac{x - \frac{1}{\sqrt{3}}}{-\frac{1}{\sqrt{3}} - \frac{1}{\sqrt{3}}} dx$$

Name	Untere Grenze	Obere Grenze	Formel
Legendre	-1	1	$p_n(x) = \int_{-1}^1 f(x) dx \approx \sum_{i=0}^n A_i f(x_i)$
Chebyshev	-1	1	$T_n(x) = \int_{-1}^1 (1-x^2)^{-1/2} f(x) dy \approx \frac{\pi}{n+1} \sum_{i=0}^n f(x_i)$
Laguerre	0	∞	$L_n(x) = \int_0^\infty e^{-x} f(x) dx \approx \sum_{i=0}^n A_i f(x_i)$
Hermite	$-\infty$	∞	$H_n(x) = \int_{-\infty}^\infty f(x) dx \approx \sum_{i=0}^n A_i f(x_i)$

Tabelle 10.6: Formen der Gauss-Quadratur

$$A_2 = \int_{-1}^1 \frac{x - \frac{-1}{\sqrt{3}}}{\frac{1}{\sqrt{3}} - \frac{-1}{\sqrt{3}}} dx. \quad (10.46)$$

Vereinfacht man die Terme, erhält man:

$$A_1 = \int_{-1}^1 -\frac{1}{2} \cdot \sqrt{3} \cdot \left(x - \frac{1}{\sqrt{3}} \right) dx = -\frac{\sqrt{3}}{2} \left(\cdot \int_{-1}^1 x dx - \frac{1}{\sqrt{3}} \cdot \int_{-1}^1 1 dx \right) \quad (10.47)$$

$$A_2 = \int_{-1}^1 \frac{1}{2} \cdot \sqrt{3} \cdot \left(x + \frac{1}{\sqrt{3}} \right) dx = \frac{\sqrt{3}}{2} \left(\cdot \int_{-1}^1 x dx + \frac{1}{\sqrt{3}} \cdot \int_{-1}^1 1 dx \right). \quad (10.48)$$

Es ist $\int_{-1}^1 x dx = 0$ und $\int_{-1}^1 1 dx = 2$, somit:

$$A_1 = -\frac{\sqrt{3}}{2} \cdot \left(0 - \frac{1}{\sqrt{3}} \cdot 2 \right) = 1 \quad (10.49)$$

$$A_2 = \frac{\sqrt{3}}{2} \cdot \left(0 + \frac{1}{\sqrt{3}} \cdot 2 \right) = 1. \quad (10.50)$$

Es folgt somit, dass

$$I \approx f(x_1) + f(x_2) = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right). \quad (10.51)$$

10.3.5 Formen der Gauss-Quadratur

Wie im Abschnitt 10.3.2, Orthogonale Polynome, bereits angedeutet, gibt es verschiedene Ausprägungen der Gauss-Integration. Für verschiedene Folgen von Orthogonalpolynomen gibt es eine dazugehörige Form der Gauss-Quadratur mit eigenen Grenzwerten und Berechnungsformeln. Die vier häufigsten Formen sind in der Tabelle 10.6 zusammengestellt. In den bisherigen Beispielen wurde die Gauss-Formel mit der Legendre-Form, auch Gauss-Legendre-Formel genannt, verwendet. Interessant ist bei der Gauss-Chebyshev-Formel, dass für die Berechnung des Integrals keine Gewichte berechnet werden müssen, sondern dass für jedes x_i das selbe Gewicht $\frac{\pi}{n+1}$ verwendet werden kann. Außerdem lassen sich die Stützstellen x_i mit der Formel

$$x_i = \cos \frac{(2i+1)\pi}{2n+2} \quad (10.52)$$

berechnen und sind symmetrisch um $x = 0$ verteilt. Die Gauss-Laguerre-Formel verwendet für die Bestimmung der Stützstellen die Laguerre-Polynome. Diese Stützstellen liegen alle auf dem Intervall $[0, \infty]$

Möchte man nun ein beliebiges Integral mittels der Gauss-Quadratur berechnen, deren Grenzwerte nicht zu den in der Tabelle 10.6 dargestellten Formeln passt, abgesehen von weiteren hier nicht genannten Formen, muss man die Grenzwerte (a, b) gemäss dem folgenden Beispiel auf den Bereich $(-1, 1)$ abbilden.

Vorgehen bei Integralgrenzen, die nicht durch die Tabelle abgedeckt werden

Angenommen man hat ein Integral

$$\int_a^b f(\xi) d\xi \quad (10.53)$$

zu berechnen. Mit der Transformation

$$\xi = \frac{b+a}{2} + \frac{b-a}{2}x \quad (10.54)$$

wird $d\xi = dx(b-a)/2$ und die Quadraturformel wird

$$\int_a^b f(\xi) d\xi = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right) dx \approx \frac{b-a}{2} \sum_{i=1}^n A_i f\left(\frac{b-a}{2}x_i + \frac{b+a}{2}\right). \quad (10.55)$$

10.3.6 Fehler der Gauss-Quadratur

Wie im Abschnitt 10.3.2 erklärt, lassen sich Polynome mit der Gauss-Quadratur exakt berechnen. Der Fehler für die Annäherung eines Integrals mittels eines Polynoms liegt also in der Wahl des Annäherungspolynoms und ein Fehler in der Berechnung des Integrals. Diesen Fehler können wir mit den im Abschnitt 3.2.2 hergeleiteten Methoden ermitteln. Weiterhin gibt es in der Gauss-Quadratur eigene Methoden, um den Fehler der Quadratur zu ermitteln. Die einfachere Methode ist, für eine Quadratur mit n Stützstellen eine weitere Quadratur mit $n+1$ Stützstellen durchzuführen und die absolute Differenz zwischen beiden Resultaten zu berechnen. Die komplexere Methode ist, die Fehlerformel für die Gauss-Quadraturformel anzuwenden.

Fehlerrechnung mittels Quadratur höherer Ordnung

In der Praxis ist diese Methode für die Berechnung des Fehlers der Gauss-Quadratur simpler und schneller in der Anwendung, da die Fehlerformel der Gauss-Quadratur von der jeweiligen Methode der Gauss-Quadratur (Legendre, Laguerre, etc.) abhängt. Nachfolgend wird für die Formeln je nach Anzahl n der Stützstellen für die Position der Stützstellen und Gewichte die Notation x_i^n und A_i^n verwendet. Für eine Gauss-Legendre-Quadratur

$$I = \int_{-1}^1 f(x) dx \approx \sum_{i=1}^n A_i^n f(x_i) \quad (10.56)$$

kann der Fehler mit der Formel

$$E \approx \sum_{i=1}^{n+1} A_i^{n+1} f(x_i) - \sum_{k=1}^n A_k^n f(x_k) \quad (10.57)$$

approximiert werden. Für $n = 4$ wird die Fehlerrechnung

$$E \approx \sum_{i=1}^5 A_i^5 f(x_i) - \sum_{k=1}^4 A_k^4 f(x_k) \quad (10.58)$$

$$= A_1^5 f(x_1^5) + A_2^5 f(x_2^5) + A_3^5 f(x_3^5) + A_4^5 f(x_4^5) + A_5^5 f(x_5^5) \quad (10.59)$$

$$- A_1^4 f(x_1^4) + A_2^4 f(x_2^4) + A_3^4 f(x_3^4) + A_4^4 f(x_4^4) \quad (10.60)$$

Fehlerrechnung mit der Fehlerformel

Die Fehlerformel

$$E = \int_a^b w(x)f(x) dx - \sum_{i=1}^n A_i f(x_i) \quad (10.61)$$

hat die allgemeine Form

$$E = K(n) f^{(2n+2)}(c). \quad (10.62)$$

wobei $a < c < b$ und $K(n)$ von der jeweiligen Form der Quadratur abhängt.

Die Funktion $f(x)$ muss dabei $2n + 2$ mal abgeleitet werden. Falls die Ableitung der Funktion $f(x)$ vor der $2n + 2$ -ten Ableitung 0 ergibt, lässt sich das Integral mit der Gauss-Quadratur exakt berechnen. Für Polynome vom Grad $2n + 1$ ist dies der Fall und im Abschnitt 10.3.2 wurde bereits erklärt, dass diese Polynome mit n Stützstellen exakt berechnen werden können.

Für die Gauss-Legendre-Quadraturformel ist $K(n)$:

$$E = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n+2)}(c) \quad (10.63)$$

in der allgemeinen Form oder für $a = -1, b = 1$:

$$E = \frac{2^{2n+1}[(n!)^4]}{(2n+1)[(2n)!]^3} f^{(2n+2)}(c). \quad (10.64)$$

Die allgemeine Form wird dann benötigt, falls ein Integral der Form

$$I = \int_a^b f(\xi) d\xi \quad (10.65)$$

in die Form

$$I = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right) dx \quad (10.66)$$

gebracht wird.

Die Variable c ist dabei keine vorgegebene Zahl, sondern muss im Intervall (a, b) gefunden werden und ist von der jeweiligen Funktion $f(x)$ abhängig. Die Tabelle 10.7 zeigt für die verschiedenen Formen der Gauss-Quadratur die Fehlerformeln und deren Grenzen für c . Um c zu finden, muss die Funktion $f(\xi)$ $2n+2$ mal abgeleitet werden und man berechnet dasjenige c im Intervall $[a, b]$ welches den höchsten Wert für $f^{(2n+2)}(c)$ ergibt.

Name	Fehlerformel	Intervall für c
Legendre	$E = \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n+2)}(c)$	$-1 < c < 1$
Chebyshev	$E = \frac{2\pi}{2^{2n+2}(2n+2!)} f^{(2n+2)}(c)$	$-1 < c < 1$
Laguerre	$E = \frac{[(n+1)!]^2}{(2n+2)!} f^{(2n+2)}(c)$	$0 < c < \infty$
Hermite	$E = \frac{\sqrt{\pi}(n+1)!}{2^2(2n+2)!} f^{(2n+2)}(c)$	$0 < c < \infty$

Tabelle 10.7: Fehlerformeln der Gauss-Quadratur

Beispiel: Fehlerrechnung für $I = \int_2^4 x^{-1} dx$ mit zwei Stützstellen

Zuerst wird das Intervall [2,4] mit der linearen Transformation in das Intervall [-1,1] überführt:

$$\int_a^b f(\xi) d\xi = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right) dx \quad (10.67)$$

Für die Zwei-Punkt-Gauss-Quadratur sind die Stützstellen $x_0 = -\frac{1}{\sqrt{3}}$, $x_1 = \frac{1}{\sqrt{3}}$ und die Gewichte $A_0 = A_1 = 1$. Mit $f(x) := x^{-1}$ erhalten wir die Formel:

$$I = \frac{4-2}{2} \int_{-1}^1 f\left(\frac{4-2}{2}x + \frac{4+2}{2}\right) dx \quad (10.68)$$

$$= \int_{-1}^1 f(x+3) dx \quad (10.69)$$

$$\approx \sum_{i=0}^1 A_i f(x_i + 3) \quad (10.70)$$

$$= f\left(-\frac{1}{\sqrt{3}} + 3\right) - f\left(\frac{1}{\sqrt{3}} + 3\right) \quad (10.71)$$

$$= \left(-\frac{1}{\sqrt{3}} + 3\right)^{-1} - \left(\frac{1}{\sqrt{3}} + 3\right)^{-1} \quad (10.72)$$

$$\approx 0.41277118 \dots + 0.27953651 \dots \quad (10.73)$$

$$\approx 0.69230769 \dots \quad (10.74)$$

Um den Fehler zu berechnen, verwenden wir nun die Fehlerformel 10.63 bei der wir die Werte $a = 2$, $b = 4$ und $n = 2$ einsetzen

$$E = \frac{(4-2)^{4+1}[(2)!]^4}{(4+1)[(4)!]^3} f^{(6)}(c) \quad (10.75)$$

$$= \frac{(2)^5[2]^4}{(5)[24]^3} f^{(6)}(c) \quad (10.76)$$

$$= \frac{2^5 \cdot 2^4}{5 \cdot 24^3} f^{(6)}(c) \quad (10.77)$$

n	Gauss-Quadratur Resultat	Gauss-Quadratur Fehler	Trapezregel Resultat	Trapezregel Fehler
2	1.6329931619	0.3670068381	0.0000000000	1.5707963268
3	1.5916172578	0.0413759040	1.0000000000	0.5707963268
4	1.5802775277	0.0113397301	1.2570787221	0.3137176047
5	1.5759063349	0.0043711928	1.3660254038	0.2047709230
7	1.5727819554	0.0010820282	1.4587766894	0.1120196374
10	1.5715139556	0.0002531675	1.5096159164	0.0611804104
20	1.5708921460	$1.55366 \cdot 10^{-5}$	1.5507798233	0.020016503
30	1.5708253858	$3.05870 \cdot 10^{-6}$	1.5601695280	0.010626799
40	1.5708087326	$9.66660 \cdot 10^{-7}$	1.5639786384	0.006817688
50	1.5708027245	$3.95730 \cdot 10^{-7}$	1.5659537235	0.004842603
100	1.5707971383	$2.47153 \cdot 10^{-8}$	1.5691090196	0.001687307

Tabelle 10.8: Resultatvergleich zwischen der Gauss-Quadratur und der Trapezregel

$$= \frac{2^9}{5 \cdot 13824} f^{(6)}(c) \quad (10.78)$$

$$= \frac{512}{69120} f^{(6)}(c). \quad (10.79)$$

Möchte man nun c für $f^{(6)}(c)$ herausfinden, benötigt man die sechste Ableitung von $f(x)$ und berechnet dasjenige c im Intervall $[2, 4]$ welches den höchsten Wert für $f^{(6)}(c)$ ergibt.

Für $f(x) = 1/x$ ist die sechste Ableitung $f^{(6)}(x) = 720/x^7$. Der Wert von $f^7(c)$ wird somit grösser, je kleiner c ist. Im Intervall $[2, 4]$ ist 2 die kleinste Zahl und wird in die Fehlerformel eingesetzt:

$$E \leq \frac{512}{69120} f^7(c) = \frac{512}{69120} \cdot \frac{720}{2^7} = \frac{512 \cdot 720}{69120 \cdot 128} = \frac{368640}{8847360} \approx 0.041667 \dots \quad (10.80)$$

10.4 Folgerungen

10.4.1 Vergleich Gauss-Quadratur mit Trapezregel

In diesem Abschnitt werden die Resultate der Gauss-Quadratur mit Resultaten der Trapezregel verglichen, um zu zeigen wie effizient die Integration mit der Gaußschen-Integration ist. Dazu wird nochmals die Funktion $\sqrt{1-x^2}$ integriert. Die Resultate sind in der Tabelle 10.8 dargestellt. Als Referenz: Das analytische Resultat des Integrals ist $\frac{\pi}{2} \approx 1.5707963268$. Man kann sehr schön sehen, dass die Resultate der Gauss-Quadratur mit nur wenigen Funktionsauswertungen schon sehr geringe Abweichungen vom analytischen Resultat haben, während bei der Trapezregel viel mehr Funktionsauswertungen benötigt werden, um die selbe Genauigkeit zu erreichen. Man sieht, dass die Genauigkeit der Trapezregel mit $n = 100$ bei der Gauss-Quadratur bereits mit $n = 7$ erreicht wird.

10.4.2 Programm für Gauss-Quadratur in Python

In Python lässt sich die Gauss-Quadratur in gerade mal 25 Zeilen Code umsetzen. Dafür wird aus dem Modul `scipy` die Klasse `integrate` importiert. Außerdem wird für die mathematischen Funktionen das Modul `numpy` verwendet

```

1 import numpy as np
2 from scipy import integrate
3
4 # Warnungen werden erzeugt, wenn mit der gewählten Anzahl
5 # Stützstellen die Toleranz tol=1.49e-08 nicht erreicht werden
6 # kann
7
8 # Zu integrierende Funktion
9 func = lambda x: np.sqrt(1 - x ** 2)
10
11 # Anzahl Stützstellen in einem Array.
12 nodes = [1, 2, 3, 4, 5, 7, 10, 20, 30, 40, 50, 100]
13
14 # Quadraturen mit node = Anzahl Stützstellen
15 for node in nodes:
16     x = np.linspace(-1, 1, node)
17     y = func(x)
18     print(f"Number of nodes: {node}")
19     # Berechnung mit Gauss-Quadratur
20     print(f"Gauss-Legendre : {integrate.quadrature(func, -1, 1,
21                         maxiter=node)}")
22     # Berechnung mit Trapezformel
23     trapez = integrate.trapz(y, x)
24     print(f"Trapezoidal Rule: {trapez}")
25     # Berechnung mit Simpsonscher Regel
26     print(f"Simpson's Rule : {integrate.simps(y, x)}")
27     print()

```

10.4.3 Fazit

Die Gauss-Quadratur ist ein numerisches Integrationsverfahren, welches mit wenigen Funktionsauswertungen das bestimmte Integral von beliebigen Funktionen sehr genau berechnen kann. Polynome oder Annäherungen von Funktionen mittels Polynomen lassen sich sogar exakt berechnen und benötigen für ein Polynom vom Grad $2n + 1$ nur n Funktionsauswertungen.

Es gibt verschiedene Wege für die Bestimmung der Stützstellen und Gewichte. Vorgerechnete Werte für x_i und A_i sind für grosse n in Tabellen vorhanden und erlauben schnelle Lookups. Für grössere n existieren Formeln, die die Berechnung der Stützstellen und Gewichte erlauben und in Computerprogrammen einfach umgesetzt werden können. Für verschiedene Integralgrenzen gibt es eigene Formen der Gauss-Quadratur die auf orthogonalen Polynomen aufbauen, deren Nullstellen die Stützstellen bestimmen.

Literatur

- [1] *Orthogonale Polynome*. URL: <https://mathworld.wolfram.com/OrthogonalPolynomials.html>.
- [2] *Orthogonale Polynome*. 8. Aug. 2019. URL: https://de.wikipedia.org/wiki/Orthogonale_Polynome.
- [3] *Orthogonalität*. 9. Juni 2020. URL: <https://de.wikipedia.org/wiki/Orthogonalit%C3%A4t>.

11

Van der Pol-Differentialgleichung

Manuel Cattaneo und Niccolò Galliani

11.1 Einleitung

In diesem Kapitel wird die Empfindlichkeit einiger numerischer Methoden der Runge-Kutta-Familie auf die Länge der Integrationsschritte erläutert. Chaotische Systeme reagieren sehr empfindlich auf die Wahl der Anfangsbedingungen. Eine kleine Änderung z. B. führt zu einer völlig anderen Antwort. Die Ausbreitung des numerischen Fehlers kann den gleichen Effekt haben und für verschiedene Werte völlig unterschiedliche Lösungen ergeben. In unserem Fall ist das untersuchte chaotische System der Van der Pol Oszillator. Die betreffende Differentialgleichung wurde im Jahr 1927 von Balthasar Van der Pol während seiner Studien über Vakuumröhren Oszillatoren formuliert [3]. Sie wird wie folgt beschrieben:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2) \frac{dx}{dt} + x = 0. \quad (11.1)$$

Die numerischen Methoden, die in diesem Kapitel verwendet werden, sind der Runge-Kutta-Algorithmus vierter Ordnung (Abschnitt 5.4.2) und der Euler-Algorithmus, der als Runge-Kutta-Algorithmus erster Ordnung betrachtet werden kann (Satz 5.4).

11.2 Die Van der Pol Gleichung

Die Van der Pol Gleichung (11.1) ist eine *nichtlineare* Differentialgleichung zweiter Ordnung. Die Nichtlinearität wird durch die Tatsache festgelegt, dass der Koeffizient von \dot{x} nicht linear ist. Tatsächlich sind x^2 und $x \cdot \dot{x}$ zwei nichtlineare Operationen.

11.2.1 Van der Pol Gleichung als Oszillator mit nichtlinearem Widerstand

In der Abbildung 11.1 steht ein klassischer RLC-Serienschwingkreis. In diesem Fall ist aber der Widerstandswert nicht linear. Dies bedeutet, dass die Beziehung zwischen Spannung und Strom

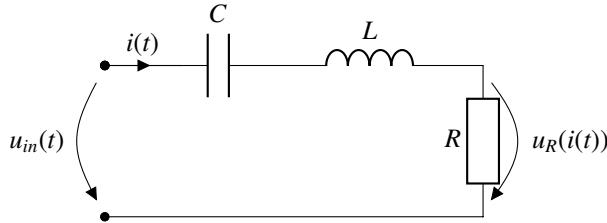


Abbildung 11.1: RLC-Schaltung mit nichtlinearem Widerstand

nicht mehr vom Typ $u_R(i) = a \cdot i(t)$ lautet. Es kann jedoch als Polynom dargestellt werden, und zwar in unserem Fall dritter Ordnung anstatt des ersten, wie hier gezeigt:

$$u_R(i) = a \cdot i^3 + b \cdot i^2 + c \cdot i + d. \quad (11.2)$$

Eine solche Kennlinie könnte man zum Beispiel mit einer Tunneldiode, einer Anordnung von Feldeffekttransistoren oder Operationsverstärkern erhalten. Die Konstanten könnten auf diese Weise zugeordnet werden:

$$a = \frac{R_0}{3 \cdot i_0^2}, \quad c = -R_0, \quad b = d = 0,$$

somit:

$$\begin{aligned} u_R(i) &= \frac{R_0}{3 \cdot i_0^2} \cdot i^3 - R_0 \cdot i \\ &= -R_0 \cdot i_0 \left(\frac{i}{i_0} - \frac{1}{3} \left(\frac{i}{i_0} \right)^3 \right). \end{aligned}$$

Für die Spannungen $u_L(t)$ und $u_C(t)$, von Spule bzw. Kondensator, sind aus der Elektrotechnik folgende Zusammenhänge bekannt:

$$u_C(t) = \frac{1}{C} \int_0^t i(\tau) d\tau, \quad u_L(t) = L \frac{di}{dt}.$$

Die Maschengleichung wird nun nach dem zweiten Kirchhoffschen Gesetz:

$$u_{in}(t) = u_R(t) + u_C(t) + u_L(t). \quad (11.3)$$

Unter der Annahme, dass die Spannung u_{in} eine über die Zeit unveränderliche Grösse ist, d.h. $u_{in}(t) = U_0$, werden die beiden Seiten der Gleichung (11.3) nach t abgeleitet:

$$\frac{dU_0}{dt} = \frac{d}{dt} (u_R(t) + u_C(t) + u_L(t)) \quad (11.4)$$

$$0 = -R_0 \cdot i_0 \left(\frac{1}{i_0} \frac{di}{dt} - \frac{1}{3 \cdot i_0^3} \frac{d}{dt} (i(t)^3) \right) + \frac{1}{C} i(t) + L \frac{d^2 i}{dt^2}$$

$$0 = -R_0 \cdot i_0 \left(\frac{1}{i_0} \frac{di}{dt} - \frac{1}{3 \cdot i_0^3} \cdot \left(3 \frac{di}{dt} i(t)^2 \right) \right) + \frac{1}{C} i(t) + L \frac{d^2 i}{dt^2}$$

$$0 = \frac{d^2 i}{dt^2} - \frac{R_0}{L} \frac{di}{dt} \cdot \left(1 - \left(\frac{i(t)}{i_0} \right)^2 \right) + \frac{1}{LC} i(t). \quad (11.5)$$

Die verschiedenen Konstanten werden wie folgt definiert:

$$\frac{R_0}{L} = \mu, \quad C = \frac{\mu}{R_0}, \quad i_0 = 1,$$

so folgt die Gleichung:

$$\frac{d^2i}{dt^2} - \mu(1 - i^2) \frac{di}{dt} + i = 0. \quad (11.6)$$

Es ist daher gezeigt, welche Art von System durch diese Differentialgleichung beschrieben werden könnte.

Offensichtlich gibt es viele Möglichkeiten, diese Art von Nichtlinearität für den Spannungswert $u_R(t)$ zu erreichen, einige sind oben aufgelistet. Wenn versucht wird, die gleiche Gleichung zu lösen, aber mit einem konstanten R Wert, erhalten wir eine Differentialgleichung des Typs

$$\frac{d^2i}{dt^2} + \frac{R}{L} \frac{di}{dt} + \frac{1}{LC} i = 0. \quad (11.7)$$

Sie kann umgeschrieben werden als

$$\frac{d^2i}{dt^2} + 2\zeta \frac{di}{dt} + \omega_0^2 i = 0, \quad (11.8)$$

wobei ω_0 die Resonanzfrequenz und ζ die Dämpfungskonstante sind. Die analytische Lösung ist

$$i(t) = e^{-\zeta t} \cos(\omega_d t + \varphi_0). \quad (11.9)$$

Das bedeutet, dass im Falle der Gleichung (11.6), d.h. die Van der Pol Gleichung eine Schwingung vorliegt, die bei grossen Werten von i^2 einen positiven Dämpfungswert hat, der ihre Amplitude verringert. Bei kleinen Werten von i^2 ist der Dämpfungswert negativ, was dazu führt, dass die Amplitude grösser wird und somit in diesen beiden Zuständen schwingt.

11.2.2 Homogene Gleichung

Die Differentialgleichung wird als homogen bezeichnet, wenn die rechte Seite auf Null gesetzt ist, d. h. ohne Störfunktion. Sie wird daher wie folgt beschrieben:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0. \quad (11.10)$$

Es ist möglich, die Gleichung mit der folgenden Substitution auf den ersten Grad zu reduzieren wie im Kapitel 5.1 schon beschrieben. Die Reduktion ist

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \mu(1 - x^2) & -\mu \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \quad (11.11)$$

Die Lösung der Differentialgleichung neigt unabhängig vom Anfangswert dazu, nach einer bestimmten Zeit in einer bestimmten Trajektorien zu oszillieren. Diese Orbit wird Grenzzyklus oder *limit cycle* genannt. Mit dem Parameter μ wird die Form der Umlaufbahn verändert, wie in Abbildung 11.2 dargestellt. Es ist interessant zu bemerken, dass bei $\mu = 0$ die Gleichung zu $\ddot{x} + x = 0$ wird, also eine linear harmonische Schwingung. Die homogene Van der Pol Gleichung ist stabil und zeigt *kein* chaotisches Verhalten.

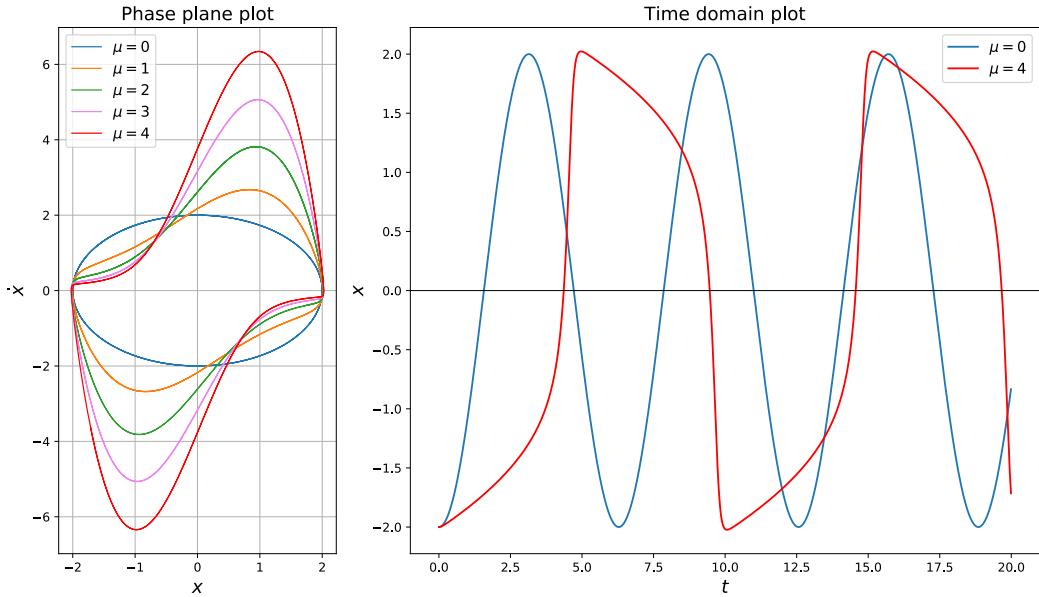


Abbildung 11.2: Phase und Zeit Raum Diagramm der Gleichung (11.11) mit verschiedenen Werten von μ

11.2.3 Inhomogene Gleichung

Die inhomogene Form der Van der Pol Gleichung

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = f(t),$$

wird die sogenannte *angeregte Van der Pol Gleichung*. In unserem Fall wird die Störfunktion auf $f(t) = A \cdot \sin(\omega t)$ gesetzt:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = A \cdot \sin(\omega t). \quad (11.12)$$

Aus praktischer Sicht ist es so, als würde man einen Wechselstromgenerator an den Oszillatoren-Eingang anschliessen. Wie in der Gleichung (11.11), wird die Differentialgleichung auf die erste Ordnung reduziert:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ \mu(1 - x^2)\frac{dx}{dt} - x + A \cdot \sin(\omega t) \end{pmatrix}. \quad (11.13)$$

11.3 Numerische Lösung der Van der Pol Gleichung

Im folgenden Abschnitt werden die verschiedenen Implementierungen in Python gezeigt und erklärt, die auch für die Realisierung der verschiedenen Plots eingesetzt wurden.

```

1 || def RK_4(func, z_n, t_n, h):
2 ||     """
3 ||     z_n: Aktueller Zustandsvektor [y_n, y_n']
4 ||     t_n: Zeitvariable
5 ||     h : Integrationsschritt
6 ||     """
7 ||     k1 = h*func(z_n, t_n)
8 ||     k2 = h*func(z_n + 0.5*k1, t_n + 0.5*h)
9 ||     k3 = h*func(z_n + 0.5*k2, t_n + 0.5*h)
10 ||    k4 = h*func(z_n + k3, t_n + h)
11
12 ||    return (k1 + 2*k2 + 2*k3 + k4)/6

```

Listing 11.1: Implementierung des Runge Kutta Algorithmus in Python

11.3.1 Implementation des Runge-Kutta-Verfahrens

Der Codesabschnitt in Listing 11.1 stellt die Python Implementierung des Runge-Kutta-Algorithmus vierter Ordnung vor. Die Koeffizienten werden gemäss der analytischen Definition

$$\begin{aligned} k_1 &= h \cdot f(t_n, y_n), \\ k_2 &= h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\ k_3 &= h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\ k_4 &= h \cdot f(t_n + h, y_n + hk_3), \end{aligned}$$

berechnet und dann wird der gewichtete Durchschnitt

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5) \quad (11.14)$$

ermittelt.

Implementation des Euler-Verfahrens

Der Code des Euler Algorithmus wurde nicht gezeigt, denn es handelt sich um eine vereinfachte Version des Codesabschnitt 11.1, in dem nur der Koeffizient k_1 berechnet werden muss. Da dies ein einziger Wert ist, ist es nicht notwendig, einen Mittelwert zu bilden. Die Berechnung wird daher auf

$$\begin{aligned} k_1 &= h \cdot f(t_n, y_n), \\ y_{n+1} &= y_n + k_1 + O(h^2). \end{aligned}$$

vereinfacht.

11.3.2 Implementation der Differentialgleichung

Der Codesabschnitt 11.2 zeigt stattdessen die Implementierung der Differentialgleichung (11.13). Die entwickelte Idee lautet wie folgt: In beiden Fällen stellt die Variable z_n einen Vektor dar, der

```

1  def VDP_funk(z_n, t_n):
2      """
3          z_n: Aktueller Zustandsvektor [y_n, y_n']
4          t_n: Zeitvariable
5      """
6          mu = 8.53
7          x_n = z_n[0]
8          v_n = z_n[1]
9          stoer_funk_n = 1.2*np.sin(t_n * 2*np.pi * 0.1)
10
11         f_x = v_n
12         f_v = mu*(1 - x_n**2)*v_n - x_n + stoer_funk_n
13
14     return np.array([f_x, f_v], float)

```

Listing 11.2: Implementierung der Van der Pol Gleichung in Python

die beiden Werte x_n und \dot{x}_n enthält. Beide Funktionen erhalten auch einen zweiten Eingabeparameter t_n , der dem aktuellen Zeitpunkt entspricht. Der Code in Listing 11.2 kümmert sich um die Definition der Differentialgleichung, die dann in der in Listing 11.1 eingeführten Funktion verwendet wird, um den Kutta-Runge-Algorithmus auszuführen und x_{n+1} und \dot{x}_{n+1} als Ausgabe zu erhalten. Zusammengefasst: ein bestimmter Zeitwert t_n , der aktuelle Wert der Funktion x_n und seine Ableitung \dot{x}_n werden als Input für die beiden Funktionen gegeben. Es wird eine Ausgabe zurückgegeben, die dem nächsten Wert der Funktion x_{n+1} und ihrer Ableitung \dot{x}_{n+1} entspricht. Es wird in der Form

$$\begin{pmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ \dot{x}_n \end{pmatrix} + \frac{1}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \quad (11.15)$$

dargestellt. Die Konstanten von \vec{k}_1 bis \vec{k}_4 sind Vektoren, die die Koeffizienten für beide benötigten Lösungen enthalten, nämlich x_{n+1} und \dot{x}_{n+1} . Diese werden dann gleichzeitig berechnet. Die Implementierung des in der Gleichung (11.15) beschriebenen Verfahrens wird in Listing 11.3 gezeigt. Um eine vollständige Lösung für einen beliebigen Bereich $[a, b]$ zu erhalten, muss man die darin enthaltenen Zeitwerte berechnen. Dieses Intervall muss also diskretisiert werden, indem es in n gleiche Teile aufgeteilt wird. Daraus ergibt sich eine Menge von Zeitwerten, die von t_{\min} bis t_{\max} läuft. Dies ist genau das, was im Codesabschnitt 11.3 ausgeführt wird. Zu Beginn werden das Zeitintervall und die Anfangsbedingungen angegeben, ab denen die Iteration dann beginnt, was in diesem Fall zu einer Lösung von $t_{\min} = 0.0$ bis $t_{\max} = 950.0$ führt.

11.4 Chaos im Van der Pol System

Die Änderung der Amplitude A in die Gleichung (11.13) verändert das Systemverhalten. Bei der Beobachtung der Frequenzspektrum des Oszillators wurden drei verschiedene Fälle erkannt. Wie in der Abbildung 11.3 gezeigt, ist die Frequenz des Van der Pol Oszillators dominant, wenn der Wert A kleiner als der kritische Wert ist. Wenn der Wert höher ist, dominiert die Frequenz ω . Beim kritischen Punkt gibt es eine Summe von Frequenzen im System. Durch Setzen der Werte $\mu = 0.2$ und $\omega = 1.15$ beträgt der kritische Wert von A in diesem Fall $A = 0.32$. Die Funktion

$$\ddot{x} - 0.2(1 - x^2)\dot{x} + x = 0.32 \cdot \sin(1.15t). \quad (11.16)$$

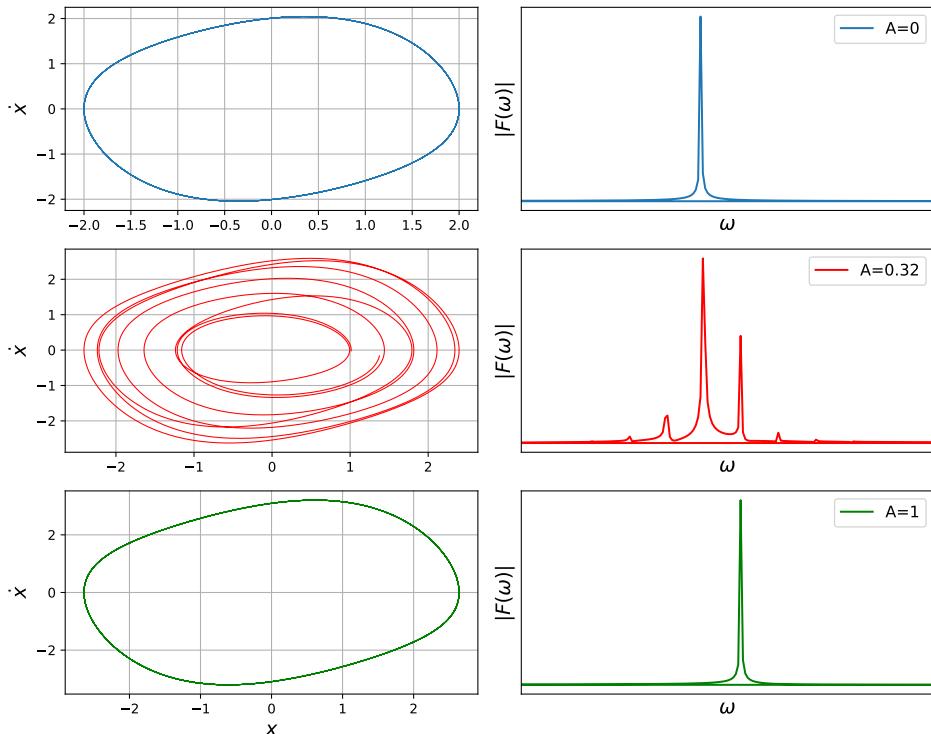
zeigt daher ein chaotisches Verhalten.

```

1  def solve(step_input, initial_condition=(0.0,1.0)):
2      t_min, t_max = 0.0, 950.0
3      h            = step_input
4      t_points     = np.arange(t_min, t_max, h)
5      x_points    = np.zeros([])
6      y_points    = np.zeros([])
7
8      # Setzen von Anfangsbedingungen fuer die Zustandsvariablen
9      x_0,v_0      = initial_condition
10     temp_z       = np.array([x_0, v_0], float)
11
12    # Loesen fuer die Zeitentwicklung
13    for t_n in t_points:
14        x_points = np.append(x_points, [temp_z[0]])
15        y_points = np.append(y_points, [temp_z[1]])
16        temp_z += RK_4(VDP_funk, temp_z, t_n, h)
17
18    return (x_points, y_points, t_points)

```

Listing 11.3: Python Algorithmus zur Berechnung der Lösung

Abbildung 11.3: Phasoraum Diagramm und Frequenzspektrum der Gleichung (11.16) mit verschiedenen Werten von A .

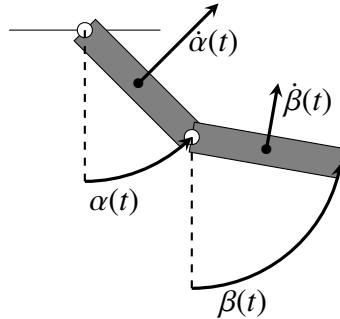


Abbildung 11.4: Doppelpendel

11.4.1 Anfangsbedingung

“Chaos: When the present determines the future, but the approximate present does not approximately determine the future.” [2]

– Edward Lorenz

Das angeführte Zitat ist genau die Definition, die das in diesem Kapitel untersuchte Problem erklärt, nämlich die Instabilität der numerischen Methoden. Letztere führen zu einem Rundungsfehler, der sich während der verschiedenen Iterationen ansammelt. Diese hängt von der für den Integrationsschritt gewählten Länge ab. Mehrere Längen entsprechen daher unterschiedlichen Werten des Fehlers. Per Definition ist ein chaotisches System sehr empfindlich gegenüber Veränderungen, auch wenn diese minimal sind. Zwei Fehler aufgrund von zwei unterschiedlichen Integrationsschritten können somit zu zwei völlig unterschiedlichen Lösungen führen. In der Regel tritt die Divergenz zwischen den beiden Lösungen nach einer bestimmten Zeitspanne auf. Etwas Ähnliches geschieht immer in einem chaotischen System, wenn eine kleine Variation der Anfangsbedingungen eingeführt wird. Dieser kleine Unterschied kann, analog zu dem durch Fehler verursachten, das Langzeitverhalten des Systems stark beeinflussen. Diese Empfindlichkeit gegenüber Veränderungen im Ausgangszustand ist auch bekannt als *Butterfly Effect*. Zur Erklärung dieses Phänomens verwendete Edward Lorenz in 1972 auf einer Konferenz den Satz: “Ein Schmetterling schlägt in Peking mit den Flügeln, und in New York kommt Regen statt Sonne”. Kann also eine solch minimale Veränderung der Anfangsbedingungen das Endergebnis vollständig verändern? Die Antwort ist offensichtlich ja, was auch durch die Grafiken auf den folgenden Seiten bewiesen wird.

Doppelpendel [1]

Im Falle des Bildes Abbildung 11.5 wurde die Lösung des im Abbildung 11.4 dargestellten Systems, ein Doppelpendel, dargestellt. Es wird natürlich nicht im Detail behandelt, da dies nicht die Aufgabe dieses Kapitels ist. Es handelt sich jedoch um ein praktisches Beispiel, das genau zeigt, was im Anfangszitat zusammengefasst ist. Die beiden gezeigten Funktionen stellen den α Winkel zweier Doppelpendel mit Anfangsbedingungen dar, die eine geringe Abweichung aufweisen. Die Ausgangssituation wird also durch die folgenden Werte beschrieben:

- Rote Kurve in Abbildung 11.5: $\alpha_1(0) = \frac{\pi}{2} \text{ rad}$
- Blaue Kurve in Abbildung 11.5: $\alpha_2(0) = \alpha_1(0) + \delta, \quad \delta = 10^{-3} \text{ rad}$

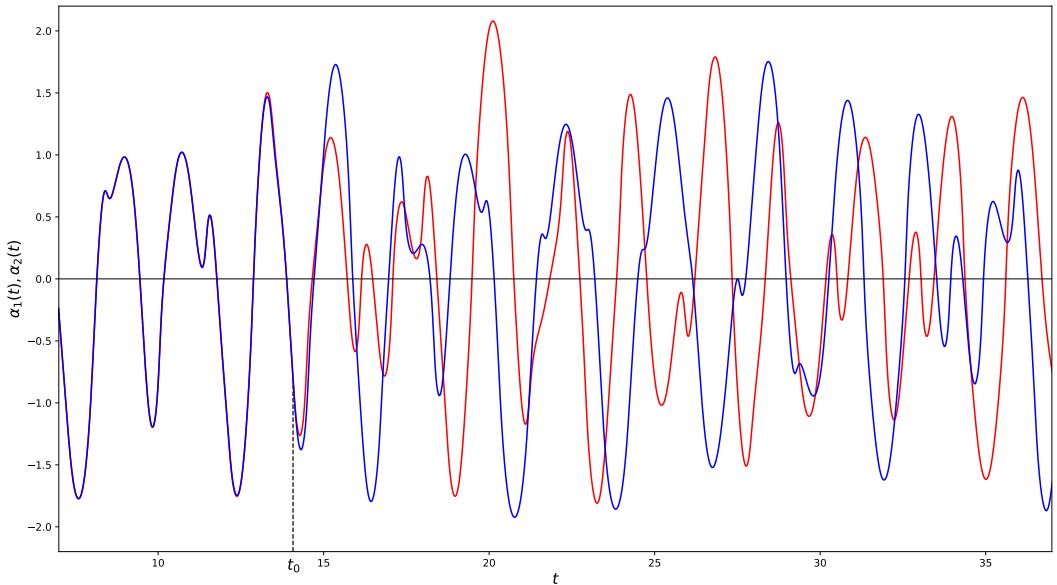


Abbildung 11.5: Verlauf der Winkel $\alpha_1(t)$, $\alpha_2(t)$ des Doppelpendels in Abbildung 11.4 und Angabe des Divergenzpunktes t_0

- $\dot{\alpha}_1(0) = \dot{\alpha}_2(0) = 1 \frac{m}{s}$, $\dot{\beta}_1(0) = \dot{\beta}_2(0) = 1 \frac{m}{s}$, $\beta_1(0) = \beta_2(0) = \frac{\pi}{2} \text{ rad}$

Wie in Abbildung 11.5 zu sehen ist, überlagern sich die beiden Lösungen im Anfangsmoment und sind nicht zu unterscheiden. Ab dem Punkt t_0 merkt man, dass sie plötzlich ganz andere Bahnen verfolgen. Der geringe anfängliche Unterschied von δ hat daher die Endergebnisse in nicht gleichgültiger Weise beeinflusst.

Van der Pol Gleichung

Dasselbe geschieht auch im Fall der Van der Pol Gleichung. Im Abbildung 11.6 sind zwei Darstellungen des Verlaufs der beiden mit dem gleichen Integrationsschritt erhaltenen numerischen Lösungen der Gleichung (11.13) im Zeitbereich gezeigt. Die Anfangsbedingung sind in diesem Fall wie folgt gewählt worden:

- Rote Kurve in Abbildung 11.6: $\dot{x}_1(0) = 5$
- Blaue Kurve in Abbildung 11.6: $\dot{x}_2(0) = \dot{x}_1(0) + \delta$, $\delta = 10^{-3}$
- $x_1(0) = x_2(0) = 0$

Daraus lässt sich natürlich schliessen, dass beide Systeme aufgrund ihrer chaotischen Natur sehr empfindlich auf die Anfangsbedingungen reagieren. Dies lässt sich auf das Zitat zurückführen, denn im Falle einer numerischen und daher approximativen Lösung \hat{y} , gilt dass

$$|\hat{y}(x_n) - y(x_n)| = \epsilon. \quad (11.17)$$

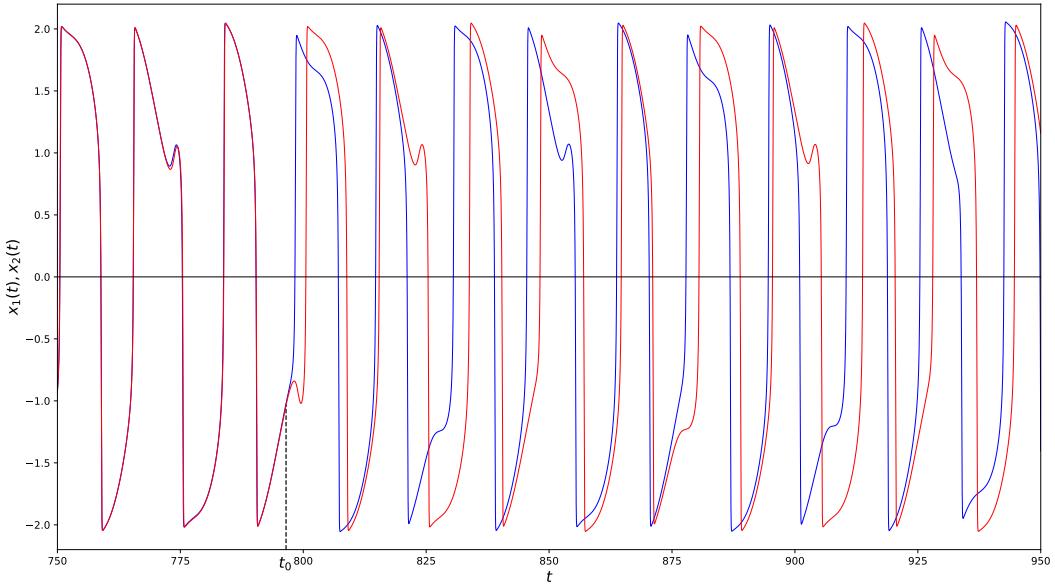


Abbildung 11.6: Verlauf der Lösungen $x_1(t)$, $x_2(t)$ der Gleichung (11.13) und Angabe des Divergenzpunktes t_0

Obwohl ϵ ein sehr kleiner Wert sein kann, kann er unter bestimmten Umständen die gleiche Wirkung haben wie eine Änderung der Anfangsbedingungen. Dank der beiden Diagramme Abbildung 11.6 und Abbildung 11.5 konnte man feststellen, dass der Unterschied zwar in der Größenordnung von 10^{-3} lag, aber nach einiger Zeit zu einem völlig anderen Verlauf führte. Wie im nächsten Kapitel erläutert, gilt dies auch für die Änderung des Integrationsschrittes.

11.4.2 Ergebnisse

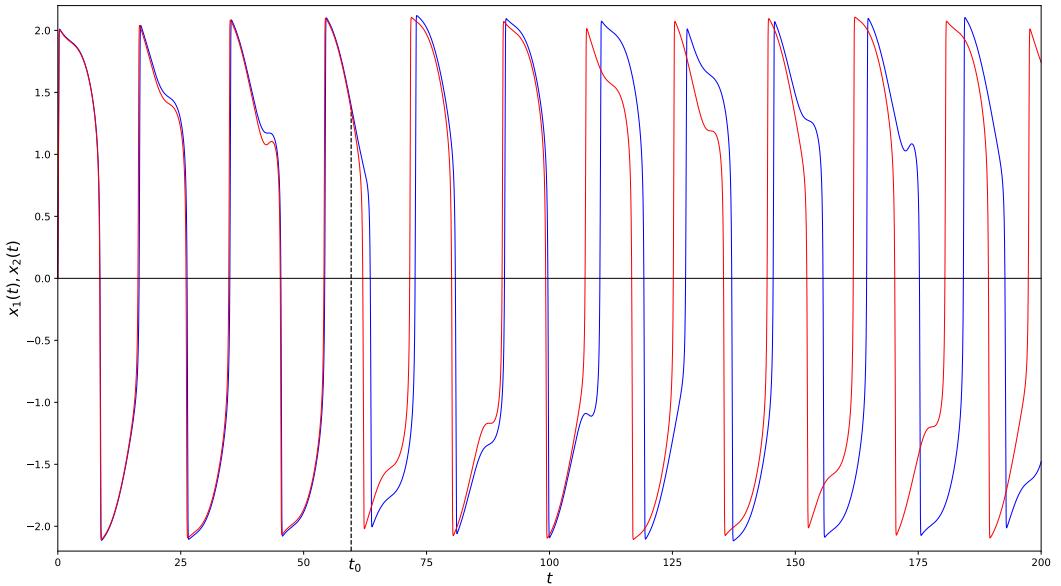
Nachstehend sind die Ergebnisse der beiden oben beschriebenen Algorithmen aufgeführt. Die ausgewählten Anfangsbedingungen sind:

- $\dot{x}_1(0) = \dot{x}_2(0) = 5$
- $x_1(0) = x_2(0) = 0$

Im Gegensatz zu den in Abbildung 11.6 gezeigten Daten werden nun zwei verschiedene Integrationsschritte ausgewählt, während die Anfangsbedingungen gleich bleiben. Die Ergebnisse sind jedoch fast die Gleichen. Im Falle der Runge-Kutta-Lösung wird deutlich, dass der Divergenzpunkt aufgrund der Änderung der Integrationsschritt noch vor demjenigen liegt, der sich durch die Veränderung der Anfangsbedingungen ergibt. Auch in diesem Fall verursachte eine Abweichung von 10^{-3} eine starke Änderung. Der Runge-Kutta-Algorithmus hat einen Rundungsfehler in der Größenordnung von $O(h^5)$, während der Euler-Algorithmus $O(h^2)$ hat. Dies wirkt sich auf die Zeit t_0 aus. Wo also der Fehler grösser ist, d. h. im Falle von Euler, haben wir eine Vorwegnahme der Divergenz. In der Tabelle 11.1 wurden die verschiedenen Parameter mit den relativen Abbildungen aufgelistet, in denen die berechneten numerischen Lösungen dargestellt sind. Es zeigt sich, dass die Lösungen trotz

Verfahren	h_1	h_2	Δ_h	$t_0 \cong$	Abbildung
Euler	0.01	0.009	$1 \cdot 10^{-3}$	59.7	11.7
Runge-Kutta	0.01	0.009	$1 \cdot 10^{-3}$	770.5	11.8
Runge-Kutta	0.009	0.007	$2 \cdot 10^{-3}$	551.2	11.9
Runge-Kutta	0.007	0.005	$2 \cdot 10^{-3}$	476.5	11.10

Tabelle 11.1: Ergebnisse der Divergenzzeiten, berechnet mit der Software in Python.

Abbildung 11.7: Verlauf der beiden mit der *Euler-Methode* berechneten Lösungen x_1 und x_2 der Gl. (11.13) im Zeitbereich und Angabe des Divergenzpunktes t_0 . Integrationsschritte: $h_1 = 0.01$, $h_2 = 0.009$

der Abnahme des Integrationsschrittes immer einen Punkt der Divergenz t_0 erreichen. Ein kleinerer Integrationsschritt führt also nicht zu einer höheren Genauigkeit, sondern zu einer Verschiebung des Punktes t_0 nach rechts. Jenseits dieses Punktes konvergieren die beiden Lösungen nicht und können daher nicht als korrekt betrachtet werden.

11.5 Folgerungen

Zusammenfassend kann man sagen, dass es möglich ist, Differentialgleichungen mit verschiedenen numerischen Methoden ohne allzu grosse Probleme zu lösen. Es kann auch entschieden werden, mit welcher Präzision man es tun möchte, natürlich auf Kosten längerer Rechenzeiten. Dieses Kapitel zeigt jedoch, dass einige Systeme, wie das von Van der Pol, sich unter bestimmten Umständen chaotisch verhalten können. Wenn man letztere in Betracht zieht, selbst wenn die Länge des Integrationsschrittes mit der Idee der Genauigkeitssteigerung verkürzt werden kann, wird das Ergebnis immer noch falsch sein. Der Wert des durch die numerische Methode eingeführten Fehlers wird

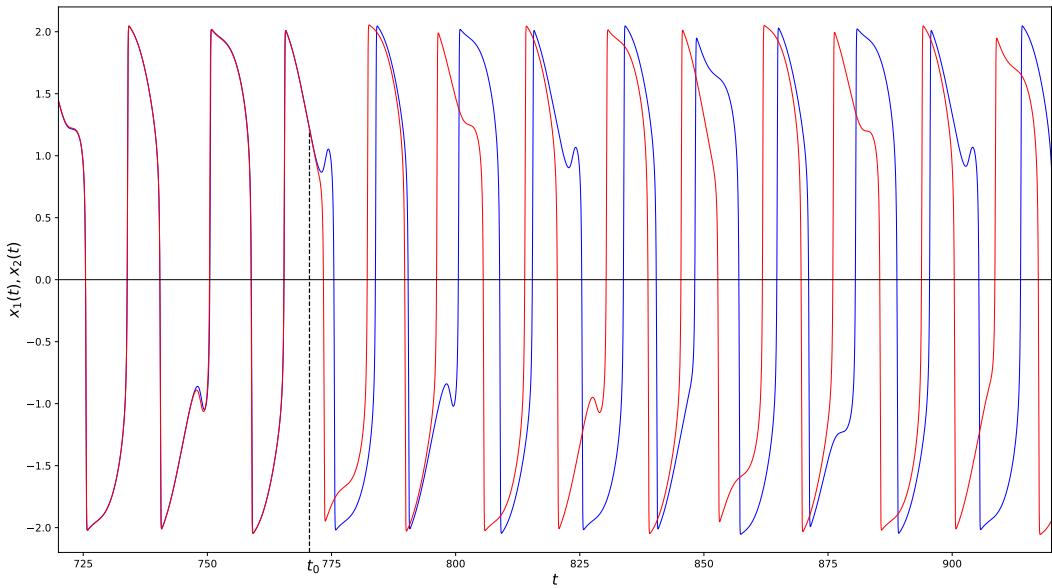


Abbildung 11.8: Verlauf der beiden mit der *Runge-Kutta-Methode* vierter Ordnung berechneten Lösungen x_1 und x_2 der Gleichung (11.13) im Zeitbereich und Angabe des Divergenzpunktes t_0 . Integrationsschritte: $h_1 = 0.01$, $h_2 = 0.009$

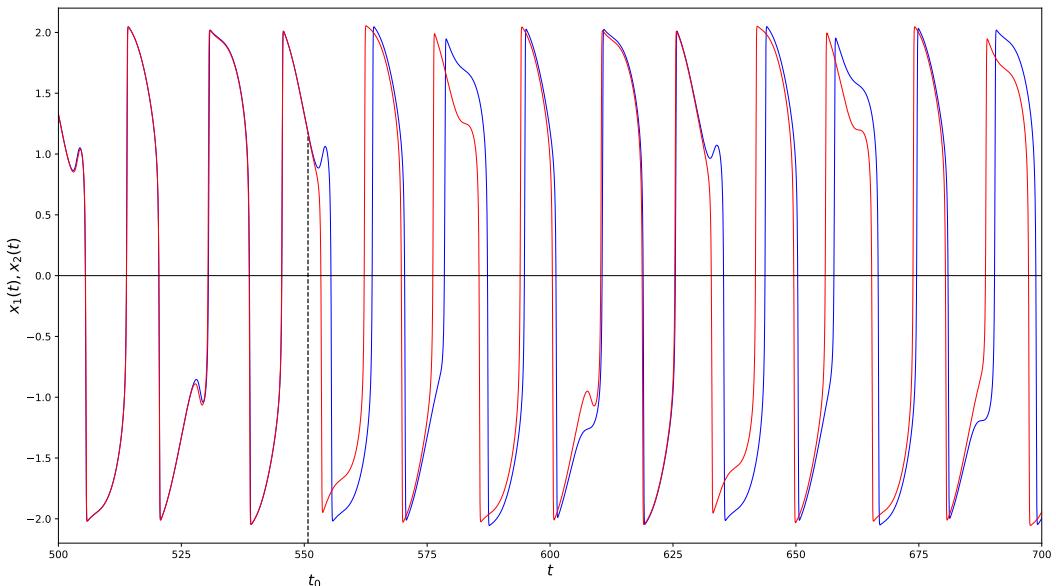


Abbildung 11.9: Verlauf der beiden mit der *Runge-Kutta-Methode* vierter Ordnung berechneten Lösungen x_1 und x_2 der Gleichung (11.13) im Zeitbereich und Angabe des Divergenzpunktes t_0 . Integrationsschritte: $h_1 = 0.009$, $h_2 = 0.007$

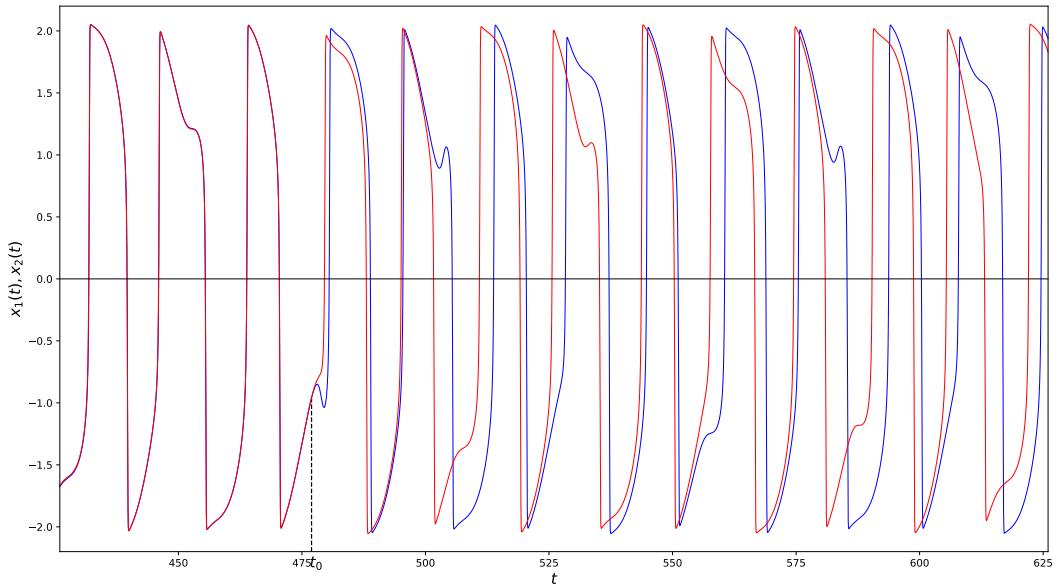


Abbildung 11.10: Verlauf der beiden mit der *Runge-Kutta-Methode* vierter Ordnung berechneten Lösungen x_1 und x_2 der Gleichung (11.13) im Zeitbereich und Angabe des Divergenzpunktes t_0 . Integrationsschritte: $h_1 = 0.007$, $h_2 = 0.005$

mit fortschreitender Iteration grösser, was eher zu einer Divergenz als zu einer Annäherung der Lösung führt. Eine mögliche Methode, um die Auswirkungen der letzteren zu reduzieren, könnte zum Beispiel die Schrittängensteuerung sein, d.h. eine variable Integrationsschritt zu haben. Kapitel 13 dieses Buches ist diesem letzten Thema gewidmet.

Literatur

- [1] Joe Chen. “Chaos from simplicity: an introduction to the double pendulum”. In: (2008). URL: <http://hdl.handle.net/10092/12659>.
- [2] Edward N. Lorenz. “Deterministic Nonperiodic Flow”. In: *Journal of the Atmospheric Sciences* 20.2 (März 1963), S. 130–141.
- [3] Balthasar van der Pol. *A theory of the amplitude of free and forced triode vibrations*. Radio Review (later Wireless World), 1920.

12

Taylor-Reihe und Differentialgleichungen

Fabio Daniel Marti

12.1 Einleitung

In diesem Kapitel geht es um die numerische Lösung einer Differentialgleichung mit Hilfe der Taylor-Reihe. Man möchte auf die Lösung einer Differentialgleichung kommen, ohne die Differentialgleichung lösen zu müssen. Dazu werden die Ableitungen n -ter Ordnung benötigt, welche numerisch aus den vorhergehenden Punkten oder auch algebraisch aus der bekannten Differentialgleichung berechnet werden können.

12.2 Problemstellung

Um die numerische Approximation an einem konkreten Beispiel zeigen zu können wird hier die logistische Differentialgleichung analysiert. Es handelt sich hierbei um eine begrenzte Wachstumsfunktion. Mit dieser Differentialgleichung können zum Beispiel der Verlauf einer Krankheit, der Lebenszyklus eines Produktes oder auch Zerfallsprozesse beschrieben werden. Die Parameter der logistischen Differentialgleichung

$$y' = k \cdot y(L - y) \quad \text{mit der Anfangsbedingung} \quad y(0) = 0.5$$

haben die Bedeutung: L = Endwert und k = Wachstumsrate. Wir normieren den Endwert L auf 1. Der Endwert entspricht zum Beispiel der Anzahl Infizierten am Ende einer Epidemie. In der Abbildung 12.1 ist zu erkennen, dass der Wachstumsfaktor k bestimmt, wie steil die Funktion wird. Es existiert nun also für jeden möglichen Punkt der logistischen Funktion in diesem Richtungsfeld ein genau definierter Wert k . Um das k zu bestimmen, müsste die Differentialgleichung gelöst werden. Möchte man dies umgehen, weil das zum Beispiel viel Rechenaufwand bedeuten würde, bzw. für einen Computer viel Rechenzeit bedeutet, kann die Lösung der Differentialgleichung mit kleinen Schritten

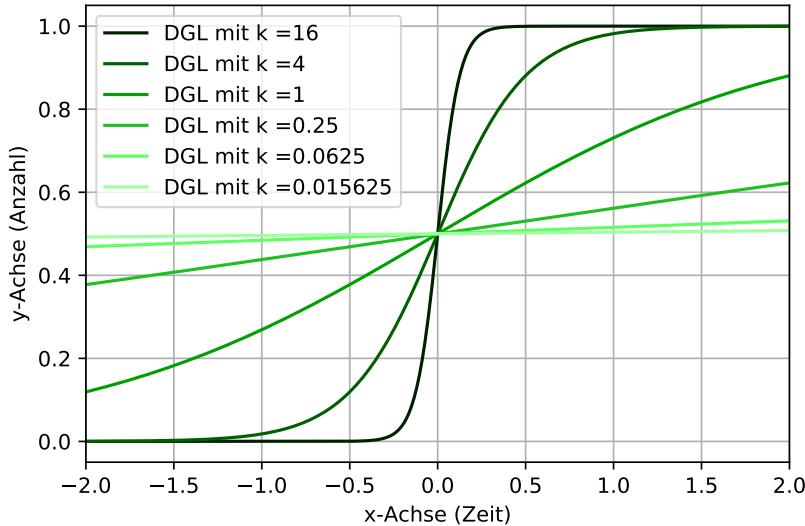


Abbildung 12.1: Logistische Funktion in Abhängigkeit von k

erarbeitet werden. Dazu vergleichen wir die beiden Inkrementfunktionen Runge-Kutta- und Taylor-Approximation. Der Wachstumsfaktor

$$k = \frac{-\ln(\frac{1}{y} - 1)}{x} \quad \left(\text{aus der Lösung } y(x) = \frac{1}{1 + e^{-kx}} \right) \quad (12.1)$$

ist also sowohl von x als auch von y abhängig, allerdings nur durch x und y berechenbar wenn die Differentialgleichung gelöst wurde. Die Taylor-Approximation darf sich dieser Formel deshalb nicht bedienen.

12.3 Das Taylor-Verfahren

Wir benötigen für die Taylor-Approximation die Ableitungen bis zur n -ten Ordnung. Also bis zur Ordnung, mit welcher wir die Differentialgleichung approximieren wollen. Wie man auf diese kommt, ist in Kapitel 5.1.4 beschrieben. Man erhält die Gleichungen

$$\begin{aligned} y' &= k \cdot (y - y^2) \\ y'' &= k \cdot (y' - 2y' \cdot y) \\ y''' &= k \cdot (y'' - 2y'' \cdot y + y'^2) \\ y'''' &= k \cdot (y''' - 2y''' \cdot y + 3y'' \cdot y') \\ y''''' &= k \cdot (y'''' - 2y'''' \cdot y + 4y'' \cdot y' + 3y'^2) \end{aligned} \quad (12.2)$$

durch Ableiten der logistischen Funktion.

n	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$
1	0.45519	0.31990	0.11077	$4.68470 \cdot 10^{-2}$	$6.19930 \cdot 10^{-2}$
2	0.38755	$3.85299 \cdot 10^{-2}$	$5.97521 \cdot 10^{-2}$	$1.05400 \cdot 10^{-2}$	$4.07133 \cdot 10^{-2}$
5	0.14751	$2.39004 \cdot 10^{-2}$	$2.59349 \cdot 10^{-2}$	$1.33486 \cdot 10^{-2}$	$1.88327 \cdot 10^{-2}$
10	$6.88690 \cdot 10^{-2}$	$4.37589 \cdot 10^{-3}$	$4.37432 \cdot 10^{-3}$	$2.58736 \cdot 10^{-3}$	$3.50711 \cdot 10^{-3}$
1000	$6.19312 \cdot 10^{-4}$	$5.23506 \cdot 10^{-7}$	$1.84704 \cdot 10^{-7}$	$1.85763 \cdot 10^{-7}$	$1.85765 \cdot 10^{-7}$
100000	$6.18453 \cdot 10^{-6}$	$5.13468 \cdot 10^{-11}$	$1.96497 \cdot 10^{-11}$	$1.96507 \cdot 10^{-11}$	$1.96507 \cdot 10^{-11}$

Tabelle 12.1: Fehler der numerischen Lösung einer Differentialgleichung nach dem Runge-Kutta-Verfahren: m = Ordnung, n = Schritte, Start = -5, Ende = -0.02, (letzte Ziffer abgerundet)

Die allgemeine Formel der Einschritttheorie sieht so aus:

$$y(x + h) = y(x) + h \cdot f(x, y), \quad (12.3)$$

wobei die Funktion $f(x, y)$ der Approximationsfunktion entspricht, mit welcher der nächste Punkt berechnet wird.

Beim Taylor-Verfahren soll nun also mit kleinen Schritten der Verlauf der Funktion ausgehend von einem Anfangswert ermittelt werden. Die Taylor-Reihe

$$f_T(x, a) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} \cdot (x - a)^n \quad (12.4)$$

setzt sich zusammen aus der korrekten Gewichtung der Ableitungen bis zur gewünschten Ordnung. Je höher der Grad der Auswertung der Taylor-Reihe gewählt wird, desto grösser ist der Bereich um den Auswertungspunkt a in welchem die Approximation mit der logistischen Funktion übereinstimmt. Am Auswertungspunkt wird nun mit der beschriebenen Approximation der nächste Punkt berechnet. Dieses Verfahren mit der Inkrementfunktion

$$y(x + h) = \sum_{n=0}^g \frac{f^{(n)}(x + h)}{n!} \cdot h^n \quad (12.5)$$

nennt man nun das Taylor-Approximationsverfahren. Dort werden erneut die Ableitungen ausgerechnet und eine weitere Approximationsfunktion aufgestellt, mit der dann der darauf folgende Punkt bestimmen wird. Je kleiner nun die Schrittweite gewählt wird, desto weniger spielen höhere Ableitungen, beziehungsweise höhere Frequenzanteile, eine Rolle. In der Abbildung 12.2 sieht man ein Vergleich des Runge-Kutta-Verfahrens und des Taylor-Verfahrens. Detaillierte Resultate bieten die beiden Tabellen 12.1 und 12.2.

12.3.1 Auswertung des Vergleiches

Die ersten Spalten der beiden Tabellen 12.1 und 12.2 der beiden Verfahren sehen gleich aus. Das ist so, weil die Approximationen erster Ordnung auf das selbe herauskommen, nämlich auf das Approximieren der Funktion mit Geraden erster Ordnung. Das entspricht der Steigung im Auswertungspunkt.

Beim Punkt $n = 1$ und $m = 2$ ist die Taylor-Approximation besser als Runge-Kutta. Eine Approximation mit dem Grad 2 und einem Schritt über eine sich so stark verändernde Kurve ist aber eher ein Glückstreffer als ein mathematischer Erfolg. Es kann gesagt werden, dass das Runge-Kutta-Verfahren in diesem Fall ein kleineren Fehler aufweist. Dies ist auch in der Abbildung 12.3 zu sehen.

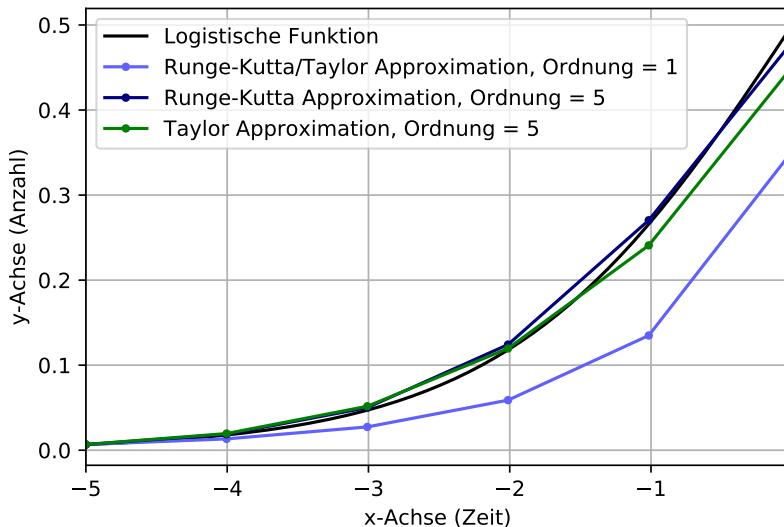


Abbildung 12.2: Gleichung und Approximationen der Logistische Funktion für $k = 1$, Approximationen mit 5 Schritten von -5 bis -0.02 und Startwert der selbe wie bei $k = 1$

n	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$
1	0.45519	0.27840	0.27948	0.29078	0.29515
2	0.38755	0.20040	0.19400	0.19579	0.19663
5	0.14751	$5.30816 \cdot 10^{-2}$	$4.80256 \cdot 10^{-2}$	$4.79241 \cdot 10^{-2}$	$4.79851 \cdot 10^{-2}$
10	$6.88690 \cdot 10^{-2}$	$1.60657 \cdot 10^{-2}$	$1.38830 \cdot 10^{-2}$	$1.38673 \cdot 10^{-2}$	$1.38751 \cdot 10^{-2}$
1000	$6.19312 \cdot 10^{-4}$	$2.11126 \cdot 10^{-6}$	$1.90164 \cdot 10^{-6}$	$1.90183 \cdot 10^{-6}$	$1.90183 \cdot 10^{-6}$
100000	$6.18453 \cdot 10^{-6}$	$2.10965 \cdot 10^{-10}$	$1.90226 \cdot 10^{-10}$	$1.90226 \cdot 10^{-10}$	$1.90226 \cdot 10^{-10}$

Tabelle 12.2: Fehler der numerischen Lösung einer Differentialgleichung nach dem Taylor-Verfahren: $m = \text{Ordnung}$, $n = \text{Schritte}$, Start = -5 , Ende = -0.02 , (letzte Ziffer abgerundet)

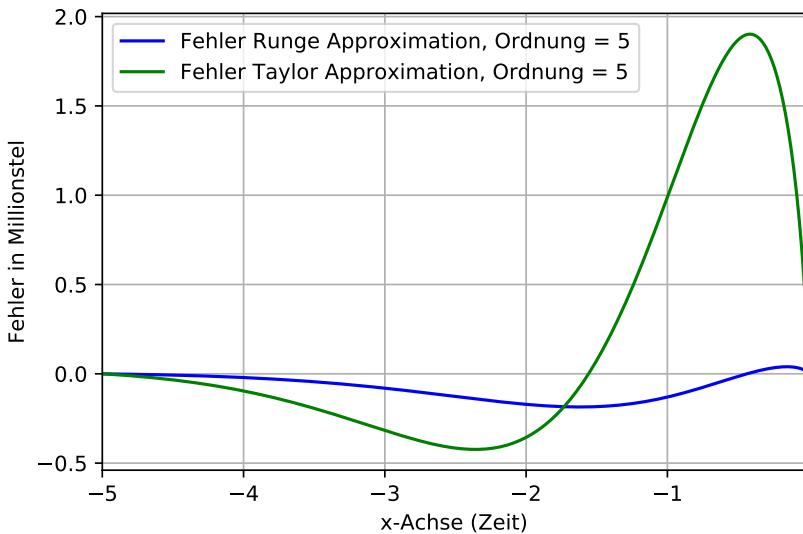


Abbildung 12.3: Fehler der Approximationen bei 1000 Schritten von -5 bis -0.02 und Startwert der selbe wie bei $m = 1$

12.4 Probleme mit der logistischen Funktion

Die logistische Funktion wurde gewählt, weil es eine anschauliche, nichtlineare Differentialgleichung ist welche sich somit gut eignet, um das Runge-Kutta-Verfahren mit dem Taylor-Verfahren zu vergleichen. Dabei traten allerdings folgende zwei Probleme auf.

12.4.1 Konvergenz

Wie in Abbildung 12.1 ersichtlich ist, läuft die Lösung der Differentialgleichung im Punkt 0 gegen 0.5 und bei ∞ gegen 1. Wird nun also diese Differentialgleichung approximiert, dann ist sowieso klar auf welchem Wert wir ungefähr landen werden. Um eine aussagekräftige Antwort auf die Frage der Qualität der Approximation geben zu können wurden in Abbildung 12.3 die Fehler über den ganzen Verlauf der Approximation verglichen.

12.4.2 Gefährliche Umgebung der Anfangsbedingung

Wie in Abbildung 12.1 klar dargestellt ist, ist die Ableitung in direkter Umgebung von der gewählten Anfangsbedingung $(0, 0.5)$ zwischen 0 und ∞ da der Punkt in der Lösung der Differentialgleichung einer Polstelle entspricht. Wenn also eine Approximation der Differentialgleichung durch diesen Punkt verläuft, oder auch nur in seine unmittelbare Umgebung kommt kann die Approximation komplett anders weiter verlaufen als man erwarten würde. Dies liegt daran, dass selbst kleine Fehler, welche eine Approximation zwangsläufig haben muss, sich in der Nähe des Punktes sehr stark auf den weiteren Verlauf auswirken. Um dies zu umgehen wurde die nähere Umgebung dieses Wendepunkts einfach aus der Approximation ausgeschlossen indem nur bis 0.02 davor approximiert

wurde. Man könnte nun aber die erhaltenen linke Kurve an dem Wendepunkt punktspiegeln und hätte so die zweite Hälfte ebenfalls, welche durch die Spiegelung sowieso redundant und für uns deshalb nicht interessant wäre.

12.5 Lösung

Die logistische Funktion wird also besser mit Runge-Kutta approximiert. Man könnte noch weitere Funktionen analysieren und versuchen, einen Fall zu finden wo Taylor besser geeignet wäre. Vermutlich könnte durch eine Kombination der beiden Verfahren, also durch ein Auswerten der Differentialgleichung zwischen zwei Auswertungspunkten mit dem Taylor eine Verbesserung erzielt werden.

12.6 Folgerungen aus dem Beispiel

Wie in Abbildung 12.3 ersichtlich ist, ist die Runge-Kutta-Approximationen im Vergleich zur Taylor-Approximation jeweils 5. Ordnung bezüglich der logistischen Funktion genauer. Dies gilt allerdings nur für den ausgewerteten Bereich der Gleichung als Ganzes. Da die Entwicklungen des Fehlers unterschiedlich verlaufen, sind beide Verfahren in gewissen Abschnitten besser. Da man aber normalerweise die originale Funktion nicht kennt und somit auch der Fehler unbekannt ist, ziehen wir hier den defensiven Schluss und sagen, dass im Fall der logistischen Funktion die Runge-Kutta-Approximation besser geeignet ist als die Taylor-Approximation.

12.7 Analytischer Vergleich der Inkrementfunktionen

Wird eine Funktion nur in einem Punkt und seiner unmittelbaren Umgebung betrachtet, so kann er durch eine Funktion 1. Grades approximiert werden, bzw. einer Geraden mit der selben Steigung. Die höheren Ordnungen spielen erst ab einem gewissen Abstand zum Auswertungspunkt eine Rolle. Wenn wir also genug nahe bleiben, können wir eine Funktion nur durch die erste Ableitung ziemlich gut approximieren. In diesem Fall sind die Taylor-Approximation und die Runge-Kutta-Approximation gleich gut, beziehungsweise die gleiche Formel. Bei den höheren Ordnungen optimiert das Runge-Kutta-Verfahren die Stelle von der es die Ableitung berechnet, indem versucht wird, einen repräsentativen Auswertungspunkt zu finden, welcher die Steigung zwischen zwei Auswertungspunkten bestmöglich trifft. Das Taylor-Verfahren hingegen nimmt höhere Ableitungen dazu, welche (bei kleinen Schritten!) kaum eine Rolle spielen. Der Vorteil von der Taylor-Approximation ist, dass die Approximationsfunktionen wesentlich länger mit der originalen Funktion übereinstimmen 12.4 und somit bei einer grossen Schrittweite deutlich besser sein müssen.

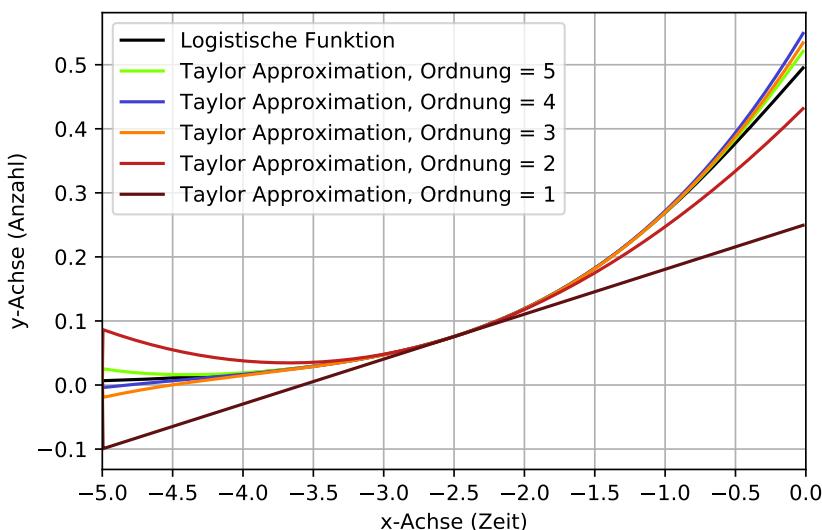


Abbildung 12.4: Taylor-Polynome unterschiedlichen Grades im Punkt -2.5 als Approximation für die logistische Gleichung mit $k = 1$ von -5 bis -0.02

13

Schrittängensteuerung

Reto Fritsche

13.1 Einleitung

Numerische Verfahren zur Berechnung der Lösungen von Differentialgleichungen wie das Euler- oder Runge-Kutta-Verfahren beruhen bekanntlich auf iterativem Vorgehen in Schritten, das bedeutet, dass der Verlauf der Kurve mit vielen Einzelschritten bis zum gesuchten Endpunkt angenähert wird und nicht exakt bestimmt werden kann. Um den Fehler möglichst klein zu halten, kann bei gegebenem Lösungsverfahren beispielsweise die Schrittänge zwischen den einzelnen Punkten verkleinert werden, wodurch jedoch der Rechenaufwand steigt. Die für die geforderte Genauigkeit nötige Schrittweite richtet sich nach dem Punkt, an dem die "Krümmung" der Kurve am grössten ist. Die Aufgabe der Schrittängensteuerung ist, die Schrittänge laufend der Dynamik der Kurve anzupassen, sodass der Zielpunkt mit möglichst wenig Schritten und der erforderlichen Genauigkeit berechnet werden kann.

13.2 Problemstellung

Die Schrittängensteuerung, oft auch als Schrittweitensteuerung bezeichnet, soll dazu dienen, numerische Berechnungen von Differentialgleichungen (z. B. mit dem Runge-Kutta-Verfahren) zu beschleunigen. Um die optimale Schrittänge zu bestimmen ist eine Fehlerschätzung nötig, was jedoch auch Rechenleistung beansprucht. Aus diesem Dilemma stammt auch einer der wichtigsten Knackpunkte des Themas: Wie findet man die optimale Schrittänge, ohne zu viel Rechenleistung dafür zu verschwenden? Denn wenn zu viel Energie in das Finden der optimalen Schrittweite gesteckt wird, ist der Gewinn im Vergleich zu einer fixen, kleineren Schrittänge zunicht.

Welche Wirkung verschiedene Schrittgrössen auf das Resultat haben können, wird in Grafik 13.1 gezeigt. In dieser Grafik wird die Bahn eines Teilchens um eine Punktmasse (Gravitationszentrum) mit unterschiedlichen Schrittweiten berechnet. Dabei ist ersichtlich, dass fernab dem Gravitationszentrum auch grössere Schritte ausreichende Genauigkeit liefern und die kleinen Schritte somit nur in der Nähe des Gravitationszentrums nötig sind, wo die Geschwindigkeit und die Beschleunigung gross sind.

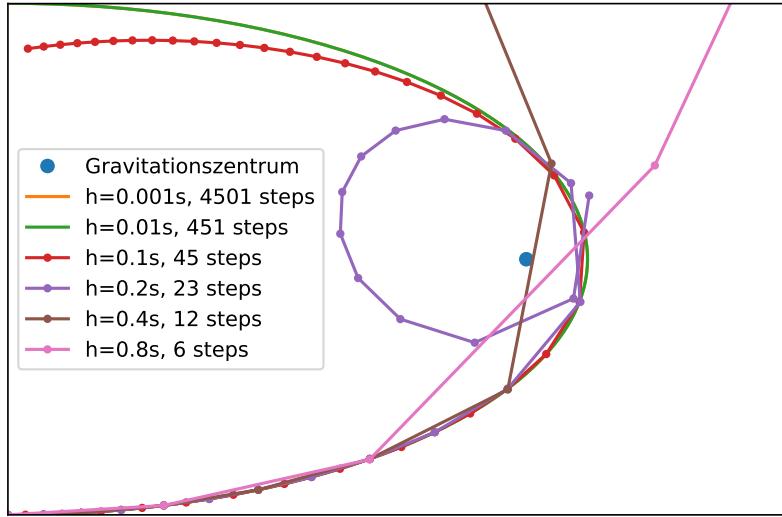


Abbildung 13.1: Bahn eines Teilchens, von unten links kommend, um ein Gravitationszentrum (P_{fix}), berechnet mit unterschiedlichen Schrittweiten (Runge-Kutta vierter Ordnung): $a(t) = P''(t) = \frac{(P_{\text{fix}} - P(t)) \cdot m \cdot g}{|P_{\text{fix}} - P(t)|^3}$ mit $P(0) = (0, 0)$, $P_{\text{fix}} = (2, 1)$, $P'(0) = v(0) = (0.669, 0)$, $g = 1$, $m = 1$.

13.3 Realisierung

Schrittweitensteuerungen lassen sich auf vielseitige Art und Weise realisieren, zwei Varianten davon werden nun näher vorgestellt.

13.3.1 Simple Schrittweitensteuerung mit konstanter Testschrittweite

Eine Möglichkeit zur Realisierung einer simplen Schrittweitensteuerung ist mittels einem konstanten Testschritt, dargestellt in Abbildung 13.2. Die Funktionsweise dieser Methode wird nun anhand der Wahrscheinlichkeitsfunktion der Standardnormalverteilung vorgeführt. Für die Verteilungsfunktion der Normalverteilung existiert zwar keine analytische Lösung, doch sie lässt sich aus der Wahrscheinlichkeitsdichtefunktion

$$\varphi(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{mit } \mu = 0, \sigma = 1$$

berechnen:

$$F(x) = \int_{-\infty}^x \varphi(\tilde{x}) \cdot d\tilde{x}.$$

Damit die Funktion mittels gängiger Methoden der Differentialrechnung analysiert werden kann, wird die Integralfunktion in die Differenzialgleichung

$$F'(x) = \varphi(x), \quad F(-\infty) = 0$$

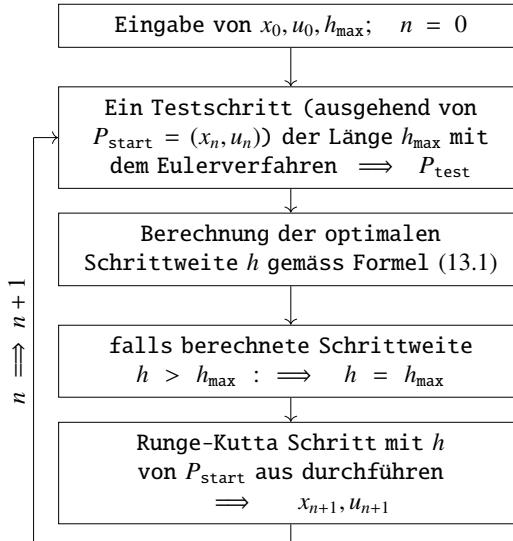


Abbildung 13.2: Algorithmus für eine simple Schrittweitensteuerung

umgewandelt. Im folgenden Beispiel zur Abbildung 13.3 wurde für die numerische Annäherung $\tilde{F}(x)$ an $F(x)$ bei $x_0 = -7$ mit Startwert $\tilde{F}(-7) = 0$ begonnen. Dabei wird für den Schritt n die Steigung $F'(x_n) = \varphi(x_n)$ am Punkt x_n, \tilde{F}_n (in Abbildung 13.3 oranger Punkt mit rotem Kreuz) berechnet. Damit wird ein Eulerschritt mit der Testschrittweite $h_{\text{test}} = 2$ (In Abbildung 13.3 als schwarzer Balken oberhalb der x -Achse) gemacht. Somit erhält man den Testpunkt $P_{\text{test}} = (x_n + h_{\text{test}}, \tilde{F}_n + \varphi(x_n, \tilde{F}_n) \cdot h_{\text{test}})$ (In Abbildung 13.3 als einsames rotes Kreuz am oberen Ende der Grafik). Nun wird auch an diesem Punkt die Steigung ermittelt. Sind die beiden Steigungen am Start und am Endpunkt des Testschrittes ziemlich ähnlich, wird davon ausgegangen, dass die Kurve in diesem Bereich nur wenig ihre Richtung ändert und deshalb eine grosse Schrittweite gewählt werden darf. Ist der Unterschied (Abbildung 13.3, untere Grafik, Differenz zwischen $\varphi(\text{start}), \varphi(\text{test})$) etwas grösser, wird die Schrittweite (In Abbildung 13.3 roter Balken innerhalb der Testschrittweite) etwas kleiner gewählt. Konkret berechnet sich in diesem Beispiel die zu wählende Schrittweite wie folgt:

$$h = \frac{h_{\text{test}}}{|\varphi(x_n, \tilde{F}_n) - \varphi(P_{\text{test}})| \cdot q_{\text{factor}}} \quad \text{mit } q_{\text{factor}} = 20. \quad (13.1)$$

Der Faktor q_{factor} wurde hierfür empirisch ermittelt. Zusätzlich wird die Schrittweite nach oben durch die Testschrittweite begrenzt, da über den Verlauf der Kurve jenseits des Testpunktes noch nichts bekannt ist. Auch eine Division durch Null muss abgefangen werden, falls die Steigung in beiden Punkten identisch ist. In diesem Fall soll die maximale Schrittweite $h = h_{\text{test}}$ gewählt werden. Mit der nun ermittelten Schrittweite wird ein Runge-Kutta-Schritt (4. Ordnung) gemacht, wobei einer der Koeffizienten (k_1) bereits beim Testschritt ermittelt worden ist.

Dieses Verfahren ist zwar in der Lage, die Schrittweite zu verkleinern, falls mit der Testschrittweite ein zu grosser Fehler resultieren würde, doch über den Fehler des gemachten Schrittes ist dann am Ende nichts bekannt. Eine weitere Schwäche dieses konstanten Testschrittes ist, dass bereits sehr kleine Schritte gewählt werden, sobald der Testschritt eine andere Steigung ertastet, obwohl der “Knick” der Kurve noch genügend weit entfernt wäre.

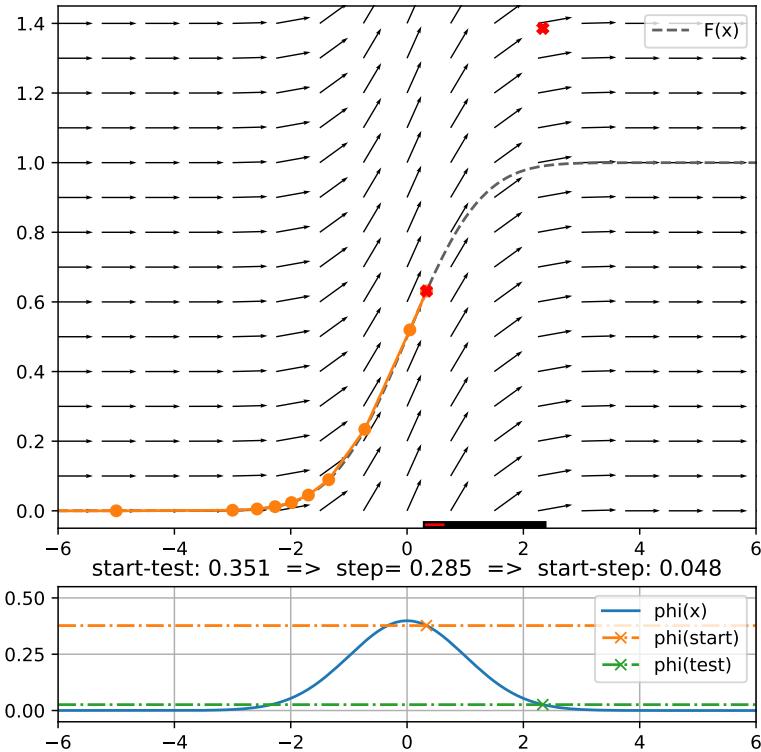


Abbildung 13.3: Berechnung der Wahrscheinlichkeitsfunktion $F(x)$ der Standardnormalverteilung aus dessen Dichtefunktion (oben mit Pfeilen, unten als Funktion $\phi(x)$) mit einer einfachen Schrittweitensteuerung

13.3.2 Schrittweitensteuerung nach Fehlberg

Ein anderer Ansatz zur Realisierung einer Schrittweitensteuerung ist mittels der Fehlerschätzung nach Fehlberg. Dabei wird der Fehler des aktuellen Schrittes anhand der Differenz zweier unterschiedlicher, paralleler Runge-Kutta Schritte geschätzt. Durch geschickte Kombination dieser beiden Schritte lässt sich der Mehrrechenaufwand dazu aber merklich reduzieren, was in der Literatur als *einbetten* bezeichnet wird. Damit diese Einbettung möglich wird, sind die einzelnen Stufen etwas anders verteilt und gewichtet als im Abschnitt 5.4.2. Wie solche Parameter für die einzelnen Stufen bestimmt werden können, wurde im Abschnitt 5.4.1 an Hand des verbesserten Euler-Verfahrens und des vereinfachten Runge-Kutta-Verfahrens erläutert. Weitere Details zum folgenden Beispiel sind im Kapitel *Eingebettete Verfahren und Schrittweitensteuerung* des Buchs [1] zu finden. In diesem Beispiel wird gezeigt, wie ein Runge-Kutta Schritt 4. Ordnung (5 Stufen) in ein Schritt 5. Ordnung (6 Stufen) eingebettet wird. Dabei werden zuerst alle Koeffizienten berechnet, welche für den Schritt 4. Ordnung nötig sind:

$$k_1 = f(x_j, u_j),$$

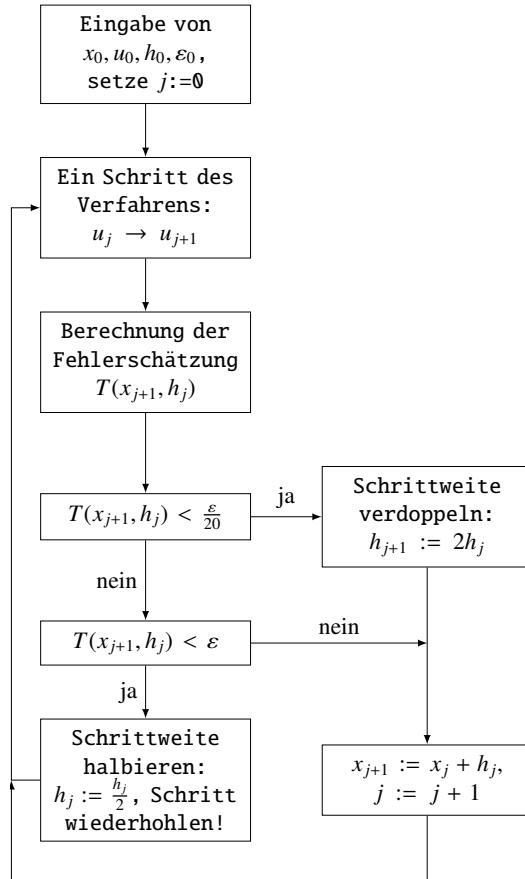


Abbildung 13.4: Algorithmus zur Schrittängensteuerung mittels Fehlerschätzung nach Fehlberg [1]

$$\begin{aligned}
 k_2 &= f\left(x_j + \frac{2}{9}h, u_j + \frac{2}{9}hk_1\right), \\
 k_3 &= f\left(x_j + \frac{1}{3}h, u_j + \frac{1}{12}hk_1 + \frac{1}{4}hk_2\right), \\
 k_4 &= f\left(x_j + \frac{3}{4}h, u_j + \frac{69}{128}hk_1 - \frac{243}{128}hk_2 + \frac{135}{64}hk_3\right), \\
 k_5 &= f\left(x_j + h, u_j - \frac{17}{12}hk_1 + \frac{27}{4}hk_2 - \frac{27}{5}hk_3 + \frac{16}{15}hk_4\right).
 \end{aligned}$$

Daraus kann eine erste Schätzung für den Punkt am Schrittende gemacht werden:

$$u_{j+1} = u_j + h \cdot \left(\frac{1}{9}k_1 + \frac{9}{20}k_3 + \frac{16}{45}k_4 + \frac{1}{12}k_5 \right).$$

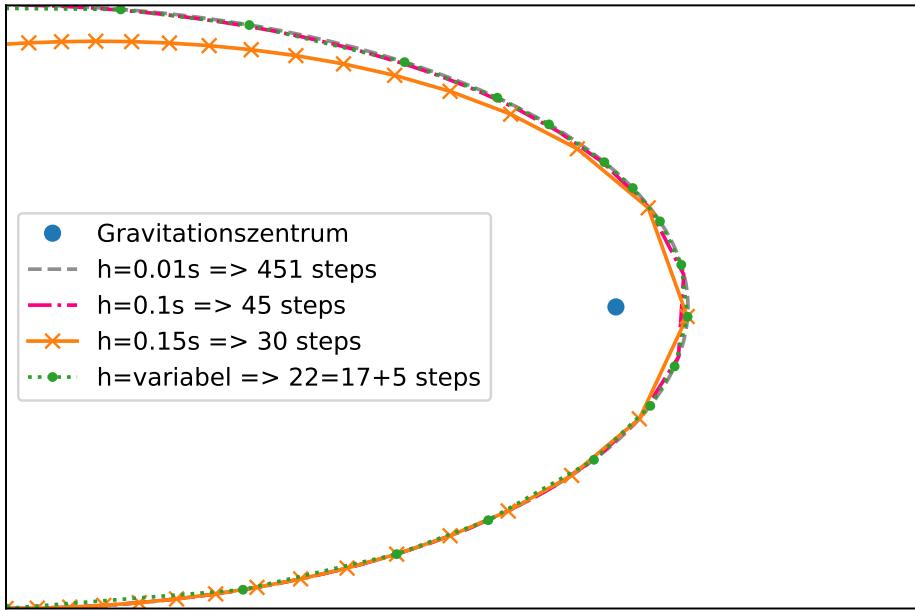


Abbildung 13.5: Bahn eines Teilchens um ein Gravitationszentrum: Vergleich von unterschiedlichen fixen Schrittweiten mit der variablen Schrittweite mittels Fehlerschätzung nach Fehlberg und Steueralgorithmen nach Abbildung 13.4, wiederholte Schritte beim Verkleinern der Schrittweite mitberücksichtigt

Mit einem weiteren Koeffizienten

$$k_6 = f\left(x_j + \frac{5}{6}h, u_j + \frac{65}{432}hk_1 - \frac{5}{16}hk_2 + \frac{13}{16}hk_3 + \frac{4}{27}hk_4 + \frac{5}{144}hk_5\right)$$

lässt sich nun eine zweite Schätzung für den Endpunkt berechnen:

$$\hat{u}_{j+1} = u_j + h \cdot \left(\frac{47}{450}hk_1 + \frac{12}{25}hk_3 + \frac{32}{225}hk_4 + \frac{1}{30}hk_5 + \frac{6}{25}hk_6 \right).$$

Der lokale Fehler des Schrittes lässt sich nun mit

$$T(x_{j+1}, h) = |u_{j+1} - \hat{u}_{j+1}| = \frac{h}{300} \cdot |-2k_1 + 9k_3 - 64k_4 - 15k_5 + 72k_6|$$

schätzen. Da beide Schätzungen eine Linearkombination der Koeffizienten k_1 bis k_6 sind, lässt sich die Differenz auch direkt aus den Linearkombinationen ermitteln, sodass der zweite Schätzwert \hat{u}_{j+1} überflüssig wird.

Mit dieser Fehlerschätzung lässt sich nun ein Algorithmus wie in Abbildung 13.4 steuern, welcher die Schrittgröße kontrolliert. Falls die Schätzung des Fehlers $T(x_{j+1}, h_j)$ grösser ist als die erlaubte Toleranz ε , wird die Schrittweite halbiert und der Schritt wiederholt. Falls der Fehler viel

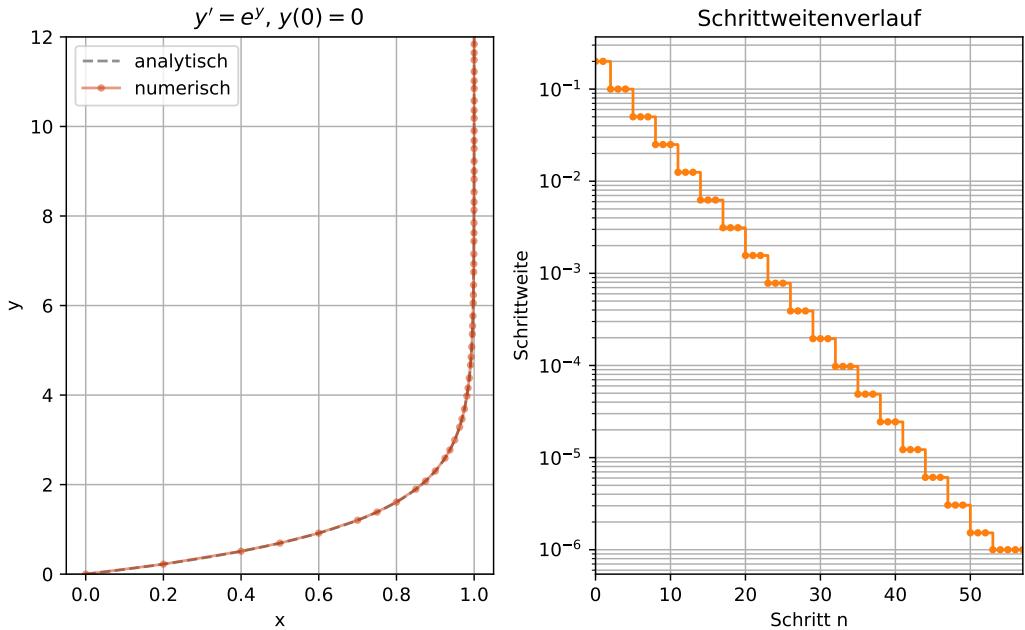


Abbildung 13.6: Visualisierung des Vorteils einer Schrittweitensteuerung anhand der Berechnung einer Annäherung für $y(1 - \varepsilon)$ mit der Gleichung $y' = e^y$ mit $y(0) = 0$. ε ist > 0 , beispielsweise $\varepsilon = 10^{-3}$. Links: numerische Berechnung der Lösung der DGL $y' = e^y$, jeder Schritt mit einem Punkt hervorgehoben. Rechts: Schrittweite (in x -Richtung) während der Berechnung von $y' = e^y$. Aufgrund der hohen Steilheit in der Nähe von $x = 1$ wird dort eine massiv kleinere Schrittweite benötigt als in der Umgebung von $x = 0$. Durch die hier angewendete Schrittweitensteuerung lässt sich die benötigte Anzahl von Schritten und somit der Rechenaufwand merklich reduzieren.

kleiner ist als die erlaubte Toleranz, wird die Schrittweite für den nächsten Schritt verdoppelt. Im Beispiel in Abbildung 13.5 lässt sich somit die nötige Anzahl Schritte im Vergleich zu einer ähnlich genauen, fixen Schrittweite etwa halbieren.

13.4 Folgerungen

Schrittweitensteuerungen existieren in diversen Variationen, doch alle haben das selbe Ziel, nämlich das gesuchte Resultat mit möglichst wenig Schritten und mit der erforderlichen Genauigkeit zu berechnen. Nur dank der Schrittängensteuerung ist es überhaupt möglich, bei Simulationsprogrammen wie Spice (Simulator für elektronische Schaltungen) innerhalb nützlicher Zeit ein Ergebnis zu haben. Zur Illustration des Vorteils einer Schrittweitensteuerung dient der Graph 13.6. Dabei wird eine Annäherung an $y(1 - \varepsilon)$ der bereits in Übungsaufgabe 5.2 studierten Differenzialgleichung $y' = e^y$ mit $y(0) = 0$ berechnet. Der Algorithmus reduziert die Schrittweite über sechs Größenordnungen.

Doch auch die Verwendung der Schrittweitensteuerung löst nicht alle Probleme. So kann es beispielsweise vorkommen, dass kleine Änderungen (relativ zur aktuellen Schrittweite) übersehen werden. Als Beispiel sei hier die Berechnung der Bahn eines Kometen erwähnt, welcher mit grosser

Geschwindigkeit am Mars vorbeifliegt, wobei dann aufgrund des Übersehens des Gravitationsfeldes des Mars eine falsche Bahn berechnet wird. Gegen dieses Phänomen lässt sich bei den meisten Applikationen eine maximale Schrittweite festlegen, doch dadurch steigt zwangsläufig auch der Rechenaufwand.

Literatur

- [1] Hans Rudolf Schwarz und Norbert Köckler. *Numerische Mathematik*. Vieweg+Teubner, 2011.
ISBN: 978-3-8348-1551-4.

14

Numerische Laplace-Inversion

Severin Weiss

Die Laplace-Transformation einer auf $[0, \infty)$ definierten und in jedem endlichen Intervall $[0, a)$ absolut integrierbaren Funktion, ist die Funktion

$$F(s) = \int_0^\infty e^{-st} f(t) dt, \quad Re(s) > \gamma_0. \quad (14.1)$$

Das Argument s kann eine komplexe Zahl sein, im Folgenden wird vorausgesetzt, dass das Integral für $Re(s) > \gamma_0$ konvergieren muss. Das inverse Laplace-Problem ist, die Funktionen $f(t)$ im Zeitbereich aus der bekannten Funktionen $F(s)$ im Frequenzbereich zu rekonstruieren.

Mit der Riemann-Inversionsformel ergibt sich $f(t)$ aus $F(s)$

$$f(t) = \frac{1}{2\pi i} \oint_B e^{st} F(s) ds, \quad t > 0, \quad (14.2)$$

wobei B die Bromwich-Kontur von $\gamma - i\infty$ bis $\gamma + i\infty$, mit $\gamma > \gamma_0$, parallel zur imaginären Achse ist. Die Bromwich-Kontur ist die geschlossene Kurve in Abbildung 14.1. Sie besteht aus zwei Teilen, aus dem zum einen der Kreis mit Radius R vom Mittelpunkt und zum anderen aus der vertikalen Linie γ , welche sich zur Rechten aller Singularitäten von $F(s)$ befindet. Bei Vergrößerung des Radius R wird die Bromwich-Kontur alle Singularitäten umschließen. Unter diesen Bedingungen konvergiert das Integral über dem Kreisbogen-Teil der Kontur gegen Null, wenn der Radius R gegen unendlich strebt. Damit man das Integral mit der Bromwich-Kontur numerisch berechnen kann, braucht man, dass $F(s)$, für alle $s \rightarrow \infty$ mit $Re(s) < \gamma_0$ genügend schnell gegen 0 gehen, weil der Umfang des Kreises anwächst.

Im Allgemeinen möchte man die Kontur zur linken Seite bewegen, sodass der Betrag des Faktors e^{st} im Integranden kleiner wird. Die Kontur sollte jedoch nicht zu nahe an Singularitäten von $F(s)$ angenähert werden, dies würde Spitzen des Integranden zur Folge haben, welche die numerische Berechnung erschweren.

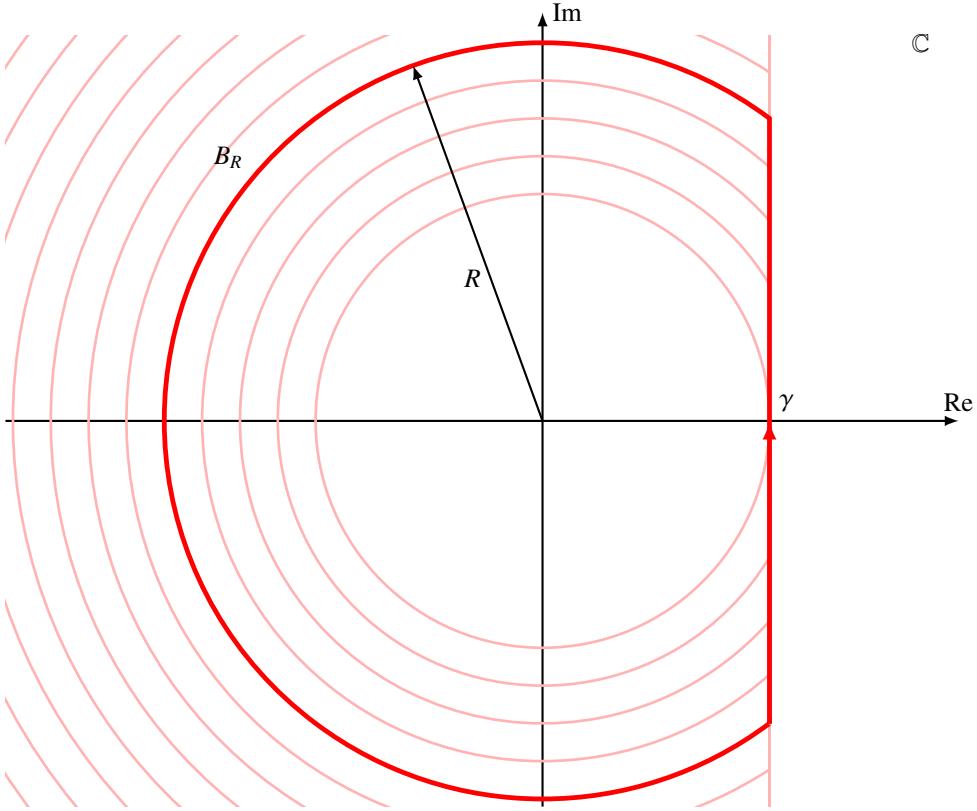


Abbildung 14.1: Die Bromwich-Kontur in der komplexen Ebene. Dargestellt ist die wachsende Figur abhängig vom Radius R .

14.1 Motivation: Lösung einer linearen Differentialgleichung

Wir wollen eine Lösung $f(t)$ der linearen Differentialgleichung

$$\frac{df}{dt} + \lambda f(t) = 0, \quad \text{mit} \quad f_0 = f(t=0). \quad (14.3)$$

Wir wenden die Laplace-Transformation auf die Gleichung (14.3) an. Mit der Laplace-Transformation folgt

$$(sF(s) - f_0) + \lambda F(s) = 0. \quad (14.4)$$

Die Gleichung (14.4) im Frequenzbereich kann nach $F(s)$ aufgelöst werden. Es folgt somit

$$F(s) = \frac{f_0}{s + \lambda}. \quad (14.5)$$

Um die Laplace-Inverse von $F(s)$ zu finden, existieren für gewisse Funktionstypen tabellierte zugehörige Rücktransformationen. Für die Funktion $F(s)$ in Gleichung (14.5) ergibt sich zum Beispiel

$$\mathcal{L}^{-1}\{F(s)\} = \mathcal{L}^{-1}\left\{\frac{f_0}{s + \lambda}\right\} = f_0 e^{-\lambda t}. \quad (14.6)$$

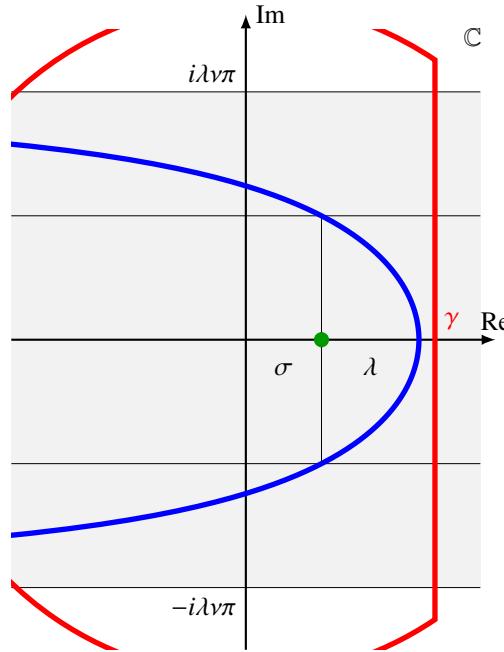


Abbildung 14.2: Illustration der Bromwich- (rot) und Talbot-Kontur (blau). Das (breite) Band $2i\lambda\mu\pi$ markiert den Bereich in dem die Singularitäten liegen können (zur linken der Talbot-Kontur).

Solche Tabellen sind jeweils in der Literatur zu finden, welche sich mit der Laplace-Transformation beschäftigt. Wenn für die gegebene Funktion $F(s)$ keine zugehörige Funktion tabelliert ist, muss das Integral (14.2) wie in der Einleitung beschrieben numerisch berechnet werden.

14.2 Lösung

Eine Bedingung, um eine passende Kontur zu finden, wurde von Talbot gefunden. Diese fordert, dass die Singularitäten s_j von $F(s)$ bekannt sind, sowie für $F(s)$ gelten muss, dass

1. $|F(s)| \rightarrow 0$ mit $|s| \rightarrow \infty$ in $\text{Re}(s) < \gamma_0$
2. Alle Singularitäten s_j liegen links von K , $|\text{Im}(s_j)| < K$, wobei der Wert von K bekannt ist.

Die erste Bedingung besagt, dass $F(s)$ für grosse $|s|$ schnell gegen 0 geht, solange man links von γ_0 ist. Dies hat zur Folge, dass Integrale über Konturen vernachlässigt werden können, wenn sie sich in diesem Gebiet bewegen. Das ist die entscheidende Idee von Talbot und ein Vorteil gegenüber der Bromwich-Kontur, da diese $F(s)$ für grosse $|s|$ nicht genügend schnell gegen 0 gehen. In der Konsequenz deformiert man die Bromwich-Kontur zur Talbot-Kontur hin. Solange bei der Deformation keine Singularitäten durchfahren werden, ist dies zulässig. Die beiden Abbildungen 14.2 und 14.3 veranschaulichen die Kontur von Talbot. Darin ist zu erkennen, dass $|s|$ im Verlaufe der Kontur grösser werden muss.

Die Parameter λ , σ und ν dienen dazu die Talbot-Kontur zu charakterisieren. Eine Erhöhung von σ verschiebt die Kontur nach rechts, während eine Erhöhung von ν Ausdehnung der Kontur in

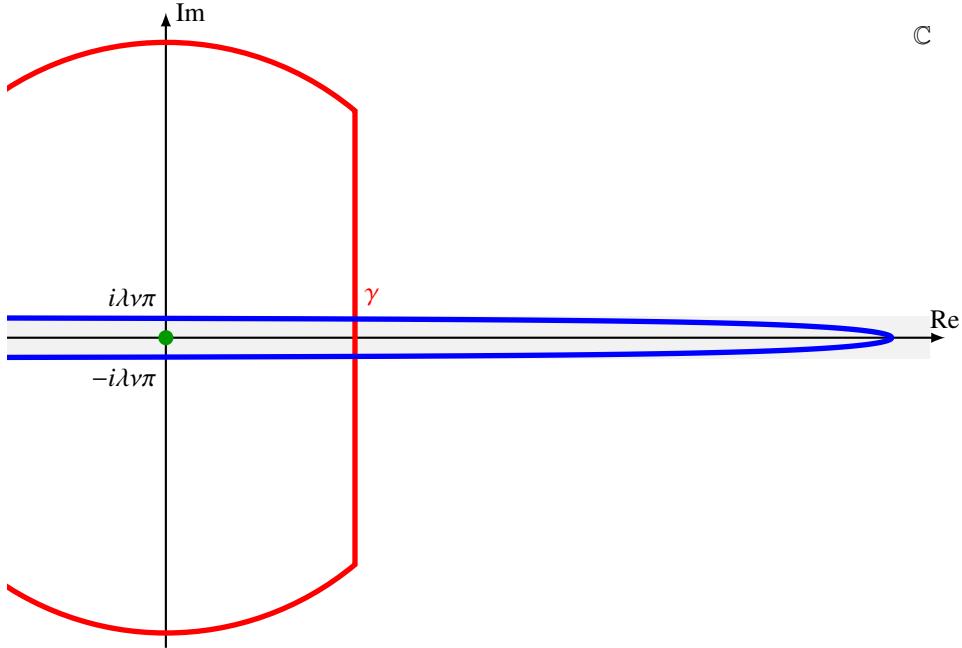


Abbildung 14.3: Illustration der Bromwich- (rot) und Talbot-Kontur (blau). Das (schmale) Band $2i\lambda\mu\pi$ markiert den Bereich in dem die Singularitäten liegen können (zur linken der Talbot-Kontur).

vertikaler Richtung zur Folge hat. Dabei entsteht ein Band der Breite $2i\lambda\mu\pi$, welches symmetrisch zur reellen Achse ist. Das beschriebene Band und dessen Abhängigkeit von den Parametern ist zu erkennen in den Abbildungen 14.2 und 14.3. Die Parameter sollten so gewählt werden, dass die Singularitäten der Laplace-Transformation innerhalb der Talbot-Kontur, respektive innerhalb des Bandes liegen. Die Parametrisierung der Kontur ist gegeben durch

$$s(z) = \sigma + \lambda s_\nu(z), \quad z \in (-2\pi i, 2\pi i) \quad (14.7)$$

wobei

$$s_\nu(z) = \frac{z}{1 - e^{-z}} + \frac{z(\nu - 1)}{2}. \quad (14.8)$$

Wir nutzen die Parametrisierung $z = 2i\theta$ und setzen diese in die Gleichungen (14.7) und (14.8) ein. Es folgt

$$s(\theta) = \sigma + \lambda s_\nu(\theta), \quad \theta \in (-\pi, \pi) \quad (14.9)$$

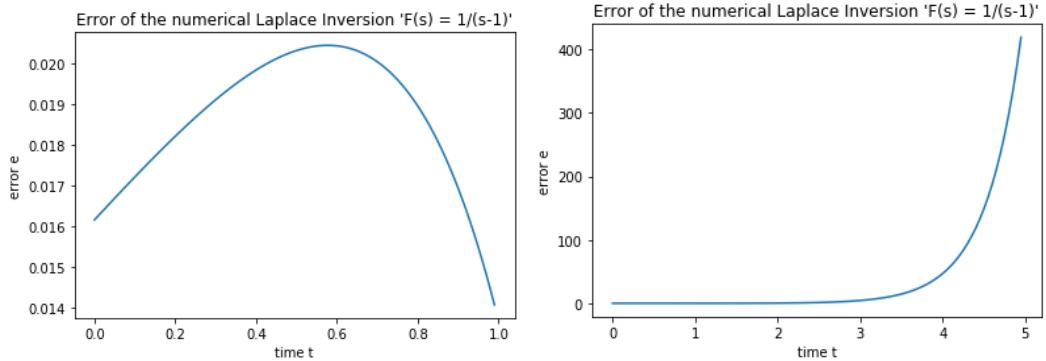
wobei

$$s_\nu(\theta) = \theta \cot \theta + i\nu\theta. \quad (14.10)$$

Die Parametrisierung $z = 2i\theta$ wird verwendet, weil sowohl Gleichung (14.7) als auch (14.8) rein komplex sind. Daher verwendet man θ um die Formeln in Real- und Imaginärteil aufteilen zu können.

Das Riemannsche Inversionsintegral nimmt die Form

$$f(t) = \frac{\lambda e^\sigma}{2\pi i} \int_{-\pi}^{\pi} e^{-\lambda t s_\nu(\theta)} F[\sigma + \lambda s_\nu(\theta)] s'_\nu(\theta) d\theta \quad (14.11)$$

Abbildung 14.4: Fehlerentwicklung von $F_1(s)$

an, wobei

$$s'_v(\theta) = i \left\{ v + \frac{\theta - \cos(\theta) \sin(\theta)}{\sin^2(\theta)} \right\}. \quad (14.12)$$

Nun folgt die eigentliche numerische Berechnung des Integrals (14.2). Die folgende Berechnung stammt aus [1]. Unter Berücksichtigung der Symmetrie, sowie der Trapezregel (für detaillierte Informationen siehe Kapitel 4) erhält man die Auswertung des Integrals

$$\tilde{f}(t) = \frac{\lambda e^{\sigma t}}{n} T_n(t) \quad (14.13)$$

wobei

$$T_n(t) = \sum_{j=0}^{n''} e^{\lambda s_v(\theta_j)} F[\sigma + \lambda s_v(\theta_j)] \frac{1}{i} s'_v(\theta_j) \quad (14.14)$$

$$\theta_j = j \frac{\pi}{n}. \quad (14.15)$$

Die Notation $\sum_{j=0}^{n''}$ besagt, dass der erste und letzte Term der Summe halbiert werden. In diesem Falle wird der Term $j = n$ gleich 0 und der $j = 0$ Term wird zu

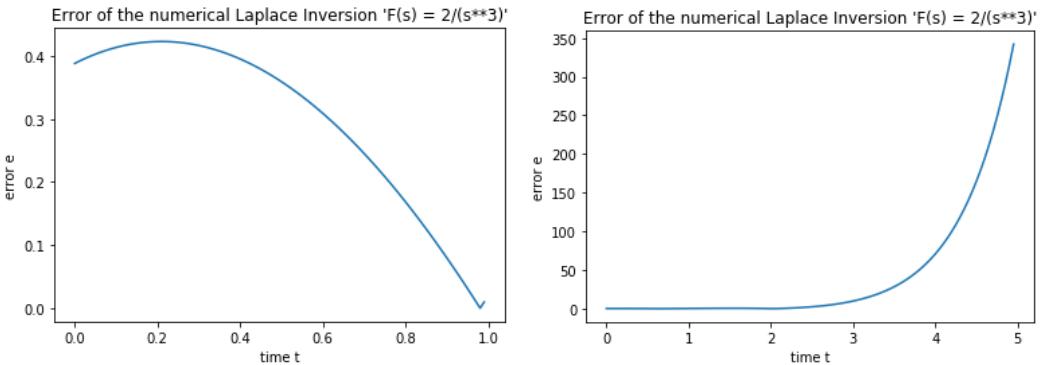
$$\frac{v}{2} e^{\lambda t} F(\sigma + \lambda). \quad (14.16)$$

Die beschriebene Methode wurde in Python implementiert und diente dazu, numerische Experimente durchzuführen. Der Code ist am Ende in Abschnitt 14.4 ersichtlich.

14.3 Folgerungen

Damit die Approximation von $\tilde{f}(t)$ möglichst genau $f(t)$ repräsentiert, müssen die passenden Werte für λ, σ und v gefunden werden. Dies geschah in diesem Falle rein iterativ durch Probieren. Insbesondere wurden folgende Funktionen ausgewertet $F_1(s) = \frac{1}{1-s}$ und $F_2(s) = \frac{2}{s^3}$. Die Werte, welche für die Auswertung verwendet wurden, sind in der Tabelle 14.1 abgebildet.

Es ist deutlich ersichtlich, dass der Fehler nur für ein gewisses Zeitintervall akzeptabel ist. Im Bereich, wo der Fehler sich in einem tolerierbaren Bereich befindet, wurden die Parameter λ, σ und v

Abbildung 14.5: Fehlerentwicklung von $F_2(s)$

	$F_1(s)$	$F_2(s)$
Parameter	$\lambda = 1.288$ $\sigma = 1.000001$ $\nu = 0.81$	$\lambda = 0.101$ $\sigma = 0.965$ $\nu = 0.098953$
t_0	1	1
Iterationszahl n	63000	100000
Fehler	$1.7076 \cdot 10^{-8}$	$1.8447 \cdot 10^{-6}$

Tabelle 14.1: Fehler zum Zeitpunkt t_0 mit iterativ optimierten Talbot-Parametern λ, σ und μ

für ein bestimmtes t_0 ermittelt. Diese Parameter sind für spätere Zeitpunkte nicht mehr sinnvoll (der Fehler wird sehr gross). Zu späteren Zeitpunkten müssten die Parameter auf ein Neues iterativ ermittelt werden. Der Versuch die optimierten Parameter durch einen Algorithmus zu finden, ist in [1] für Interssierte nachzuschlagen. Für den Zeitpunkt $t_0 = 1$ wurden die Fehler in der Größenordnung 10^{-8} respektive 10^{-6} erreicht. Dies geschah mittels Erhöhung der Iterationszahl n . Um noch bessere Resultate zu erhalten, könnte die Romberg-Beschleunigung verwendet werden (für detaillierte Information siehe Kapitel 4.2).

14.4 Programmcode

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Apr 30 08:34:15 2020
4
5 @author: Severin Weiss
6 """
7 import math
8 import numpy as np
9 import sympy as sym
10 from sympy import *
11 from sympy.integrals import laplace_transform
12 import matplotlib.pyplot as plt
13
14 t = symbols('t')

```

```

15 | s = symbols('s')
16 |
17 |
18 |
19 | class Talbot:
20 |     def __init__(self, lamda, sigma, mu, n):
21 |         self.lamda = lamda
22 |         self.sigma = sigma
23 |         self.mu = mu
24 |         self.n = n
25 |
26 |
27 |     def fTilde(self, Tn, Tn0):
28 |         """ Calculates fTilde
29 |         The first element fTilde[0] is calculated separately by using
30 |         Tn0.
31 |         """
32 |
33 |         sizeofTn = len(Tn)
34 |
35 |         fTilde = [complex(0) for x in range(0,sizeofTn)]
36 |
37 |         fTilde[0] = self.lamda*np.e**(t*self.sigma)/self.n*Tn0
38 |
39 |         for r in range(1,sizeofTn):
40 |             fTilde[r] = self.lamda*np.e**(t*self.sigma)/self.n*Tn[r]
41 |
42 |         return fTilde
43 |
44 |     def Calculate_Tn_Parameters(self):
45 |
46 |         """ Calculates the Tn Summation.
47 |         Tn will be a vector with n elements """
48 |
49 |         # Initialize s_mu, F_tilde. s_mu_prime, Tn with zeros
50 |
51 |         s_mu = s_mu_prime = [complex(0) for x in range(0,self.n)]
52 |         theta_j = [complex(0) for x in range(0,self.n)]
53 |
54 |         for k in range(0,self.n):
55 |
56 |             # Calculate theta_j
57 |             theta_j[k] = k*np.pi/self.n
58 |
59 |             # Calculate s_mu
60 |             s_mu[k] = theta_j[k]*(1/np.tan(theta_j[k])) + 1j*self.mu*
61 |                         theta_j[k]
62 |
63 |             # Calculate s_mu_prime
64 |             s_mu_prime[k] = 1j*(self.mu + (theta_j[k] - np.cos(theta_j[
65 |                           k])*np.sin(theta_j[k]))/np.square(np.sin(theta_j[k])))
66 |
67 |
68 |         return theta_j, s_mu, s_mu_prime
69 |
70 |     def Calculate_Tn(self,F):
71 |
72 |         theta_j, s_mu, s_mu_prime = self.Calculate_Tn_Parameters()

```

```

71
72     Tn0 = 0.5*self.mu*np.e**(t*self.lamda)*F[0]
73     Tn = [complex(0) for x in range(0, self.n-1)]
74
75     for j in range(1, len(F)-1):
76         Tn[j] = np.e**((t*self.lamda*s_mu[j])*F[j]*1/1j*s_mu_prime[j]
77
78     return Tn, Tn0
79
80
81 --- Definition of Functions
82
83 def evaluate_fTilde_at_t_zero(fTilde, t_zero):
84
85     evaluated_fTilde = [complex(0) for x in range(0, len(fTilde))]
86
87     for i in range(0, len(fTilde)):
88
89         evaluated_fTilde[i] = fTilde[i].evalf(subs={t:t_zero})
90
91
92     return evaluated_fTilde
93
94
95 def calculate_absolute_error_at_tzero(fExact, fTilde, t_zero):
96
97     evaluated_fTilde = evaluate_fTilde_at_t_zero(fTilde, t_zero)
98
99     magnitude_evaluated_fTilde = 0
100
101    for i in range(0, len(evaluated_fTilde)):
102
103        magnitude_evaluated_fTilde = abs(evaluated_fTilde[i]) +
104            magnitude_evaluated_fTilde
105
Severin Weiss
106    absolute_error_at_tzero = abs(fExact.evalf(subs={t:t_zero}) -
107        magnitude_evaluated_fTilde)
108
109
110
111 def calculate_absolute_error_over_time(fExact, fTilde, t_intervall):
112
113     # t_intervall contains all time_steps to evaluate
114
115     absolute_error_over_time = [float(0) for i in range(0, len(
116         t_intervall))]
117
118     for j in range(0, len(t_intervall)):
119
120         evaluated_fTilde = evaluate_fTilde_at_t_zero(fTilde, t_intervall
121             [j])
122
123         magnitude_evaluated_fTilde = 0
124
125         for i in range(0, len(evaluated_fTilde)):
```

```

125         magnitude_evaluated_fTilde = abs(evaluated_fTilde[i]) +
126         magnitude_evaluated_fTilde
127
128     absolute_error_over_time[j] = abs(fExact.evalf(subs={t:
129         time_intervall[j]})) - magnitude_evaluated_fTilde)
130
131 return absolute_error_over_time
132
133
134 def plot_absolute_error_over_time(fExact, fTilde, time_intervall,
135     nsteps):
136
137     # time_intervall must be a list with starting time t_a and ending
138     # time t_b --> [t_a, t_b]
139
140     intervall = time_intervall[1] - time_intervall[0]
141
142     timestep = intervall/nsteps
143
144     timevector = [float(0) for i in range(0,nsteps)]
145
146     time = time_intervall[0]
147
148     for i in range(0,nsteps):
149
150         time = time + timestep
151
152         timevector[i] = time + timestep
153
154     absolute_error_over_time = calculate_absolute_error_over_time(
155         f_of_t_exact, fTilde, timevector)
156
157     xvals = np.arange(time_intervall[0], time_intervall[1], timestep)
158     yvals = absolute_error_over_time
159
160     plt.figure()
161     plt.plot(xvals, yvals,label='error')
162     plt.title("Error of the numerical Laplace Inversion 'F(s) = 2/(s
163         **3)'")
164     plt.xlabel("time t")
165     plt.ylabel("error e")
166     plt.show
167
168
169 def plot_absolute_error_at_certain_points(fExact, fTilde, timevector):
170
171     # time_vector must be a list with all points in time to evaluate
172
173     absolute_error_at_certain_points =
174         calculate_absolute_error_over_time(f_of_t_exact, fTilde,
175             timevector)
176
177     xvals = timevector
178     yvals = absolute_error_at_certain_points
179
180     plt.figure()
181     plt.plot(xvals, yvals, label='error')
182     plt.title("Error of the numerical Laplace Inversion 'F(s) = 2/(s
183         **3)'")

```

```
175     plt.xlabel("time t")
176     plt.ylabel("error e")
177     plt.show()
178     plt.legend()
179
180
181 def table_absolute_error_at_certain_points(fExact, fTilde, timevector):
182
183     # time_vector must be a list with all points in time to evaluate
184
185     absolute_error_at_certain_points =
186         calculate_absolute_error_over_time(f_of_t_exact, fTilde,
187             timevector)
188
189     data = [[] for x in range(0, len(timevector))]
190
191     fig, axs = plt.subplots(2, 1)
192
193     if len(timevector) == len(absolute_error_at_certain_points):
194         for i in range(0, len(timevector)):
195             data[i] = [timevector[i], absolute_error_at_certain_points[
196                 i]]
197
198     print(data)
199
200     clust_data = data
201     collabel = ("time", "Error")
202     axs[0].axis('tight')
203     axs[0].axis('off')
204     the_table = axs[0].table(cellText = clust_data, colLabels=collabel,
205                             loc='center')
206
207     plt.show()
208
209
210 def laplace_transform_comparison(fTilde, F_of_s):
211
212     frequency_vector = [0.1, 1, 10, 100]
213     laplace_transform_fTilde = [float(0) for x in range(0, len(fTilde))]
214
215     for i in range(0, len(fTilde)):
216
217         laplace_transform_fTilde[i] = laplace_transform(fTilde[i], t, s
218             )
219
220
221     print(laplace_transform_fTilde[0])
222     print(laplace_transform_fTilde[0].evalf(subs={s:1}))
223
224     difference = [float(0) for x in range(0, len(frequency_vector))]
225
226     for i in range(0, len(difference)):
227
228         difference[i] = F_of_s.evalf(subs={s:frequency_vector[i]})-
229             summe_laplace_transform_fTilde.evalf(subs={s:frequency_vector[i]})
```

```

227 # 
228 #
229 #      return None
230
231
232 --- Defining Constants
233
234 # Define lamda
235 LAMDA = 0.101
236 # Define sigma
237 SIGMA = 0.965
238 # Define mu
239 MU = 0.098953
240
241 # Define n
242 N = 100
243 # Time to evaluate
244 T_ZERO = 1
245
246
247 --- Defining mathematical Functions for the Inversion and Evaluation
248
249 f_of_t_exact = t**2
250
251 F_of_s_exact = 2/(s**3)
252
253 # Initializing an Object (Talbot1) of the Class Talbot
254 Talbot1 = Talbot(LAMDA,SIGMA,MU,N)
255 theta_j, s_mu, s_mu_prime = Talbot1.Calculate_Tn_Parameters()
256
257 # Now the Function F(s) has to be modified by setting s = sigma + lamda
258     *s_mu
259 # Example for F(s) = 1/(s+1)
260
261 F = [complex(0) for x in range(0,len(theta_j))]
262
263 for i in range(1,len(s_mu)): # "-1" because of the first element in Tn
264     substition_of_s = Talbot1.sigma + Talbot1.lamda*s_mu[i]
265     F[0] = F_of_s_exact.evalf(subs={s:(SIGMA+LAMDA)})
266     F[i] = F_of_s_exact.evalf(subs={s:substition_of_s})
267
268
269 # Calculate numeric Approximation of the Laplace Inversion
270 Tn, Tn0 = Talbot1.Calculate_Tn(F)
271 fTilde = Talbot1.fTilde(Tn,Tn0)
272
273
274 # Calculate the absolute Error evaluated at
275 # certain t
276 # Plot of the error dependent on t ?
277
278 #error=calculate_absolute_error_at_tzero(f_of_t_exact, fTilde, 0)
279 #print(error)
280
281 #time_vector = [0, 0.1, 0.2, 0.3, 0.5,
282 #                0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 2, 3]
282 #
283 #plot_absolute_error_at_certain_points(f_of_t_exact, fTilde,

```

```
284     time_vector)
285 time_intervall = [0,1]
286
287 plot_absolute_error_over_time(f_of_t_exact, fTilde, time_intervall,
288                               100)
289 #laplace_transform_comparison(fTilde, F_of_s_exact)
290
291
292
293 # show dependency of Paramters and poles of F(s)
294
```

Literatur

- [1] A. Murli und M. Rizzardi. “Talbot’s Method for the Laplace Inversion”. In: *ACM Transactions on Mathematical Software* 47 (1990), S. 158–168. URL: https://www.academia.edu/19873877/Algorithm_682_Talbots_method_of_the_Laplace_inversion_problems.

15

Störungstheorie

Daniel Bucher und Thomas Kistler

15.1 Einleitung

Die Störungstheorie befasst sich mit den Effekten von kleinen Einflüssen auf sich bewegende Objekte. Oft werden diese weggelassen, um die Probleme in geschlossener Form überhaupt lösen zu können. Wenn es jedoch um die Berechnung von Satellitenbahnen geht, wo die Planeten eine kleine Anziehung auf die Satelliten ausüben, kann dies langfristig zu grossen Abweichungen führen. Auch bei Raketen gibt es Störungen der Bahn. So wird diese unter anderem vom Luftdruck beeinflusst, der von der Höhe abhängig ist.

In der Störungstheorie wird mit einer vereinfachten Form des ursprünglichen Problems, welches sich exakt lösen lässt, gestartet. Die Konstanten der Gleichung werden danach langsam angepasst, um die Lösung des Problems zu approximieren. Dieses Vorgehen beschreiben wir anhand eines einfachen Beispiels in den nächsten Abschnitten.

15.2 Problemstellung

Eine Kanonenkugel wird in einem 45° Winkel abgefeuert. Sie besitzt eine Startgeschwindigkeit von 100 m/s in x sowie y -Richtung. Die Aufgabe besteht darin, die Position zum Zeitpunkt t zu berechnen.

Dies ist ein gängiges Beispiel für den schießen Wurf. In der Regel werden dabei folgende Formeln verwendet:

$$\begin{aligned}x(t) &= x_0 + v_{0x}t \\y(t) &= y_0 + v_{0y}t - \frac{1}{2}gt^2.\end{aligned}\tag{15.1}$$

Nun möchten wir jedoch ein genaueres Ergebnis erzielen und den Luftwiderstand mitberücksichtigen. Der Luftwiderstand F_w ist proportional zum Quadrat der Geschwindigkeit. Es gilt folgende

Formel:

$$F_W = \underbrace{c_W A \frac{1}{2} \rho_{\text{Luft}} v^2}_k.$$

c_W bezeichnet den Strömungswiderstand, A die Widerstandsfläche und ρ_{Luft} die Luftdichte. Um die Berechnung simpel zu gestalten, gehen wir von einer konstanten Luftdichte ρ_{Luft} aus und können so den Luftwiderstand zusammen fassen als $F_W = k \cdot v^2$.

Formt man die Formel weiter um, erhält man das folgende Resultat:

$$F_W = k \cdot |\vec{v}| \cdot \vec{v} = k \cdot \sqrt{v_x^2 + v_y^2} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}.$$

Die ganze Kraft, die auf die Kugel wirkt, ist gemäss Newtons zweitem Axiom $F = m \cdot a$. Neben dem Luftwiderstand wirkt in y -Richtung zusätzlich die Gravitationskraft $F_G = m \cdot g$. Somit lassen sich folgende Gleichungen aufstellen:

$$\begin{aligned} m \cdot a_x &= k \cdot \sqrt{v_x^2 + v_y^2} \cdot v_x \\ m \cdot a_y &= k \cdot \sqrt{v_x^2 + v_y^2} \cdot v_y - m \cdot g. \end{aligned}$$

Dividiert durch m ergibt dies:

$$\begin{aligned} a_x &= \frac{k}{m} \cdot \sqrt{v_x^2 + v_y^2} \cdot v_x \\ a_y &= \frac{k}{m} \cdot \sqrt{v_x^2 + v_y^2} \cdot v_y - g. \end{aligned}$$

Die Beschleunigung ist nicht bekannt. Sie kann jedoch als zweite Ableitung der Position nach der Zeit ausgedrückt werden. Die Geschwindigkeit entspricht der ersten Ableitung. Somit ergibt sich das folgende System von Differentialgleichungen zweiter Ordnung:

$$\begin{aligned} \ddot{r}_x &= \frac{k}{m} \cdot \sqrt{\dot{r}_x^2 + \dot{r}_y^2} \cdot \dot{r}_x \\ \ddot{r}_y &= \frac{k}{m} \cdot \sqrt{\dot{r}_x^2 + \dot{r}_y^2} \cdot \dot{r}_y - g. \end{aligned} \tag{15.2}$$

15.3 Numerische Lösung

Die Gleichungen (15.2) lassen sich durch numerische Verfahren lösen. Wenn man sich nun jedoch einen Satelliten oder eine Rakete vorstellt, haben diese nur eine begrenzte Rechenkapazität. Aus diesem Grund wird die komplexere Berechnung durch eine Bodenstation durchgeführt. Die Resultate der Berechnung werden jeweils dem Flugobjekt mitgeteilt, sodass dieses mit einfacheren Formeln seine Bahn für die nahe Zukunft annähernd abschätzen kann. Dieses Kapitel beschäftigt sich mit der genaueren, komplexeren Lösung des Problems, welche durch die Bodenstation übernommen wird.

Da das Runge-Kutta-Verfahren nur mit Differentialgleichungen erster Ordnung umgehen kann, muss die Ordnung des Gleichungssystems (15.2) zuerst um eins reduziert werden. Dies kann erreicht werden, indem zwei Hilfsvariablen $v_x = \dot{x}$ und $v_y = \dot{y}$ eingeführt werden. Das System wird somit um

```

1 || def dgl(t, y):
2 ||     '''The differential equation to solve.'''
3 ||     r_x = y[0]
4 ||     r_y = y[1]
5 ||     v_x = y[2]
6 ||     v_y = y[3]
7 ||     p0 = v_x
8 ||     p1 = v_y
9 ||     p2 = -k/m * np.sqrt(v_x**2 + v_y**2) * v_x
10 ||    p3 = -k/m * np.sqrt(v_x**2 + v_y**2) * v_y - g
11 ||    return [p0, p1, p2, p3]
12
13 def runge_kutta(t):
14     '''Runs Runge Kutta to find position and speed at time t.'''
15     rk = sp.solve_ivp(fun=dgl, t_span=(0, t),
16     y0 = [r0x, r0y, v0x, v0y], method='RK45', vectorized=True)
17     assert rk.status == 0
18     time_at_end = rk.t[-1]
19     position_at_end = rk.y[0:2, -1]
20     speed_at_end = rk.y[2:4, -1]
21     return time_at_end, position_at_end, speed_at_end

```

Listing 15.1: Programm zur Berechnung des Differentialgleichungssystems

eine Ordnung reduziert, erhält aber im Austausch zusätzliche Dimensionen:

$$\begin{aligned}\dot{r}_x &= v_x \\ \dot{r}_y &= v_y \\ \dot{v}_x &= \frac{k}{m} \cdot \sqrt{v_x^2 + v_y^2} \cdot v_x \\ \dot{v}_y &= \frac{k}{m} \cdot \sqrt{v_x^2 + v_y^2} \cdot v_y - g.\end{aligned}$$

Als Vektor formuliert:

$$\frac{d}{dt} \begin{pmatrix} r_x \\ r_y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ \frac{k}{m} \cdot \sqrt{v_x^2 + v_y^2} \cdot v_x \\ \frac{k}{m} \cdot \sqrt{v_x^2 + v_y^2} \cdot v_y - g \end{pmatrix}.$$

Damit haben wir eine mit dem Runge-Kutta-Verfahren lösbar Variante der Differentialgleichung gefunden. Der Python Code in Listing 15.1 löst die Differentialgleichung mit Hilfe der Library *SciPy*. Die Bodenstation, die zu komplexen Berechnungen in der Lage ist, führt also in einem regelmässigen Zeitintervall die Funktion `runge_kutta(t)` auf dem Differentialgleichungssystem aus und erhält so die Position und die Geschwindigkeit zu einem bestimmten Zeitpunkt.

15.4 Erster Lösungsansatz

Die Formeln (15.1) sind ohne viel Rechenleistung berechenbar und somit auch als Flugobjekt durchführbar. Die Störungstheorie befasst sich damit, die Anfangswerte x_0 , y_0 , v_{0x} und v_{0y} so abzuändern,

dass dennoch gute Approximationen für $x(t)$ und $y(t)$ gefunden werden können. Hierbei ist es die Aufgabe der Bodenstation, die notwendigen Startparameter dem Flugobjekt mitzuteilen, sodass diese die Berechnung auf Basis der trivialen Formeln 15.1 vornehmen kann.

Als ersten Lösungsansatz haben wir v_{0x} und v_{0y} durch lineare Funktionen abhängig von der Zeit ersetzt, aber x_0 und y_0 vorerst unberücksichtigt gelassen. Hintergrund dieser Überlegung ist, dass der Luftwiderstand nur die Geschwindigkeit beeinflusst:

$$\begin{aligned} v_{0x} &\rightarrow v_{0x}(t) = v_{0x} + \phi_x \cdot t \\ v_{0y} &\rightarrow v_{0y}(t) = v_{0y} + \phi_y \cdot t. \end{aligned}$$

Wir passen die Startgeschwindigkeit im Verlaufe der Zeit t also anhand eines unbekannten Faktors ϕ an. Eingesetzt in die Formeln (15.1) erhalten wir:

$$\begin{aligned} x(t) &= x_0 + t \cdot (v_{0x} + \phi_x \cdot t) \\ y(t) &= y_0 + t \cdot (v_{0y} + \phi_y \cdot t) - \frac{1}{2}gt^2, \end{aligned} \tag{15.3}$$

x_0 , y_0 , v_{0x} und v_{0y} sind die bekannten Anfangsbedingungen des Problems. Ebenfalls können $x(t)$ und $y(t)$ mit dem Runge-Kutta-Verfahren bestimmt werden. Um neue Werte mit der Störung zu bestimmen, müssen die Unbekannten ϕ_x und ϕ_y berechnet werden. Dies kann erreicht werden, indem man die Gleichungen (15.3) nach ϕ_x bzw. ϕ_y auflöst:

$$\begin{aligned} \phi_x &= \frac{x(t) - x_0 - tv_{0x}}{t^2} \\ \phi_y &= \frac{y(t) - y_0 - tv_{0y} + \frac{1}{2}gt^2}{t^2}. \end{aligned} \tag{15.4}$$

Beispielsweise kann dieses Gleichungssystem für $t = 5\text{s}$ gelöst werden. Dadurch erhält man ϕ_x und ϕ_y . Da hierzu allerdings $x(t = 5)$ und $y(t = 5)$ mittels Runge-Kutta berechnet werden müssen, ist dazu nur die Bodenstation in der Lage. Die Werte ϕ_x und ϕ_y können nun aber dem Flugobjekt übertragen werden, welches mit den Formeln (15.3) seine Position in naher Zukunft nun autonom approximieren kann, beispielsweise für das Intervall $t \in [5, 6]$. Anschliessend könnte die Bodenstation die Werte ϕ_x und ϕ_y für $t = 6\text{s}$ neu berechnen, um den Fehler nicht beliebig anwachsen zu lassen.

Das Resultat ist in Abbildung 15.1 dargestellt. Grün ist dabei die exakte Flugbahn gemäss den Differentialgleichungen (15.2), blau die linear berechnete Bahn ohne Berücksichtigung des Luftwiderstandes und rot die korrigierte Variante. Für die korrigierte Variante wurde eine Intervalllänge von $t = 1\text{s}$ gewählt. Das heisst, sie bestimmt die Unbekannten ϕ_x und ϕ_y zu jeder vollen Sekunde $t \in \{1, 2, 3, \dots, 25\}$ und nutzt jene Daten, um die Position jeweils für die folgende Sekunde vorherzusagen. Wie zu erkennen ist, kann ein Grossteil des Fehlers bereits vermieden werden.

Anstelle des linearen Ansatzes könnte man auch einen quadratischen Ansatz nutzen. Dies ergibt anstelle von (15.3):

$$\begin{aligned} x(t) &= x_0 + t \cdot (v_{0x} + \phi_{x_1}t + \phi_{x_2}t^2) &= x_0 + v_{0x}t + \phi_{x_1}t^2 + \phi_{x_2}t^3 \\ y(t) &= y_0 + t \cdot (v_{0y} + \phi_{y_1}t + \phi_{y_2}t^2) - \frac{1}{2}gt^2 &= y_0 + v_{0y}t + \phi_{y_1}t^2 + \phi_{y_2}t^3 - \frac{1}{2}gt^2. \end{aligned} \tag{15.5}$$

Damit haben wir noch bessere Resultate erhalten. Diese sind in Abbildung 15.2 ersichtlich. Von Auge ist bereits kein Unterschied der beiden Flugbahnen mehr erkennbar.

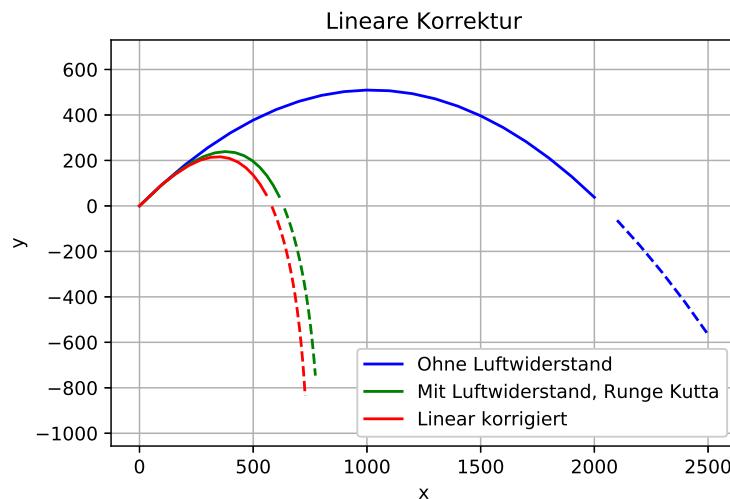


Abbildung 15.1: Linearer Korrekturterm

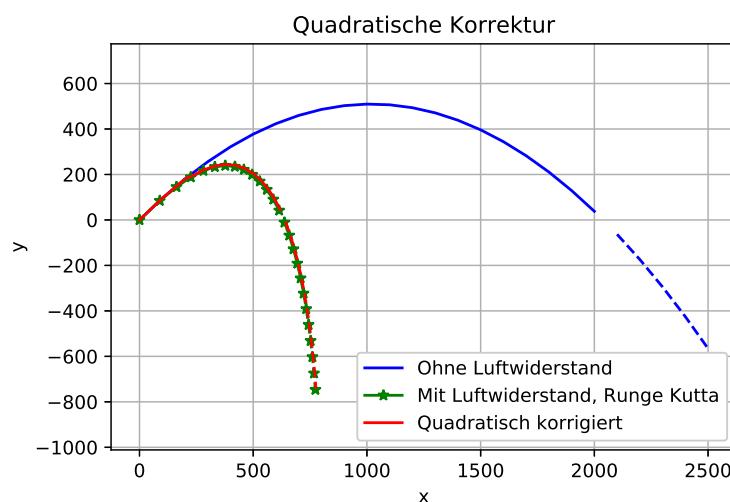


Abbildung 15.2: Quadratischer Korrekturterm

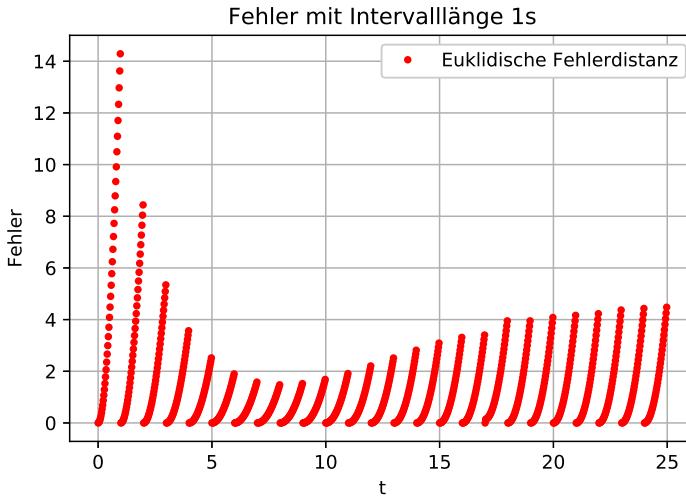


Abbildung 15.3: Fehler als euklidische Distanz

15.5 Verbesserte Lösung

Man erhält bessere Resultate, wenn man statt nur die Anfangswerte $\vec{v}_0 = (v_{0x}, v_{0y})$ auch $\vec{r}_0 = (r_{0x}, r_{0y})$ variabel gestaltet. Zudem kann man anstelle von t nur die Differenz Δt seit dem letzten Update von der Bodenstation betrachten.

Wiederum ist das Ziel, dass der Flugkörper mit den einfachen Formeln (15.1) eine möglichst gute Approximation auf einfache Art und Weise berechnen kann, indem er die Anfangswerte \vec{r}_0 und \vec{v}_0 im Laufe der Zeit gemäss Weisungen der Bodenstation anpasst.

In einem ersten Schritt verlangen wir von der Bodenstation, welche \vec{r} , aber auch \vec{v} mit dem Runge-Kutta-Verfahren berechnen kann, eben diese Werte um so eine Approximation mit den simplen Formeln (15.1) tätigen zu können. Es ergeben sich folgende Formeln:

$$\begin{aligned} r_x(t + \Delta t) &= r_{0x} + v_{0x}\Delta t \\ r_y(t + \Delta t) &= r_{0y} + v_{0y}\Delta t - \frac{1}{2}g\Delta t^2. \end{aligned} \tag{15.6}$$

Verlangt man von der Bodenstation jede Sekunde neue Werte für die Elemente r_{0x} , r_{0y} , v_{0x} und v_{0y} gibt dies eine Genauigkeit von 2 Stellen. In Abbildung 15.3 ist der Fehler ersichtlich. Dieser ist als euklidische Distanz zur effektiven Position dargestellt.

15.6 Möglichkeiten zur Genauigkeitssteigerung

In diesem Abschnitt diskutieren wir Verbesserungsmöglichkeiten, um den Fehler weiter zu minimieren. Abbildung 15.3 zeigt, dass der Fehler stark von der Intervalllänge abhängt. Eine Reduktion der Intervalllänge beschreiben wir im nächsten Abschnitt. Weiter lässt sich die Genauigkeit mit Hilfe der Störungstheorie höherer Ordnung weiter verbessern.

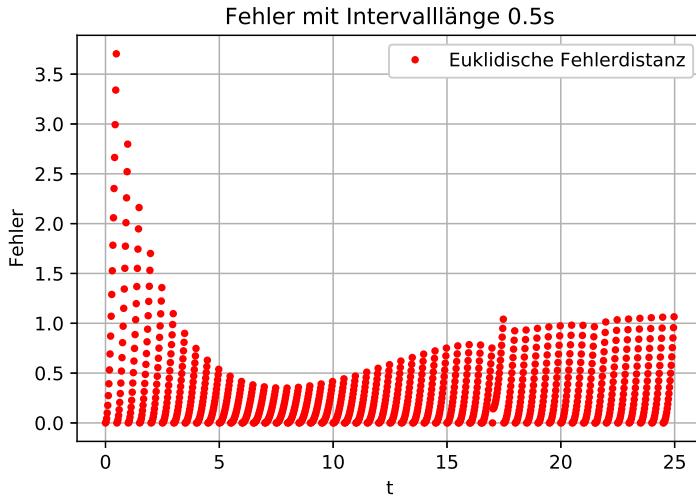


Abbildung 15.4: Fehler bei halbierter Intervalldauer

15.6.1 Reduktion der Intervalldänge

In Abbildung 15.3 ist der Fehler für die Intervalldänge von einer Sekunde ersichtlich. Wie man erkennen kann, nimmt dieser innerhalb eines solchen Intervalls stark zu. Dies ist vor allem unserem Beispiel geschuldet, da die Störung (in unserem Beispiel der Luftwiderstand) kurzfristig eine grosse Auswirkung auf das Resultat hat. Bei Berechnungen von z.B. Satellitenbahnen wäre dies anders, da Störungen, wie die Gravitation von Planeten, viel kleiner sind und die Abweichungen erst bei längeren Intervalldauern auftreten.

Für unser Beispiel haben wir in einem ersten Schritt die Intervalldänge halbiert. Daraus resultierte der Fehler in Abbildung 15.4. Wie ersichtlich ist, ist der Fehler um ca. Faktor vier kleiner. Weiter ist zum Zeitpunkt $t = 17\text{s}$ eine kleine numerische Instabilität beobachtbar.

In einem zweiten Schritt haben wir die Intervalldänge nochmals auf die Länge 0.25s halbiert. Dies ist in Abbildung 15.5 ersichtlich. Wir erhalten erneut einen Genauigkeitsgewinn um den Faktor vier bzw. zwei Bit. Dies lässt sich leider jedoch nicht weiter beliebig fortsetzen. Die Rechenlast des Satelliten bleibt konstant und ist nicht von der Intervalldänge abhängig. Der Aufwand für die Bodenstation nimmt jedoch zu. Die numerischen Berechnungen für aufwendige Probleme (wie z.B. Satellitenbahnen) benötigen auch für die Bodenstation viel Rechenleistung und können nicht beliebig oft ausgeführt werden.

15.6.2 Erhöhung der Ordnung

Eine andere Möglichkeit zur Steigerung der Genauigkeit eröffnet sich dadurch, dass man anstelle r_{0x}, r_{0y}, v_{0x} und v_{0y} , was an sich Polynome nullter Ordnung sind, Polynome höherer Ordnung ansetzt. Wir schauen uns ein Beispiel erster Ordnung an. Die Anfangsbedingungen werden folgendermassen geändert:

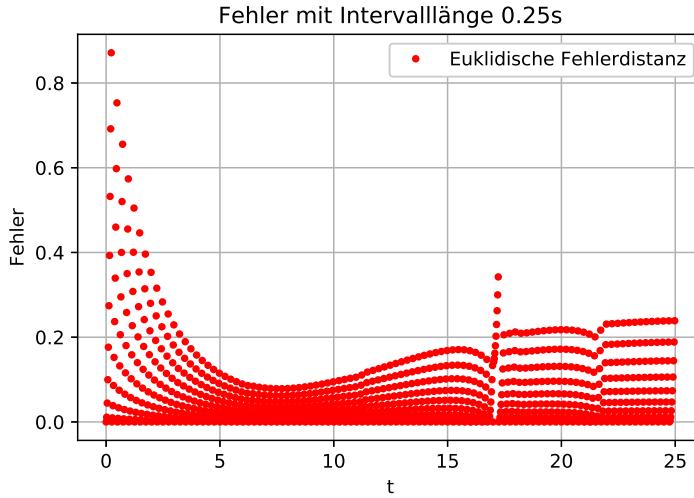


Abbildung 15.5: Fehler mit einem Viertel der ursprünglichen Intervalldauer

$$\begin{aligned}
 x_0 &\longrightarrow r_{x0} + r_{x1} \cdot \Delta t \\
 y_0 &\longrightarrow r_{y0} + r_{y1} \cdot \Delta t \\
 v_x &\longrightarrow v_{x0} + v_{x1} \cdot \Delta t \\
 v_y &\longrightarrow v_{y0} + v_{y1} \cdot \Delta t.
 \end{aligned} \tag{15.7}$$

Betrachten wir nun ein bestimmtes Intervall, etwa $t \in [5, 6)$. Innerhalb dieses Intervalls arbeiten wir mit $t = 5$, $\Delta t \in [0, 1)$. t wird also fixiert und Δt ist unsere neue Variable. So gilt nach Einsetzen in Gleichungen (15.1)

$$\begin{aligned}
 r_x(t + \Delta t) &= (r_{x0} + r_{x1} \cdot \Delta t) + (v_{x0} + v_{x1} \cdot \Delta t) \cdot \Delta t \\
 r_y(t + \Delta t) &= (r_{y0} + r_{y1} \cdot \Delta t) + (v_{y0} + v_{y1} \cdot \Delta t) \cdot \Delta t - \frac{1}{2}g\Delta t^2.
 \end{aligned} \tag{15.8}$$

Um das Problem lösen zu können, werden wir vorerst eine exklusive Betrachtung der Geschwindigkeit durchführen. Indem man Gleichung (15.1) nach t ableitet und anschliessend unseren Ansatz für die Anfangsbedingungen einsetzt, erhält man

$$\begin{aligned}
 v_x(t + \Delta t) &= (v_{x0} + v_{x1} \cdot \Delta t) \\
 v_y(t + \Delta t) &= (v_{y0} + v_{y1} \cdot \Delta t) - g\Delta t.
 \end{aligned} \tag{15.9}$$

Lösung des Problems mit linearer Interpolation

In obigen Gleichungen sind die acht Unbekannten r_{x0} , r_{x1} , r_{y0} , r_{y1} , v_{x0} , v_{x1} , v_{y0} und v_{y1} zu bestimmen. Zwar haben wir mit Gleichungen (15.8) und (15.9) nur vier Gleichungen zur Verfügung, indem wir aber Δt variieren, können wir die Anzahl Gleichungen beliebig erhöhen. Das Gleichungssystem

ist somit lösbar. Wir machen uns hierbei zu Nutze, dass die Bodenstation mit dem Runge-Kutta-Verfahren $\vec{r}(t + \Delta t)$ und $\vec{v}(t + \Delta t)$ liefern kann. Dies sind also bekannte Werte, unabhängig davon, wie Δt gesetzt wird.

Indem wir $\Delta t = 0$ setzen, erhalten wir auf relativ einfache Art und Weise all jene Unbekannten, die mit einer 0 indiziert sind. Wir nutzen hierbei Gleichung (15.8) zur Bestimmung von \vec{r}_0 , und Gleichung (15.9) zur Bestimmung von \vec{v}_0 .

$$\begin{aligned} r_{x0} &= r_x(t) \\ r_{y0} &= r_y(t) \\ v_{x0} &= v_x(t) \\ v_{y0} &= v_y(t). \end{aligned} \tag{15.10}$$

Um v_{x1} und v_{y1} zu bestimmen, können wir nun weiterhin Gleichung (15.9) verwenden, müssen aber das Δt anpassen. Für eine lineare Interpolation bietet sich $\Delta t = 1\text{s}$ an. Man könnte aber auch z.B. die halbe Intervalldauer $\Delta t = 0.5\text{s}$ wählen. Mit einer einfachen Umformung ergibt sich:

$$\begin{aligned} v_{x1} &= \frac{v_x(t + \Delta t) - v_{x0}}{\Delta t} \\ v_{y1} &= \frac{v_y(t + \Delta t) - v_{y0} + g \cdot t}{\Delta t}. \end{aligned} \tag{15.11}$$

Wie zu erkennen ist, machen wir bereits Gebrauch von den zuvor bestimmten Unbekannten. Zu guter Letzt können mit Gleichung (15.8) die zwei verbleibenden Unbekannten bestimmt werden:

$$\begin{aligned} r_{x1} &= \frac{r_x(t + \Delta t) - r_{x0} - \Delta t \cdot (v_{x0} + v_{x1} \cdot \Delta t)}{\Delta t} \\ r_{y1} &= \frac{r_y(t + \Delta t) - r_{y0} - \Delta t \cdot (v_{y0} + v_{y1} \cdot \Delta t) + \frac{1}{2}gt^2}{\Delta t}. \end{aligned} \tag{15.12}$$

Somit sind nun alle Unbekannten bestimmt und Gleichung (15.8) kann zur Bestimmung des Ortes herangezogen werden. Berechnen wir die Position schrittweise für die Intervalle $\{[0, 1), [1, 2), \dots\}$ ergibt sich die euklidische Fehlerdistanz in Bild 15.6.

Anstelle der Interpolation an den Stützstellen $\Delta t = 0$ und $\Delta t = 1$ liessen sich, wie bereits erwähnt, auch die Stützstellen $\Delta t = 0$ und $\Delta t = 0.5\text{s}$ nutzen. Dies führt zu der Fehlerverteilung gemäss Bild 15.7.

In beiden Grafiken ist schön zu erkennen, wie der Fehler jeweils bei einer Stützstelle auf 0 sinkt, zwischen den Stützstellen aber einen Anstieg verzeichnet. Es ist auch gut zu erkennen, dass die Wahl der Stützstellen jeweils am Anfang und am Ende erfolgen sollte, um einen maximalen Bereich des Intervalls abzudecken.

Vergleichen wir den Fehler aus Grafik 15.6 mit Polynomen nullter Ordnung in Grafik 15.3 erkennen wir erneut eine Genauigkeitssteigerung um Faktor vier. Die Erhöhung der Ordnung um eins hat also den selben Effekt wie die Halbierung der Intervalldauer.

Selbstverständlich könnte der Anwender auch beide Verfahren, also die Erhöhung der Ordnung und die Reduktion der Intervalllänge, kombinieren. Der Fehler davon ist in Grafik 15.8 abgebildet. Erneut ist er viermal kleiner geworden. Die Abweichung beträgt nun kaum noch mehr als 1m .

Lösung des Problems mit Taylor-Reihen

Anstelle von linearer Interpolation kann man die Unbekannten $r_{x0}, r_{x1}, r_{y0}, \dots$ auch mit anderen Verfahren bestimmen. Wir können den Ansatz (15.7) als Taylor-Entwicklungen interpretieren. Betrach-

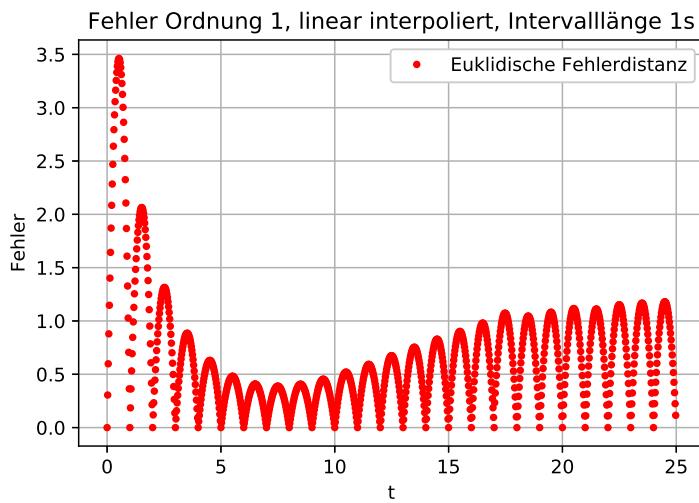


Abbildung 15.6: Fehler bei Polynomen erster Ordnung

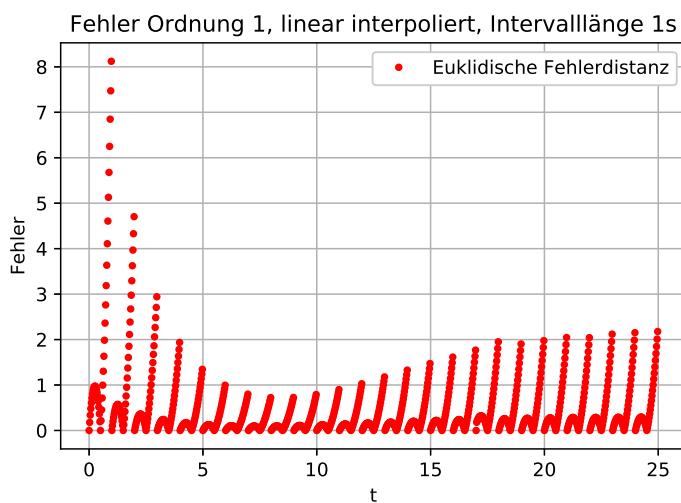


Abbildung 15.7: Fehler bei Polynomen erster Ordnung mit variierten Stützstellen

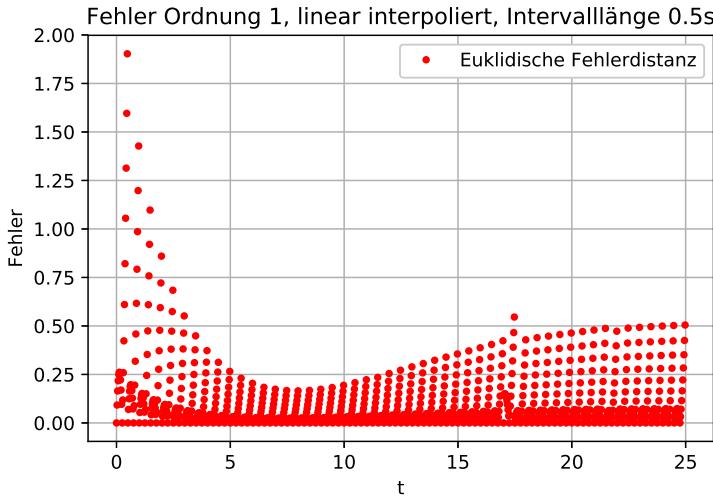


Abbildung 15.8: Fehler bei Polynomen erster Ordnung mit halbierter Intervalldauer

ten wir nur die Ortskoordinate und leiten wir diese nach Δt ab, erhalten wir das folgende System. Um die Notation kurz zu halten, verwenden wir Vektornotation anstelle der Aufteilung nach x - und y -Komponenten:

$$\begin{aligned}\vec{r} &= \vec{r}_0 + \vec{r}_1 \cdot \Delta t \\ \dot{\vec{r}} &= \vec{r}_1\end{aligned}\tag{15.13}$$

Die Unbekannte \vec{r}_0 ist erneut durch setzen von $\Delta t = 0$ sehr einfach zu erlangen, analog zu Abschnitt 15.6.2. \vec{r}_1 kann ebenfalls ohne grossen Aufwand bestimmt werden, indem wir die Ableitung mit Hilfe der Bodenstation wie folgt annähern:

$$\dot{\vec{r}}(t) \approx \frac{\vec{r}(t + \epsilon) - \vec{r}(t)}{\epsilon}.$$

Damit ist bereits eine Annäherung des Ortes möglich. Analog könnte man auch die Geschwindigkeit betrachten.

Beide Varianten, Interpolation und Taylorentwicklung, sind legitim. Die Interpolation ist am Anfang und Ende des Intervalls exakt, hat dafür aber in der Mitte einen relativ hohen Fehler. Die Taylorentwicklung ist vor allem zu Beginn des Intervalls exakt. Dies, da insbesondere bei höheren Ordnungen auch die Krümmung des Kurvenverlaufs berücksichtigt wird. Da sich allerdings alle Stützstellen am Anfang des Intervalls befinden, wird der Kurvenverlauf zum Ende des Intervalls hin immer ungenauer.

15.7 Fazit

Durch das Studium der Störungstheorie haben wir gesehen, dass auch komplexe Bahnverläufe mit begrenzter Rechenleistung sehr gut approximiert werden können. Man benötigt allerdings eine Bodenstation, die fähig ist, exakte Werte an vorgegebenen Stützstellen zu berechnen. Dies kann mit

Hilfe numerischer Verfahren zur Lösung komplexer Differentialgleichungen erreicht werden, wie beispielsweise dem Runge-Kutta-Verfahren. Es lassen sich somit Methoden kreieren, sodass auch ein Satellit oder ein anderes Objekt mit begrenzter Rechenleistung die Bahn sehr genau approximieren kann. Dies, obwohl die Bahn möglicherweise von hunderten planetarischen Objekten gestört wird. Es ist dabei möglich, die Genauigkeit nahezu beliebig zu erhöhen, indem man entweder die Bodenstation in kürzeren Intervallen um neue Werte bittet, oder Polynome höherer Ordnung anstelle der Anfangswerte einsetzt.

16

Störungstheorie für das Eigenwertproblem

Nicolas Tobler

16.1 Einleitung

Das Lösen des Eigenwertproblems ist weit verbreitet mit vielen Anwendungsgebieten. Umso wichtiger ist es, Eigenwerte und Eigenvektoren auch von grösseren Matrizen in kurzer Zeit zu berechnen. Dazu gibt es diverse numerische Verfahren, die alle ihre Vor- und Nachteile haben.

Das klassische Verfahren für kleine Matrizen ist das Finden der Nullstellen des charakteristischen Polynoms. Durch die Nullstellen können die Eigenwerte berechnet werden und anschliessend durch Lösen eines Gleichungssystems auch die dazugehörigen Eigenvektoren. Diese Methode kann jedoch nur bis und mit Ordnung vier explizit gelöst werden, da alle Nullstellen nur bis zu dieser Ordnung explizit berechnet werden können. Des Weiteren stellt sich heraus, dass der Weg über die Eigenwerte numerisch impraktikabel für Floatingpointberechnungen ist.

Viel attraktiver ist das Anwenden von iterativen Methoden, welche auch für grössere Matrizen geeignet sind. Einer der einfachsten Algorithmen ist die Potenzmethode. Dabei wird ein zufällig gewählter Vektor mehrmals mit der Matrix multipliziert und normalisiert, bis er zu einem Eigenvektor konvergiert. Diese Methode funktioniert generell für eine beliebige Matrix, doch sie liefert nur einen Eigenvektor auf einmal.

Falls man das Problem durch bestimmte Eigenschaften der Matrizen einschränken kann, können zum Teil schnellere Algorithmen angewendet werden. Für reelle und symmetrische Matrizen kann zum Beispiel der Jacobi-Algorithmus angewendet werden.

Trotz vielen Methoden bleibt das Berechnen der Eigenwerte und Eigenvektoren sehr teuer. Für manche Anwendungen genügen jedoch auch Approximationen, wenn dadurch massenhaft Rechenleistung und somit auch Zeit gespart werden kann. Die Störungstheorie, english *perturbation*, befasst sich mit der Approximation von Lösungen auf wenig abgeänderte, gestörte Probleme. Auch für das Eigenwertproblem kann ein solches Verfahren entwickelt werden, welches angewendet werden kann, wenn Eigenwerte und Eigenvektoren von einer ähnlichen Matrix schon bekannt sind.

16.2 Problemstellung

Das Eigenwertproblem sucht diejenigen Vektoren $v_i \in \mathbb{R}^n$, die angewendet mit einer Matrix \mathbf{H} nicht die Richtung ändern und dabei nur mit $\lambda_i \in \mathbb{R}$ skaliert werden:

$$\mathbf{H}v_i = \lambda_i v_i \quad (16.1)$$

Manche Applikation benötigen nur Eigenwerte und -Vektoren einer Matrix $\mathbf{H}(\varepsilon)$, die nur wenig von einer Matrix \mathbf{H}_0 mit bekannten Eigenwerten λ_i und Eigenvektoren v_{0i} abweicht. Dies lässt sich mit der Summe

$$\mathbf{H}(\varepsilon) = \mathbf{H}_0 + \varepsilon \mathbf{H}_1 \quad (16.2)$$

schreiben, wobei $\varepsilon \ll 1$ ausdrücken soll, dass der zweite Term der Summe viel kleiner ist als \mathbf{H}_0 . Das Eigenwertproblem für H_0 wird also durch eine im Betrag kleine Matrix $\varepsilon \mathbf{H}_1$ gestört.

Die Störungstheorie erlaubt, die Eigenwerte $\lambda_i(\varepsilon)$ und Eigenvektoren $v_i(\varepsilon)$ von $\mathbf{H}(\varepsilon)$ zu approximieren. Das Verfahren hat jedoch die Limitierung, dass die Eigenvektoren von \mathbf{H}_0 zueinander orthogonal sein müssen, was für alle symmetrischen \mathbf{H}_0 zutrifft. Somit fassen wir zusammen:

Problem. Finde für eine symmetrische, selbstadjungierte Matrix \mathbf{H}_0 mit bekannten Eigenwerten λ_0 und bekannten orthonormierten Eigenvektoren v_{0i} eine Approximation erster Ordnung für die Eigenwerte und Eigenvektoren für

$$\mathbf{H}(\varepsilon) = \mathbf{H}_0 + \varepsilon \mathbf{H}_1,$$

wobei $0 < \varepsilon \ll 1$.

16.3 Anwendungen

Die Störungstheorie im Allgemeinen kann überall angewendet werden, wo ein lösbares Problem mit einer kleinen Wechselwirkung gestört wird. Wie zum Beispiel das Berechnen einer Umlaufbahn eines Planeten unter Einfluss der Anziehungskraft anderer Himmelskörper. Das ungestörte Zweikörpermodell kann mit der Kepler-Gleichung gut gelöst werden, aber für viele Körper gibt es keine praktikable Formel. Ein weiteres Beispiel ist das Berechnen einer Trajektorie unter Berücksichtigung des Luftwiderstands, wie es in Kapitel 15 behandelt wird.

Eines der grössten Anwendungsgebiet der Störungstheorie von Eigenwerten ist das Lösen der Schrödinger-Gleichung in der Quantenmechanik. Dabei werden die Eigenfunktionen des Hamilton-Operators gesucht, um zum Beispiel Spektrallinien von Atomen zu berechnen. Aus diesem Anwendungsgebiet hat die Theorie auch ihren Ursprung.

16.4 Idee

Die Schreibweise als Funktion von ε deutet auf eine Potenzreihe hin. $\mathbf{H}(\varepsilon)$ könnte also als Teil einer Potenzreihe interpretiert werden:

$$\mathbf{H}(\varepsilon) = \mathbf{H}_0 + \varepsilon \mathbf{H}_1 + \varepsilon^2 \mathbf{H}_2 + \varepsilon^3 \mathbf{H}_3 + \dots$$

Ebenso können die Eigenwerte und Eigenvektoren von $\mathbf{H}(\varepsilon)$ als Potenzreihen geschrieben werden:

$$v_i(\varepsilon) = v_{0i} + \varepsilon v_{1i} + \varepsilon^2 v_{2i} + \varepsilon^3 v_{3i} + \dots \quad (16.3)$$

$$\lambda_i(\varepsilon) = \lambda_{0i} + \varepsilon\lambda_{1i} + \varepsilon^2\lambda_{2i} + \varepsilon^3\lambda_{3i} + \dots \quad (16.4)$$

Bei einer Störungsrechnung mit kleinem ε soll eine Approximation erster Ordnung genügen. Somit können alle Terme mit ε^2 und höher weggelassen werden. Falls eine höhere Genauigkeit gewünscht wird, kann auch eine Methode mit einer Approximation zweiter Ordnung gewählt werden. Diese ist jedoch aufwendiger herzuleiten und wird in diesem Kapitel nicht angeschaut.

Nach Einsetzen der Potenzreihen in das Eigenwertproblem (16.1) erhalten wir den Ansatz

$$\mathbf{H}(\varepsilon)\mathbf{v}_i(\varepsilon) = \lambda_i(\varepsilon)\mathbf{v}_i(\varepsilon) \quad (16.5)$$

$$(\mathbf{H}_0 + \varepsilon\mathbf{H}_1 + \dots)(\mathbf{v}_{0i} + \varepsilon\mathbf{v}_{1i} + \dots) = (\lambda_{0i} + \varepsilon\lambda_{1i} + \dots)(\mathbf{v}_{0i} + \varepsilon\mathbf{v}_{1i} + \dots), \quad (16.6)$$

wobei alle Variablen bekannt sind, ausser λ_{1i} und \mathbf{v}_{1i} . Dieser Ansatz soll nun umgeformt werden, um diese Variablen zu bestimmen.

16.5 Herleitung

Die hier verwendete Herleitung basiert auf [1]. Es wurde jedoch auf die in der Quantenmechanik übliche Bra-Ket-Notation verzichtet.

Als erstes wird die Gleichung ausmultipliziert und alle Terme mit ε höherer Ordnung als zwei (rot markiert) werden weggelassen:

$$\mathbf{H}_0\mathbf{v}_{0i} + \varepsilon\mathbf{H}_0\mathbf{v}_{1i} + \varepsilon\mathbf{H}_1\mathbf{v}_{0i} + \varepsilon^2\mathbf{H}_1\mathbf{v}_{1i} + \dots = \lambda_{0i}\mathbf{v}_{0i} + \varepsilon\lambda_{0i}\mathbf{v}_{1i} + \varepsilon\lambda_{1i}\mathbf{v}_{0i} + \varepsilon^2\lambda_{1i}\mathbf{v}_{1i} + \dots \quad (16.7)$$

Die grün markierten Terme entsprechen genau dem Eigenwertproblem (16.1) und können daher subtrahiert werden. Daraus erhalten wir

$$\varepsilon\mathbf{H}_0\mathbf{v}_{1i} + \varepsilon\mathbf{H}_1\mathbf{v}_{0i} = \varepsilon\lambda_{0i}\mathbf{v}_{1i} + \varepsilon\lambda_{1i}\mathbf{v}_{0i} \quad (16.8)$$

$$\mathbf{H}_0\mathbf{v}_{1i} + \mathbf{H}_1\mathbf{v}_{0i} = \lambda_{0i}\mathbf{v}_{1i} + \lambda_{1i}\mathbf{v}_{0i} \quad (16.9)$$

Durch das Linksmultiplizieren mit \mathbf{v}_{0j}^T können anschliessend weitere Vereinfachungen gemacht werden:

$$\mathbf{v}_{0j}^T \mathbf{H}_0 \mathbf{v}_{1i} + \mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i} = \lambda_{0i} \mathbf{v}_{0j}^T \mathbf{v}_{1i} + \lambda_{1i} \mathbf{v}_{0j}^T \mathbf{v}_{0i}. \quad (16.10)$$

Da in den Voraussetzungen orthogonale Eigenvektoren \mathbf{v}_{0j} gefordert sind, sind auf der rechten Seite im zweiten Term alle inneren Produkte unterschiedlicher Vektoren 0:

$$\mathbf{v}_{0j}^T \mathbf{v}_{0i} = \delta_{ij} = \begin{cases} 0 & i \neq j, \\ 1 & i = j. \end{cases} \quad (16.11)$$

Zusätzlich ist vorausgesetzt, dass \mathbf{H}_0 symmetrisch ist. Dies macht die Matrix auch selbstadjungiert, also $\mathbf{H}_0 = \mathbf{H}_0^T$. Dementsprechend ist der erste Term auf der linken Seite

$$\mathbf{v}_{0j}^T \mathbf{H}_0 \mathbf{v}_{1i} = \mathbf{v}_{0j}^T \mathbf{H}_0^T \mathbf{v}_{1i} = (\mathbf{H}_0 \mathbf{v}_{0j})^T \mathbf{v}_{1i}. \quad (16.12)$$

Da \mathbf{H}_0 jetzt mit einem Eigenvektor multipliziert wird, können wir diesen durch den entsprechenden Eigenwert ersetzen:

$$(\mathbf{H}_0 \mathbf{v}_{0j})^T \mathbf{v}_{1i} = (\lambda_{0j} \mathbf{v}_{0j})^T \mathbf{v}_{1i} = \lambda_{0j} \mathbf{v}_{0j}^T \mathbf{v}_{1i}. \quad (16.13)$$

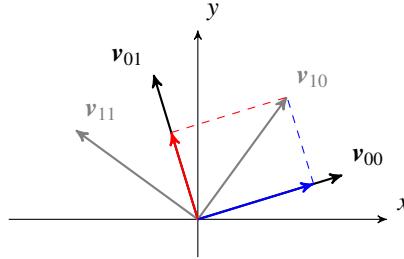


Abbildung 16.1: Beispiel einer Darstellung eines Vektors mit Abbildungen auf anderer Basis. v_{10} ist gleich der Summe der Abbildungen auf die orthonormierten Vektoren v_{01} und v_{00} , welche rot und blau markiert sind.

Daher erhalten wir für Gleichung (16.10) in gleicher Reihenfolge

$$\lambda_{0j}v_{0j}^T v_{1i} + v_{0j}^T \mathbf{H}_1 v_{0i} = \lambda_{0i}v_{0j}^T v_{1i} + \lambda_{1i}\delta_{ij}, \quad (16.14)$$

und noch ein wenig umgeformt

$$v_{0j}^T \mathbf{H}_1 v_{0i} = \delta_{ij}\lambda_{1i} + (\lambda_{0i} - \lambda_{0j})v_{0j}^T v_{1i}. \quad (16.15)$$

Aus dieser Gleichung können mittels Koeffizientenvergleich und Fallunterscheidungen von δ_{ij} zwei nützliche Zwischenergebnisse extrahiert werden:

$$i = j \rightarrow \lambda_{1i} = v_{0i}^T \mathbf{H}_1 v_{0i} \quad (16.16)$$

$$i \neq j \rightarrow v_{0j}^T v_{1i} = \frac{v_{0j}^T \mathbf{H}_1 v_{0i}}{\lambda_{0i} - \lambda_{0j}} \quad \forall i \neq j, \lambda_{0i} \neq \lambda_{0j} \quad (16.17)$$

16.5.1 Nicht entarteter Fall

Die Eigenwerte λ_{1i} können nun einfach in (16.4) eingesetzt werden und liefern bereits eine fertige Formel für das gesuchte

$$\begin{aligned} \lambda_i(\varepsilon) &= \lambda_{0i} + \varepsilon\lambda_{1i} \\ &= \lambda_{0i} + \varepsilon v_{0i}^T \mathbf{H}_1 v_{0i}. \end{aligned}$$

Die Berechnung der Eigenvektoren ist etwas aufwendiger. Die Kernidee dabei ist, dass v_{1i} als Summe von Projektionen auf v_{0j} geschrieben werden kann. Die Eigenvektoren v_{1i} werden dabei mithilfe von Skalarprodukten $v_{0j}^T v_{1i}$ in die Eigenbasis v_{0j} konvertiert:

$$v_i(\varepsilon) = v_{0i} + \varepsilon v_{1i} \quad (16.18)$$

$$= v_{0i} + \varepsilon \sum_j (v_{0j}^T v_{1i}) v_{0j}. \quad (16.19)$$

Dies ist nur möglich, da v_{0j} bereits eine orthonormierte Basis bilden. Ein graphisches Beispiel dieser Umwandlung ist in Abbildung 16.1 illustriert.

Leider gilt Formel (16.17) nur für $i \neq j$. Damit sie eingesetzt werden kann, muss der Fall $i = j$ aus der Summe entfernt werden:

$$\mathbf{v}_i(\varepsilon) = \mathbf{v}_{0i} + \varepsilon (\mathbf{v}_{0i}^T \mathbf{v}_{1i}) \mathbf{v}_{0i} + \varepsilon \sum_{j \neq i} (\mathbf{v}_{0j}^T \mathbf{v}_{1i}) \mathbf{v}_{0j} \quad (16.20)$$

$$= \mathbf{v}_{0i} \left(1 + \varepsilon (\mathbf{v}_{0i}^T \mathbf{v}_{1i}) \right) + \varepsilon \sum_{j \neq i} \frac{\mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i}}{\lambda_{0i} - \lambda_{0j}} \mathbf{v}_{0j} \quad (16.21)$$

Von dieser Gleichung sind nun alle Terme bekannt, ausser \mathbf{v}_{1i} . Das Eigenwertproblem ist aber strenggenommen schlecht gestellt. Ein Eigenvektor kann mit einem beliebigen Skalar multipliziert werden und bleibt dabei ein Eigenvektor. Für eine eindeutige Lösung wünschen wir orthonormierte Eigenvektoren. Die Länge ist also auf 1 gesetzt. Nun können die Eigenvektoren aber noch mit einem Wert mit Betrag 1 multipliziert werden und sie erfüllen immernoch das Eigenwertproblem. Der offensichtliche Fall ist das Multiplizieren mit -1 , aber auch jede komplexe Zahl auf dem Einheitskreis $e^{i\gamma}$ ist möglich. Der unbekannte Term \mathbf{v}_{1i} ist derjenige Freiheitsgrad, welcher die Eigenwerte skaliert. Aus diesem Grund können wir definieren:

$$1 + \varepsilon (\mathbf{v}_{0i}^T \mathbf{v}_{1i}) = e^{i\varepsilon\gamma} \quad (16.22)$$

und für kleine ε kann dies noch weiter vereinfacht werden, indem wir die Approximation erster Ordnung

$$e^{i\varepsilon\gamma} \approx 1 + i\varepsilon\gamma \quad (16.23)$$

akzeptieren, auf welcher die ganze Herleitung bereits beruht. Eingesetzt in Gleichung (16.21) erhalten wir das Endresultat

$$\mathbf{v}_i(\varepsilon) = \mathbf{v}_{0i} (1 + i\varepsilon\gamma) + \varepsilon \sum_{j \neq i} \frac{\mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i}}{\lambda_{0i} - \lambda_{0j}} \mathbf{v}_{0j}. \quad \square \quad (16.24)$$

Somit können wir für den nicht entarteten Fall zusammenfassen:

Auflösung 1. Berechne als erstes die Eigenwerte

$$\lambda_i(\varepsilon) = \lambda_{0i} + \varepsilon \mathbf{v}_{0i}^T \mathbf{H}_1 \mathbf{v}_{0i}.$$

Berechne dann mit den erhaltenen Eigenwerten die dazugehörigen Eigenvektoren

$$\mathbf{v}_i(\varepsilon) = \mathbf{v}_{0i} (1 + i\varepsilon\gamma) + \varepsilon \sum_{j \neq i} \frac{\mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i}}{\lambda_{0i} - \lambda_{0j}} \mathbf{v}_{0j}.$$

16.5.2 Entartung

Die hergeleitete Formel (16.24) für die Eigenvektoren ist nicht definiert für den folgenden Spezialfall: wenn zwei Eigenwerte von \mathbf{H}_0 gleich sind, entsteht eine Division durch 0, wie rot hervorgehoben:

$$\mathbf{v}_i(\varepsilon) = \mathbf{v}_{0i} (1 + i\varepsilon\gamma) + \varepsilon \sum_{j \neq i} \frac{\mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i}}{\lambda_{0i} - \lambda_{0j}} \mathbf{v}_{0j}.$$

Die gleichen Eigenwerte haben zur Folge, dass die dazugehörigen Eigenvektoren nicht mehr eindeutig definiert sind. Sie können irgendwo orthogonal auf einer Hyperebene liegen. Es liegt also kein

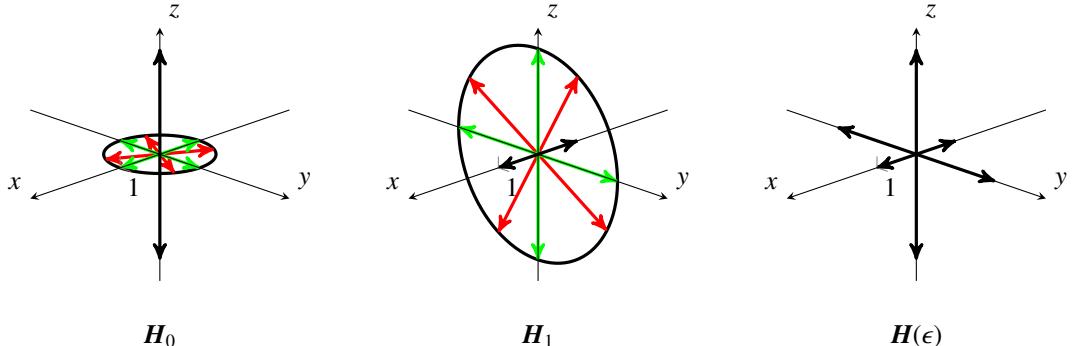


Abbildung 16.2: Eigenräume der Beispielmatrizen von (16.27). Die mehrdimensionalen Eigenräume sind als Kreise dargestellt, wobei alle orthogonale Vektoren auf der Kreisfläche gültige Eigenvektoren sind. Die roten Eigenvektoren ändern schlagartig die Richtung, wobei die grünen richtig gewählt wurden, um die Entartungsrechnung durchzuführen.

einzelner Eigenvektor mehr vor, sondern ein mehrdimensionaler Eigenraum. Mehrfache Eigenwerte werden auch *entartet* genannt.

Durch die Störung einer Matrix \mathbf{H}_0 , also durch Addition einer geeigneten Matrix $\varepsilon \mathbf{H}_1$, können Eigenwerte aufgespalten werden. Als Beispiel, wenn $\varepsilon = 0.01$,

$$\mathbf{H}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad \lambda_0 = \{1, 1, 2\}, \quad \mathbf{v}_0 = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ ? \\ ? \end{pmatrix}, \begin{pmatrix} 0 \\ ? \\ ? \end{pmatrix} \right\} \quad (16.25)$$

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad \lambda_1 = \{1, 1, 2\}, \quad \mathbf{v}_1 = \left\{ \begin{pmatrix} ? \\ ? \\ 0 \end{pmatrix}, \begin{pmatrix} ? \\ ? \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\} \quad (16.26)$$

$$\mathbf{H}(\varepsilon) = \begin{pmatrix} 1.01 & 0 & 0 \\ 0 & 2.01 & 0 \\ 0 & 0 & 2.02 \end{pmatrix}, \quad \lambda(\varepsilon) = \{1.01, 2.01, 2.02\}, \quad \mathbf{v}(\varepsilon) = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}. \quad (16.27)$$

Die Dimensionen der Eigenvektoren, die einen Eigenraum bilden, sind mit Fragezeichen gekennzeichnet. Bei einer numerischen Berechnung der Eigenvektoren von entarteten Eigenwerten, wie es zum Beispiel MATLAB oder NumPy macht, wird ein Eigenvektor in diesem Eigenraum einfach gewählt. Die Eigenräume der Matrizen des Beispiels sind in Abbildung 16.2 illustriert. Die Eigenvektoren ändern sich also schon schlagartig für ganz kleine $\varepsilon = 0.01$. Dies ist sehr unpraktisch, wenn man berücksichtigt, dass die Störung nur einen kleinen Einfluss auf v_{0i} haben soll.

Damit die Störungstheorie auch für entartete Eigenwerte funktioniert, muss das Problem mit der Division durch 0 in Formel (16.17) beseitigt werden. Um dies in den Griff zu kriegen, müssen wir Formel (16.15) für den Fall $i \neq j$ und $\lambda_{0i} = \lambda_{0j}$ anschauen, welcher zu dieser Division durch 0 geführt hatte. Dabei erhalten wir

$$\begin{aligned} \mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i} &= \delta_{ij} \lambda_{1i} + (\lambda_{0i} - \lambda_{0j}) \mathbf{v}_{0j}^T \mathbf{v}_{0i} \\ \mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i} &= 0 + 0 \mathbf{v}_{0j}^T \mathbf{v}_{0i} = 0. \end{aligned} \quad (16.28)$$

Falls der Zähler von (16.17) 0 ist, geht die Gleichung auf. Im Bruch mit Division durch 0 haben wir nun $\frac{0}{0}$.

Wir können also Formel (16.24) auch für entartete Eigenwerte brauchen, falls

$$\mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i} = 0 \quad \forall \quad i, j \quad \text{entartet.} \quad (16.29)$$

Damit dies stimmt, müssen die entarteten Eigenvektoren \mathbf{v}_{0i} so gewählt werden, dass sie in der Eigenbasis von \mathbf{H}_1 stehen. Das hat zur Folge, dass die \mathbf{v}_{0i} die Richtung nicht ändern wenn \mathbf{H}_1 auf sie angewendet wird. Somit sind \mathbf{v}_{0j} und $\mathbf{H}_1 \mathbf{v}_{0i}$ orthogonal und dadurch das Skalarprodukt 0. Dazu bilden wir \mathbf{H}_1 auf die Basis von entarteten, zufällig gewählten Eigenvektoren ab:

$$\mathbf{H}' = \mathbf{v}_{0i}^T \mathbf{H}_1 \mathbf{v}_{0i} \quad \forall \quad i \quad \text{entartet.} \quad (16.30)$$

Dadurch erhalten wir eine kleinere Matrix \mathbf{H}' , welche nur die entarteten Dimensionen enthält. Von dieser müssen wir nun die Eigenvektoren finden, die das Eigenwertproblem

$$\mathbf{H}' \mathbf{v}'_i = \lambda_i \mathbf{v}'_i \quad \forall \quad i \quad \text{entartet} \quad (16.31)$$

bilden. Allerdings ist dieses Problem einiges einfacher als das ursprüngliche Eigenwertproblem, da die Matrix in der Regel viel kleiner ist und die Eigenwerte λ_i schon bekannt sind. Die gefundenen Eigenvektoren müssen nun zurücktransformiert werden und anstelle der ursprünglichen, zufälligen Vektoren verwendet werden.

$$\mathbf{v}_{0i} \leftarrow \mathbf{v}_{0i} \mathbf{v}'_i \quad \forall \quad i \quad \text{entartet.} \quad (16.32)$$

Nun sind auch die entarteten Eigenvektoren so ausgerichtet, dass sie bei einer Störung keine Sprünge machen und für die Berechnung geeignet sind. Zusammengefasst, auch für entartete λ_0 gilt somit:

Auflösung 2. Berechne Eigenwerte

$$\lambda_i(\epsilon) \leftarrow \lambda_{0i} + \epsilon \mathbf{v}_{0i}^T \mathbf{H}_1 \mathbf{v}_{0i}.$$

Falls \mathbf{H}_0 entartet ist, berechne und benutze die neuen, korrigiere Ausrichtung der Eigenvektoren mit

$$\begin{aligned} \mathbf{H}' &= \mathbf{v}_{0i}^T \mathbf{H}_1 \mathbf{v}_{0i} \quad \forall \quad i \quad \text{entartet} \\ \mathbf{v}' &= \text{Eig}(\mathbf{H}') \\ \mathbf{v}_{0i} &\leftarrow \mathbf{v}_{0i} \mathbf{v}'_i \quad \forall \quad i \quad \text{entartet}, \end{aligned}$$

wobei für $\text{Eig}()$ ein beliebiger Algorithmus für das Finden von Eigenvektoren verwendet werden kann. Schlussendlich, berechne die Eigenvektoren

$$\mathbf{v}_i(\epsilon) \leftarrow \mathbf{v}_{0i}(1 + i\epsilon\gamma) + \epsilon \sum_{j \neq i, \text{nicht entartet}} \frac{\mathbf{v}_{0j}^T \mathbf{H}_1 \mathbf{v}_{0i}}{\lambda_{0i} - \lambda_{0j}} \mathbf{v}_{0j}.$$

16.6 Folgerungen

Die Störungstheorie für Eigenwerte ist eine elegante Methode um kleine Änderungen eines Eigenwertproblems zu approximieren. Da die Berechnung im nicht entarteten Fall mit wenigen Matrixmultiplikationen explizit durchgeführt werden kann, ist ein Bruchteil der Rechenleistung notwendig.

Literatur

- [1] Andreas Müller. *Mathematisches Seminar Quantenmechanik*. 2015. Kap. 10.

17

Kettenbrüche

Benjamin Bouhafs-Keller

17.1 Einleitung

Die Entwicklung der Theorie der Kettenbrüche ist durch das Bedürfnis, Brüche oder schwer fassbare Zahlen zu approximieren, motiviert worden. Kettenbrüche fanden Verwendung bei der Annäherung von Verhältnisgrößen in Form von Brüchen, zur Ermittlung von Schaltjahren bei der Kalenderberechnung, zur Annäherung natürlicher Konstanten wie e und π und zum Beweis der Irrationalität bestimmter Zahlen. Mit Hilfe von Kettenbruchsystemen können Approximationsalgorithmen für Funktionen entwickelt werden. Einige der vielseitigen Aspekte der Kettenbrüche werden wir in den folgenden Abschnitten erläutern.

17.1.1 Definition

- Ein *Kettenbruch* ist eine eindeutige Darstellungsform für reelle Zahlen. Ein allgemeiner Kettenbruch ist von der Form:

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{\dots + \cfrac{b_{n-1}}{a_{n-1} + \cfrac{b_n}{a_n}}}}, \quad (17.1)$$

wobei $a_i, b_i \in \mathbb{Z}$.

- Ein *regulärer* oder *einfacher Kettenbruch* ist definiert als ein Bruch der Form:

$$[x_0; x_1, x_2, \dots, x_n] = \frac{a}{b} = x_0 + \cfrac{1}{x_1 + \cfrac{1}{x_2 + \cfrac{1}{x_3 + \cfrac{1}{x_4 + \cfrac{1}{\dots + \cfrac{1}{x_n}}}}}}$$

mit $x_0, x_1, x_2, x_3, \dots \in \mathbb{N}$.

Eine alternative Schreibweise für reguläre Kettenbrüche ist $[x_0; x_1, x_2, x_3, \dots]$. Dies ist eine besonders kompakte Art, einen Kettenbruch zu beschreiben. Bei allgemeinen Kettenbrüchen kann eine Darstellung mit eckigen Klammern nicht möglich sein, da im Zähler keine Einsen mehr stehen.

Für unendliche Folge von x_0, x_1, \dots kann der Wert des Kettenbruchs durch den Grenzwert

$$\lim_{n \rightarrow \infty} [x_0; x_1, \dots, x_n]$$

definiert werden. Kettenbrüche können schnell konvergierende Approximationen liefern.

Damit also das, was ich unter der Bezeichnung Kettenbruch verstehe, besser erläutert wird, biete ich Beispiele für rationalen und irrationalen Zahlen dar.

17.2 Rationale Zahlen

17.2.1 Endliche Kettenbrüche

Die Menge aller rationalen Zahlen wird mit \mathbb{Q} bezeichnet. Ein endlicher Kettenbruch ist ein Bruch der Form

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{\dots}{\dots + \cfrac{b_{n-1}}{a_{n-1} + \cfrac{b_n}{a_n}}}}} \quad (17.2)$$

in welchem a_0, a_1, \dots, a_n und b_1, b_2, \dots, b_n ganze Zahlen darstellen, die mit Ausnahme möglicherweise von a_0 alle positiv sind. Die Kettenbruchentwicklung von $x \in \mathbb{R}$ bricht genau dann nach endlich vielen Schritten ab, wenn x rational ist. So bilden rationale Zahlen endliche Kettenbrüche.

17.2.2 Euklidischer Algorithmus

Die Umwandlung einer rationalen Zahl in einen Kettenbruch erfolgt mit Hilfe des euklidischen Algorithmus. Jede rationale Zahl x lässt sich auf eindeutige Weise in einen endlichen Kettenbruch entwickeln, dessen letzter Teilnenner grösser oder gleich 2 ist.

Beispiel. Als Beispiel bestimmen wir den Kettenbruch von $\frac{17}{10} = [1; 1, 2, 3]$:

$$\frac{17}{10} = 1 + \frac{7}{10} = 1 + \frac{1}{\frac{10}{7}} = 1 + \frac{1}{1 + \frac{3}{7}} = 1 + \frac{1}{1 + \frac{1}{\frac{7}{3}}} = 1 + \frac{1}{1 + \frac{1}{2 + \frac{1}{3}}}. \quad (17.3)$$

Dies entspricht der Durchführung des euklidischen Algorithmus:

$$\begin{aligned} 17 &= 1 \cdot 10 + 7, \\ 10 &= 1 \cdot 7 + 3, \\ 7 &= 2 \cdot 3 + 1, \\ 3 &= 3 \cdot 1 + 0. \end{aligned}$$

Diese Methode kann auch umgekehrt angewendet werden. Rechnen wir den Kettenbruch in einen Bruch zurück:

$$\begin{aligned} [1; 1, 2, 3] &\rightarrow 2 + \frac{1}{3} = \frac{7}{3} \\ &\rightarrow 1 + \frac{3}{7} = \frac{10}{7} \\ &\rightarrow 1 + \frac{7}{10} = \frac{17}{10}. \end{aligned} \quad \circlearrowright$$

Wie man im Beispiel sieht, ist die intuitive Berechnung der Näherungsbrüche eines Kettenbruchs, indem man ihn von unten her auflöst, sehr umständlich. Durch ein rekursives Bildungsgesetz für Zähler und Nenner kann man die Berechnung erheblich vereinfachen. Außerdem kann man mit Hilfe dieser Rekursionsformel die Grenzwerte unendlicher Kettenbrüche untersuchen.

17.2.3 Rekursionsformel

Die Berechnung der Konvergenten (siehe Abschnitt 17.3.3) eines Kettenbruches kann erheblich vereinfacht werden, indem eine Rekursionsformeln für den Zähler und den Nenner eingeführt wird. Anders formuliert können Grenzwerte für unendliche Zahlen oder Funktionen mit Hilfe der Rekursionsformel schneller definiert werden.

Betrachten wir zunächst die folgende Näherungsbrüche:

$$a_0 + \frac{b_1}{a_1}, \quad a_0 + \frac{b_1}{a_1 + \frac{b_2}{a_2}}, \quad \dots$$

Wie in Abschnitt 17.1.1 erwähnt wird hier der Nenner wieder als Kettenbruch dargestellt. Insofern werden Terme der Form

$$a_k + \frac{b_{k+1}}{a_{k+1} + \frac{p}{q}}$$

immer zu

$$\frac{b_{k+1}}{a_{k+1} + \frac{p}{q}} = \frac{b_{k+1} \cdot q}{a_{k+1} \cdot q + p}$$

ausgerechnet. Dies lässt sich auch durch die folgende Matrzenschreibweise ausdrücken:

$$\begin{pmatrix} \text{neuer Zähler} \\ \text{neuer Nenner} \end{pmatrix} = \begin{pmatrix} b_{k+1} \cdot q \\ a_{k+1} \cdot q + p \end{pmatrix} = \begin{pmatrix} 0 & b_{k+1} \\ 1 & a_{k+1} \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}. \quad (17.4)$$

Der “unterste” Bruch ist $\frac{b_n}{a_n}$, also in Vektorschreibweise

$$a_0 + \begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} \cdots \begin{pmatrix} 0 & b_{n-1} \\ 1 & a_{n-1} \end{pmatrix} \begin{pmatrix} b_n \\ a_n \end{pmatrix} \quad \text{Aufbau “von rechts nach links”}.$$

Aber das Matrizenprodukt kann man auch von links beginnend ausmultiplizieren als

$$\begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} \cdots \begin{pmatrix} 0 & b_{n-1} \\ 1 & a_{n-1} \end{pmatrix} \begin{pmatrix} b_n \\ a_n \end{pmatrix},$$

d.h. man kann bei b_1, a_1 beginnen, dies wird weiter unten im Detail ausgeführt.

Man kann die beiden Schritte

$$\frac{p}{q} \rightarrow a_k + \frac{p}{q} \rightarrow \frac{b_k}{a_k + \frac{p}{q}}$$

auch separat als Matrizen schreiben:

$$a_k + \frac{p}{q} = \frac{a_k \cdot q + p}{q} \quad \text{d. h.} \quad \begin{pmatrix} 1 & a_k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}$$

$$\frac{b_k}{\frac{p}{q}} = \frac{b_k \cdot q}{p} \quad \text{d. h.} \quad \begin{pmatrix} 1 & b_k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}.$$

Tatsächlich ist die Kombination der beiden Schritte

$$\begin{pmatrix} 0 & b_n \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & a_n \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & b_n \\ 1 & a_n \end{pmatrix}$$

wie in (17.4) gefunden.

Aufbau des Produktes von links nach rechts

Das Produkt von Matrizen kann man auch von links her ausmultiplizieren.

$$\begin{pmatrix} 1 & a_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} \begin{pmatrix} 0 & b_2 \\ 1 & a_2 \end{pmatrix} \cdots \begin{pmatrix} 0 & b_{n-1} \\ 1 & a_{n-1} \end{pmatrix} \begin{pmatrix} b_n \\ a_n \end{pmatrix}$$

- Start:

$$\begin{pmatrix} 1 & a_0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A_{-1} & A_0 \\ B_{-1} & B_0 \end{pmatrix} = P_0$$

- Schritt 1: von rechts mit nächsten Matrixfaktor multiplizieren

$$\begin{pmatrix} 1 & a_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & b_1 \\ 0 & a_1 \end{pmatrix} = \begin{pmatrix} a_0 & b_1 + a_0 \cdot a_1 \\ 1 & a_1 \end{pmatrix} = \begin{pmatrix} A_{-1} & A_0 \\ B_{-1} & B_0 \end{pmatrix} \begin{pmatrix} 1 & b_1 \\ 0 & a_1 \end{pmatrix} = \begin{pmatrix} A_0 & A_1 \\ B_0 & B_1 \end{pmatrix} = P_1$$

- Schritt 2:

$$P_1 \cdot \begin{pmatrix} 0 & b_2 \\ 1 & a_2 \end{pmatrix} = \begin{pmatrix} A_0 & A_1 \\ B_0 & B_1 \end{pmatrix} \begin{pmatrix} 0 & b_2 \\ 1 & a_2 \end{pmatrix} = \begin{pmatrix} A_1 & b_2 \cdot A_0 + a_2 \cdot A_1 \\ B_1 & b_2 \cdot B_0 + a_2 \cdot B_1 \end{pmatrix} = P_2$$

- Schritt 3:

$$P_2 \cdot \begin{pmatrix} 0 & b_3 \\ 1 & a_3 \end{pmatrix} = \begin{pmatrix} A_1 & A_2 \\ B_1 & B_2 \end{pmatrix} \begin{pmatrix} 0 & b_3 \\ 1 & a_3 \end{pmatrix} = \begin{pmatrix} A_2 & b_3 \cdot A_1 + a_3 \cdot A_2 \\ B_2 & b_3 \cdot B_1 + a_3 \cdot B_2 \end{pmatrix} = P_2 \quad \text{liefert.}$$

- Schritt $n - 1$: So macht man weiter bis zum Schritt $n - 1$, der die Matrix

$$P_{n-1} = \begin{pmatrix} A_{n-2} & A_{n-1} \\ B_{n-2} & B_{n-1} \end{pmatrix}$$

liefert.

- Schritt n : Zähler und Nenner des Näherungsbruchs bekommt man jetzt durch Multiplikation mit

$$\begin{pmatrix} b_n \\ a_n \end{pmatrix}.$$

Dies ist aber die zweite Spalte der Matrix

$$\begin{pmatrix} 0 & b_n \\ 1 & a_n \end{pmatrix},$$

d. h. Zähler und Nenner stehen in der zweiten Spalte von P_n .

Rekursionsformeln für A_n und B_n

Wir möchten jetzt Formeln für die Berechnung von Zähler A_n und Nenner B_n des Wertes

$$\frac{A_n}{B_n} = a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{\dots + \cfrac{a_2 + \cfrac{b_{n-1}}{\dots + \cfrac{b_n}{a_{n-1} + \cfrac{b_n}{a_n}}}}}}$$

des n -ten Näherungsbruches bekommen. Die Berechnung von A_n, B_n kann man auch ohne die Matrizenbeschreibung aufschreiben:

- Start:

$$A_{-1} = 0$$

$$A_0 = a_0$$

$$B_{-1} = 1$$

$$B_0 = 1$$

$$\rightarrow 0\text{-te Näherung: } \frac{A_0}{B_0} = a_0$$

- Schritt $k \rightarrow k + 1$:

$$\begin{aligned} k \rightarrow k + 1 : \quad A_{k+1} &= A_{k-1} \cdot b_k + A_k \cdot a_k \\ B_{k+1} &= B_{k-1} \cdot b_k + B_k \cdot a_k \end{aligned}$$

- Näherungsbruch n :

$$\frac{A_n}{B_n}$$

Einfache Kettenbrüche

Schliesslich sind für reguläre oder einfache Kettenbrüche die Formeln einfacher, weil $b_k = 1$. Es gilt

$$\begin{aligned} A_{k+1} &= A_{k-1} + A_k \cdot a_k \\ B_{k+1} &= B_{k-1} + B_k \cdot a_k. \end{aligned}$$

Hiermit haben wir die wichtigsten Zusammenhänge als Vorbereitung für die Näherungszahlen herausgearbeitet.

17.3 Irrationale Zahlen

17.3.1 Definition

Irrationale Zahlen bilden unendliche Kettenbrüche, d. h. sind durch eine periodische oder nicht periodische Kettenbruchentwicklung ausgezeichnet. Ein unendlicher regelmässiger Kettenbruch wird in folgender Form dargestellt

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \dots}}}, \quad (17.5)$$

wobei a_0, a_1, a_2, \dots eine unendliche Folge von ganzen Zahlen bilden. Sie sind auch wie beim endlichen Kettenbruch alle bis auf möglicherweise a_0 positiv.

Zunächst sollen einige Beispiele für die Kettenbruchentwicklung irrationaler Zahlen betrachtet werden.

17.3.2 Periodische Kettenbrüche

In diesem Abschnitt wollen wir nun auf eine spezielle Form eingehen und zwar auf unendliche regelmässige Kettenbrüche, die ein bemerkenswertes Bildungsgesetz befolgen. Das besondere an diesen Kettenbrüchen ist, dass gleiche Teillnenner wiederholt auftreten. Für den Kettenbruch

$$[3; 1, 2, 1, 6, 1, 2, 1, 6, \dots]$$

heisst das: der Kettenbruch mit der Periode 1,2,1,6 hat die Periodenlänge $n = 4$, er wird in der Form $[3; \overline{1, 2, 1, 6}]$ geschrieben.

Beispiel. Betrachten wir den periodischen einfachen Kettenbruch $[3; \bar{6}] = [3; 6, 6, 6, \dots]$.

$$[3; \bar{6}] = 3 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{\dots}}}}} = x. \quad (17.6)$$

Die Euklidische Methode mit der rekursiven Bildungsgesetz für Zähler und Nenner würde hier unendlich sein und deshalb schwierig zu berechnen. Um diesen Kettenbruch vollständig darzustellen müssen wir ein Muster erzeugen. Daher werden nur die ersten Brüche (Zahlen) betrachtet.

$$\begin{aligned} [3; 6, 6, 6] &= 3 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6}}}}} = 3 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{37}{6}}}} = 3 + \cfrac{1}{6 + \cfrac{37}{37}} = 3 + \cfrac{37}{228} = 3 + \cfrac{684 + 37}{228} = \cfrac{721}{228} \\ &\approx 3.\underline{162280702}. \end{aligned}$$

Wenn wir den periodischen Kettenbruch so verändern das die Kettenbruchentwicklung immer mit 6 fortläuft, dann können wir unser Kettenbruch als x bezeichnen und darauf 3 addieren. Anders formuliert ist wie in einem periodischen Kettenbruch ein **Teil oder inneren Kettenbruch**, der ähnlich ist wie der ganze Kettenbruch. Dies ergibt folgendes Muster, das wir mit Hilfe einer quadratischen Gleichung lösen können:

$$\begin{aligned} y &= x + 3 = 6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{\dots}}} = [6; \bar{6}] \\ &\Rightarrow y = 6 + \cfrac{1}{y} \quad | \cdot y \\ &\Rightarrow y^2 = 6y + 1 \\ &\Rightarrow y = 3 \pm \sqrt{9 + 1} = 3 \pm \sqrt{10} \quad \Rightarrow \quad x = \sqrt{10} \approx 3.16227766. \end{aligned}$$

Der oben gefundene Näherungsbruch $721/228 \approx 3.\underline{16228}$ stimmt auf fünf Stellen nach dem Komma mit dem exakten Wert überein. \circlearrowright

Das Beispiel zeigt, wie man den Wert eines periodischen Kettenbruchs bestimmt. Man kann die Idee auch für längere Periode verwenden.

Beispiel. Betrachten wir einen neuen Kettenbruch, $[\overline{2, 3}] = [2; 3, 2, 3, \dots]$. Sein Wert ist als unend-

licher Kettenbruch irrational und lässt sich wie folgt berechnen. Setzen wir $x := (\overline{2, 3})$, dann gilt

$$x = 2 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{3 + \dots}}} = 2 + \cfrac{1}{3 + \cfrac{1}{x}} \quad (17.7)$$

Dies führt auf die quadratische Gleichung

$$\begin{aligned} x &= 2 + \cfrac{1}{3 + \cfrac{1}{x}} \\ x - 2 &= \cfrac{1}{3 + \cfrac{1}{x}} \\ (x - 2)\left(3 + \cfrac{1}{x}\right) &= 1 \\ 3x - 6 - \cfrac{2}{x} &= 0 \quad | \quad \times \frac{x}{3} \\ x^2 - 2x - \cfrac{2}{3} &= 0 \end{aligned}$$

was die positive Lösung $x = \frac{3 + \sqrt{15}}{3}$ liefert. \circ

Der oben aufgeführte Kettenbruch x ist ein Beispiel für periodische einfache Kettenbrüche, der Nullstelle eines quadratischen Polynoms mit rationalen Koeffizienten ist. Anders gesagt ist die reelle Irrationalzahl x Wurzel einer quadratischen Gleichung

$$ax^2 + bx + c = 0, \quad (17.8)$$

dann ist die Kettenbruchentwicklung von x periodisch, das bedeutet die Existenz einer Schranke n_0 und einer Periode $k \in \mathbb{N}$ mit $x_n + k = x_n$ für alle $n \geq n_0$.

Satz 17.1 (Euler-Lagrange). *Jeder periodische regulärer Kettenbruch ist eine quadratische Irrationalzahl und umgekehrt. $x \in \mathbb{R}$ hat genau dann eine Darstellung als unendlicher, periodischer Kettenbruch, wenn x eine reell-quadratische Irrationalzahl ist (d.h. $x \notin \mathbb{Q}$ ist Lösung einer quadratischen Gleichung $ax^2 + bx + c = 0, a \neq 0$ mit rationalen Koeffizienten a, b, c) [3].*

17.3.3 Nicht periodische Kettenbrüche

Es stellen sich dieselben Fragen wie im vorangegangenen Abschnitt. Neu hinzu kommt das Problem, ob bzw. wann die Kettenbruchentwicklung überhaupt konvergiert. Für eine unendliche Folge x_0, x_1, \dots ist der Kettenbruch $[x_0; x_1, \dots]$ nur dann definiert, wenn die Folge der Näherungsbrüche p_n/q_n konvergiert. In diesem Fall hat der unendliche Kettenbruch $[x_0; x_1, \dots]$ den Wert

$$\lim_{n \rightarrow \infty} [x_0; x_1, \dots, x_n] \quad (17.9)$$

oder anders dargestellt

$$\omega = x_0 + \cfrac{1}{x_1 + \cfrac{1}{x_2 + \cfrac{1}{\dots + x_n}}}. \quad (17.10)$$

Man sagt, $\omega > 0$ sei durch einen unendlichen Kettenbruch darstellbar, wenn die endlichen Kettenbrüche n -ter Ordnung $[x_0; x_1, x_2, \dots, x_n]$ gegen ω konvergieren.

Beispiel. Betrachten wir folgenden Kettenbruch

$$\frac{1351}{3625} = [0; 2, 1, 2, 6, 2, 1, 1, 2, 1, 3] = 0.372689655172$$

und berechnen wir die Näherungsbrüche:

$$K_0 = [0] = 0$$

$$K_1 = [0; 2] = 0 + \frac{1}{2} = \frac{1}{2} = 0.5$$

$$K_2 = [0; 2, 1] = 0 + \cfrac{1}{2 + \frac{1}{1}} = \frac{1}{3} = 0.\bar{3}$$

$$K_3 = [0; 2, 1, 2] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{2}}} = \frac{3}{8} = 0.375$$

$$K_4 = [0; 2, 1, 2, 6] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \frac{1}{6}}}} = \frac{19}{51} = 0.37254$$

$$K_5 = [0; 2, 1, 2, 6, 2] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{6 + \frac{1}{2}}}}} = \frac{41}{110} = 0.372\bar{7}$$

$$K_6 = [0; 2, 1, 2, 6, 2, 1] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{2 + \frac{1}{1}}}}}} = \frac{60}{161} = 0.37267$$

$$K_7 = [0; 2, 1, 2, 6, 2, 1, 1] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1}}}}}}} = \frac{101}{271} = 0.37269$$

$$K_8 = [0; 2, 1, 2, 6, 2, 1, 1, 2] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2}}}}}}} = \frac{262}{703} = 0.372688$$

$$K_9 = [0; 2, 1, 2, 6, 2, 1, 1, 2, 1] = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1}}}}}}}} = \frac{363}{974} = 0.3726899$$

$$\begin{aligned}
 K_{10} = [0; 2, 1, 2, 6, 2, 1, 1, 2, 1, 3] &= 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{3}}}}}}}}} \\
 &= 0.372689655172. \quad \circ
 \end{aligned}$$

Mit diesem Beispiel werden die Teilkettenbrüche und Näherung zum Endresultat ersichtlich. Die Näherungsbrüche mit geradem (bzw. ungeradem) Index ist streng monoton steigend (bzw. fallend) und jeder Näherungsbruch mit geradem Index ist kleiner als jeder Näherungsbruch mit ungeradem Index [1]:

1. $K_0 < K_2 < K_4 < \dots$
2. $K_1 > K_3 > K_5 > \dots$
3. $K_{2s} < K_{2r+1}, r, s \in \mathbb{N}$

Es gilt offensichtlich $K_0 < K_2 < K_4 < \dots < K_{2n} < \dots < K_{2n+1} < \dots < K_5 < K_3 < K_1$ und $\frac{1351}{3625}$ wird von jeweils zwei aufeinanderfolgenden Näherungsbrüchen eingeschlossen.

Die Folge der Näherungsbrüche des unendlichen Kettenbruchs $[a_0; a_1, a_2, \dots]$ mit $a_0 \in \mathbb{Z}$ konvergiert von unten gegen einen Grenzwert, den wir als α bezeichnen. Anderseits sind die Näherungsbrüche mit ungeradem Index nach unten begrenzt. Somit sind beide Folgen monoton und beschränkt und konvergieren in α . Der erwähnte Ansatz gilt nur für reguläre Kettenbrüche, es wäre zum Beispiel nicht der Fall, wenn als Teilnenner beliebige positive reelle Zahlen zugelassen wären. Hiermit haben wir beobachtet wie die K_i den Grenzwert von oben und unten her approximieren.

Es gibt auch Zahlen, deren Kettenbruchdarstellung gewisse Regelmässigkeiten aufweisen, ohne periodisch zu sein. Zum Beispiel die Identität $e = [2; 1, 2, 1, 1, 4, 1, 1, 6, 1, \dots]$. Dieser Kettenbruch ist nicht periodisch, die Teilnenner können aber durch eine rekursive Folge bestimmt werden.

Zusammenfassung

- Jede positive rationale Zahl lässt sich durch einen endlichen Kettenbruch darstellen und jeder endliche Kettenbruch stellt eine positive rationale Zahl dar.
- Jeder unendliche Kettenbruch stellt eine positive irrationale Zahl dar und jede irrationale Zahl lässt sich durch einen unendlichen Kettenbruch darstellen.
- Jeder periodische Kettenbruch stellt eine quadratische Irrationalität dar und jede quadratische Irrationalität lässt sich durch einen periodischen Kettenbruch darstellen.

17.4 Approximation

In der Einleitung wurde erwähnt, dass die Bestimmung von guten Näherungsbrüchen eine wichtige Anwendung von Kettenbrüchen ist. Es gilt nämlich, dass jeder Näherungsbruch der Kettenbruchentwicklung einer reellen Zahl eine besonders gute rationale Näherung dieser Zahl ist.

17.4.1 Definition

Eine rationale Zahl $\frac{a}{b}$ mit $b > 0$ heisst beste Näherung erster Art an eine reelle Zahl x , wenn es keine von $\frac{a}{b}$ verschiedene rationale Zahl mit gleichem oder kleinerem Nenner gibt, die bezüglich des Absolutbetrages näher bei x liegt. Das heisst, dann gilt für alle rationalen Zahlen $\frac{c}{d} \neq \frac{a}{b}$ mit $0 < d \leq b$:

$$\left| x - \frac{a}{b} \right| < \left| x - \frac{c}{d} \right|. \quad (17.11)$$

17.4.2 Näherungsgesetz

Ziel dieses Abschnittes ist, eine genügend gute Approximation der Näherungsbrüche zu zeigen. Wir werden hier Resultate zusammentragen und diese dokumentieren. Gibt man sich eine beliebige Zahl x vor, so kann man sich die Frage stellen, welche “unkürzbaren” Brüche $\frac{p}{q}$ mit vorgegebenem Höchstnenner sich gut approximieren lässt.

Beispiel. Näherung von π mit dem unendlichen Kettenbruch

$$\pi = [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, \dots].$$

Die Näherung $3.14 = \frac{314}{100}$ ist eine Näherung. Aber $\frac{22}{7} = 3.14285714\dots$ hat einen viel kleineren Nenner und ist eine deutlich bessere Näherung von π . Eine noch bessere Näherung ist der Näherungsbruch

$$\frac{355}{113} = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1}}} = 3.14159\overline{2}. \quad (17.12)$$

Folgende Näherungswerte von π können schnell und einfach berechnet werden:

$$3, \quad \frac{22}{7} \approx 3.143, \quad \frac{333}{106} \approx 3.14151, \quad \frac{355}{113} \approx 3.1415929, \quad \frac{103993}{33102} \approx 3.1415926530, \quad \dots. \quad \circlearrowright$$

Die Bestapproximation ist einfach formuliert durch die Bestimmung derjenigen rationalen Brüchen, welche von einer gegebenen rationalen oder irrationalen Zahl einen festgelegten minimalen Abstand haben und dabei einen möglichst kleinen positiven Nenner besitzen. Diesbezüglich liefern Kettenbrüche bestmögliche Approximationen durch rationale Zahlen [2].

n	$f_n(1)$
2	0.750000
3	0.791667
4	0.784314
5	0.785586
6	0.785366
7	0.785404
8	0.785397
9	0.785398

Tabelle 17.1: Näherungsstufen mit Kettenbruchentwicklung von der Funktion $\tan^{-1}(1)$

17.4.3 Approximation einer Funktion

Die Funktion $\tan^{-1}(x)$ spielt bei der Berechnung von π an vielen Stellen eine Rolle. Der Kettenbruch von $\tan^{-1}(x)$ kann folgendermassen dargestellt werden.

$$\tan^{-1}(x) = \frac{x}{1 + \frac{4x^2}{3 + \frac{9x^2}{5 + \frac{16x^2}{7 + \frac{\dots}{9 + \dots}}}}} \quad (|x| < 1). \quad (17.13)$$

Diese spezielle Darstellungsform der Arkustangensfunktion wird in Kapitel 19 begründet. Somit kann der Kettenbruch auch umgeschrieben werden als Funktion f_n

$$f_n(x) = \frac{x}{1 + \frac{4x^2}{3 + \frac{\dots}{5 + \frac{(n-1)^2 x^2}{\dots + \frac{2n-1}{}}}}} \quad (|x| < 1). \quad (17.14)$$

Hiermit kann nach der n -ten Bildung der Kettenbruchrekursion einen Grenzwert erreichen:

$$\tan^{-1}(x) = \lim_{n \rightarrow \infty} f_n(x), \quad (|x| < 1). \quad (17.15)$$

Die Konvergenz der Funktion kann an einem Beispiel beurteilt werden. In Tabelle 17.1 sind die Näherungsbrüche von

$$\tan^{-1}(1) = \pi/4 \approx 0.785398 \quad (17.16)$$

zusammengestellt.

In wenigen und einfachen Schritten haben wir mit Hilfe einer Kettenbruchentwicklung eine Folge gebildet die gegen den Grenzwert $\tan^{-1}(1)$ konvergiert und schnell präzise Resultate liefert.

17.5 Folgerungen

Die Arbeit soll ein kleinen Einblick in die Theorie der Kettenbrüche geben. Logarithmen, Kreisbögen, Quadraturen, andere Kurven und schwer greifbare Zahlen können als Kettenbrüche ausgedruckt werden. Rationale Zahlen können als endliche Kettenbrüche dargestellt werden und somit irrationale Zahlen als unendliche Kettenbrüche.

Ein weiterer Aspekt der Kettenbruchtheorie ist, dass quadratische Irrationalitäten endlich durch einen periodischen Kettenbruch dargestellt werden können.

Besonders irrationale Zahlen können durch Kettenbrüche eine Näherung des Endresultats liefern. Mit Hilfe der Kettenbruchtheorie können die Konvergenten von Funktionen schneller ermittelt werden und eine Abschätzung der gesuchten irrationale Zahl ermöglichen. Damit kann der Wunsch zur unendlichen Entwicklungen von Funktionen einen genauen Ausdruck gegeben werden.

Literatur

- [1] Kettenbrüche. 2008. URL: <https://www.mathi.uni-heidelberg.de/~thaeter/anasem08/Kettenbruch2.pdf>.
- [2] David Kincaid und Ward Cheney. *Numerical Analysis*. Bd. 2. American Mathematical Society, 2002. ISBN: 978-8-8218-4788-6.
- [3] Oskar Perron. *Die Lehre von den Kettenbrüchen*. Springer Fachmedien Wiesbaden, 1977. ISBN: 978-3-663-12289-0.

18

Padé-Approximation

Cédric Renda

18.1 Warum die Taylor-Reihe nicht gut genug ist

Für praktische Berechnungen von Modellen in der Physik, Ingenieurwissenschaften und weiteren Gebieten wird die Taylor-Reihe schon früh im Studium als ein nützliches Werkzeug gelehrt. Leider liefert die Taylor-Reihe nicht immer eine genügend gute Approximation. Dieses Paper beschäftigt sich mit der Padé-Approximation, welche aus gebrochen rationalen Funktionen besteht und oft in der Notation

$$R_{[L/M]} = [L/M] = \frac{P_L(x)}{Q_M(x)},$$

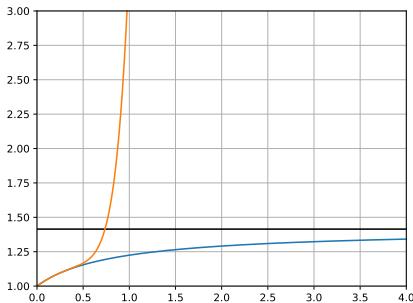
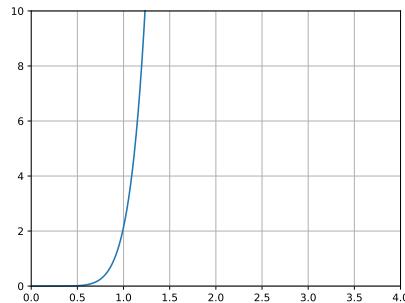
angetroffen wird. Die Definition dieser Notation wird im Abschnitt 18.2.2 erläutert. Padé-Approximationen sind Brüche, welche nur wenig komplizierter als Taylor-Reihen sind und ergänzen somit die Taylor-Reihe als Approximationsmethode. Die Padé-Approximation kann gewisse Funktionen besser approximieren als Polynome, denn Polynome divergieren bei grösser werdenden Werten immer, während Brüche beschränkte Funktionen approximieren können. Wenn die Taylor-Reihe keine genügenden Ergebnisse liefert, kann die Padé-Approximation verwendet werden, um die Approximation zu verbessern.

Das Problem der Taylor-Reihe ist, dass sie nicht immer gegen die Funktion konvergiert und das auch in der Nähe des Entwicklungspunktes. Die Funktion

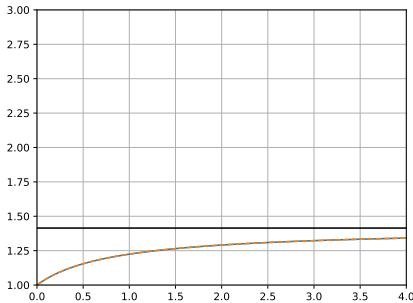
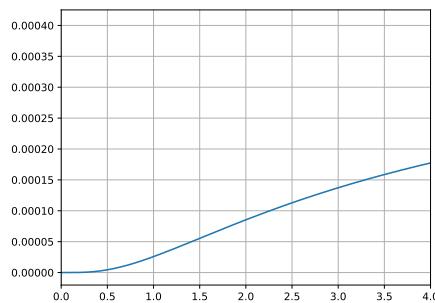
$$f(x) = \begin{cases} 0 & x \leq 0 \\ e^{-\frac{1}{x^2}} & 0 < x \end{cases} \quad (18.1)$$

hat eine Taylor-Reihe $Tf(x) = 0$. Nur für sogenannte analytische Funktionen erhält man die gewünschte $Tf(x) = f(x)$ in der Umgebung des Entwicklungspunktes.

In der Praxis wird die Funktion jedoch nur mit immer längeren Polynomen approximiert da wir keine unendlich lange Polynome verwenden können. Diese Vorgehensweise kann bei praktischen Problemen auch einen sehr unerwünschten und limitierenden Effekt haben.

(a) Plot von $f(x)$ und Taylor-Reihe 7. Ordnung.

(b) Fehler zwischen der Funktion und der Approximation.

(c) Plot von $f(x)$ und Padé-Approximation 3. Ordnung. (d) Fehler zwischen der Funktion und der Padé-Approximation 3. Ordnung.Abbildung 18.1: Visualisierung der Funktion $f(x)$ und ihren Approximationen

Beispiel. Schauen wir das Beispiel

$$f(x) = \left(\frac{1+2x}{1+x} \right)^{\frac{1}{2}} \approx 1 + \frac{1}{2}x - \frac{5}{8}x^2 + \frac{13}{16}x^3 - \frac{141}{128}x^4 + \frac{399}{256}x^5 - \frac{2353}{1024}x^6 + \frac{7205}{2048}x^7 + \dots$$

an.

Die originale Funktion $f(x)$ ist eine monoton wachsende, stetige Funktion, welche für $0 \leq x \leq \infty$ nur Werte zwischen eins und $\sqrt{2}$ beinhaltet. Die dazugehörige Taylor-Reihe mit dem Entwicklungspunkt $x_0 = 0$ konvergiert für $x > \frac{1}{2}$ nicht. Das Verhalten der originalen Funktion und deren Taylor-Reihe ist in dem Graphen 18.1(a) ersichtlich.

Die Padé-Approximation ist eine spezielle Art von rationalen Brüchen, welche eine Funktion approximiert. Diese Art der Approximation führt oft zu einem besseren Resultat als eine Taylor-Reihe. Manchmal können mit der Padé-Approximation auch dann gute Ergebnisse gewonnen werden, wenn eine Taylor-Reihe wie in diesem Beispiel nicht konvergiert.

Wenn man aus den Koeffizienten der Taylor-Reihe nach der im Abschnitt 18.2.2 gezeigten Me-

thode eine Padé-Approximation ermittelt, findet man

$$R_{[2/2]}(x) = \frac{P^{[2/2]}(x)}{Q^{[2/2]}(x)} = \frac{1 + \frac{13}{4}x + \frac{41}{16}x^2}{1 + \frac{11}{4}x + \frac{29}{16}x^2}. \quad (18.2)$$

Für immer grösser werdendes x ist der Grenzwert

$$\lim_{x \rightarrow \infty} \left(\frac{1 + \frac{13}{4}x + \frac{41}{16}x^2}{1 + \frac{11}{4}x + \frac{29}{16}x^2} \right) = \frac{49}{29} = 1.413793103,$$

man erhält ein Ergebnis, welches sehr nahe an der originalen Funktion liegt. In der Grafik 18.1(c) ist gut ersichtlich, wie viel besser sich die Padé-Approximation im Vergleich zu der Taylor-Approximation verhält. Betrachtet man die beiden Fehler der Taylor-Reihe 18.1(b) und der Padé-Approximation 18.1(d) zur originalen Funktion, kann man sehen, dass die Padé-Approximation deutlich besser ist. Dies obwohl von der originalen Funktion nicht mehr Informationen bekannt waren als bei der Taylor-Reihe, da die Padé-Approximation ja aus der Taylor-Reihe erstellt wurde. ○

In den folgenden Abschnitten wird nun inklusive einiger Beispiele aufgezeigt, wie man eine Padé-Approximation erstellt. Zum besseren Verständnis wird zuerst erklärt wie die Potenzreihe einer Funktion gefunden werden kann.

18.2 Berechnung der Padé-Approximation

Eine Padé-Approximation ist ein Bruch aus zwei Polynomen, welche aus den Koeffizienten der Taylor-Reihe einer Funktion gewonnen werden. Ziel dieses Kapitels ist, dem Leser den Nutzen der Padé-Approximation näher zu bringen und zu zeigen, wie man aus einer analytischen Funktion eine solche Approximation bildet. Des Weiteren wird auf mehrere praktische Beispiele eingegangen und mögliche Fehlerquellen aufgezeigt.

18.2.1 Potenzreihen von analytischen Funktionen

Die Koeffizienten einer Potenzreihe können verwendet werden, um die Koeffizienten einer Padé-Approximation zu gewinnen. In der Analysis kann eine analytische Funktion mit einer Taylor-Reihe um eine Stelle x_0 durch eine Potenzreihe dargestellt werden. Diese Potenzreihen werden um eine vorgegebene Stelle x_0 als eine unendliche Summe

$$f(x) = \sum_{n=0}^{\infty} c_n(x - x_0)^n \quad (18.3)$$

gebildet. Für viele Funktionen sind die dazugehörigen Potenzreihen bereits bekannt. Funktionen, welche durch eine Potenzreihe dargestellt werden können, werden auch analytische Funktionen genannt. Es gibt verschiedene Methoden, um eine Potenzreihe einer Funktion zu erhalten. Sie können durch Differentialgleichungen oder eine andere Reihenentwicklungsmethoden hergeleitet werden.

Beispiel. In diesem Beispiel möchten wir die Potenzreihe der Exponentialfunktion erhalten, welche an der Stelle $x_0 = 0$ entwickelt wird. Um dies zu erreichen wird ein Potenzreihenansatz verwendet. Dabei wird die gesuchte Funktion durch eine Potenzreihe mit unbekannten Koeffizienten dargestellt,

diese Koeffizienten werden in eine Differentialgleichung eingesetzt und dann wird versucht mit Koeffizientenvergleich eine Lösung zu finden.

Wir wissen, welche Eigenschaften die Exponentialfunktion hat, und nutzen diese, um die Differentialgleichungen aufzustellen. Gesucht ist eine Potenzreihe, welche die Differentialgleichung

$$f'(x) = f(x), \quad \text{für alle } x \in \mathbb{R}$$

erfüllt.

Die Form der Potenzreihe ist gegeben durch die schon gezeigte Summe (18.3). Weiter wissen wir, dass die Exponentialfunktion $f(0) = 1$ erfüllen muss. Daraus schliessen wir, dass

$$f(x) = \sum_{n=0}^{\infty} c_n \cdot x^n \Rightarrow \sum_{n=0}^{\infty} c_n \cdot 0^n = c_0 \cdot 0^0 + c_1 \cdot 0^1 + c_2 \cdot 0^2 \dots = 1 \Rightarrow c_0 = 1$$

und somit ist der erste Koeffizient c_0 gefunden. Anschliessend wird die Reihe abgeleitet

$$f'(x) = \sum_{n=0}^{\infty} (n+1) \cdot c_{n+1} \cdot x^n,$$

um die weiteren Koeffizienten c_n zu erhalten. Da immer noch die Anforderung $f(x) = f'(x)$ gilt, wissen wir, dass die Koeffizienten von x^n von der Form

$$(n+1) \cdot c_{n+1} = c_n, \quad \text{für alle } n \in \mathbb{R}$$

sein müssen. Mit dem bereits bekannten $c_0 = 1$ folgt rekursiv

$$c_{n+1} = \frac{1}{(n+1)!} \Rightarrow c_n = \frac{1}{n!}$$

und damit sind die Koeffizienten der Potenzreihe der Exponentialfunktion ermittelt. Ausgeschrieben erhält man

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \quad (18.4)$$

Dieses Vorgehen wird auch Potenzreihenansatz für eine Differentialgleichung genannt und ist ein allgemeiner Lösungsansatz für Differentialgleichungen.

Aus der bekannten Potenzreihe der Exponentialfunktion (18.4) können nun auch andere Potenzreihen gewonnen werden. Aus der Exponentialfunktion kann die Potenzreihe für die Sinus- und Kosinusfunktion ermittelt werden. Dies kann durch beispielsweise durch die Koeffizientenvergleichsmethode erreicht werden. Um dies zu erreichen wird, zuerst die Reihe komplex erweitert

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \Rightarrow e^{ix} = \sum_{n=0}^{\infty} \frac{(ix)^n}{n!}$$

und danach ausgeschrieben damit die Koeffizienten in

$$e^{ix} = \sum_{n=0}^{\infty} \frac{(ix)^n}{n!} = \frac{(ix)^0}{0!} + \frac{(ix)^1}{1!} + \frac{(ix)^2}{2!} + \frac{(ix)^3}{3!} + \dots \quad (18.5)$$

betrachtet werden können. Man erkennt gleich, dass die imaginären Zahlen bei den geraden Exponenten verschwinden und bei den ungeraden Exponenten bestehen bleiben. Sortiert man nun die Reihe nach imaginären und reellen Termen, erkennt man die cis-Form

$$\begin{aligned} e^{ix} &= \left(1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots\right) + i \cdot \left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots\right) \\ &= \cos(x) + i \cdot \sin(x) = \text{cis}(x). \end{aligned}$$

Man kann diese beiden Teile als Sinus und Kosinus aufschreiben, wobei die Sinusfunktion

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

und die Kosinusfunktion

$$\cos(x) = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

beide als einzelne Summen ausgeschrieben werden können. ○

18.2.2 Padé-Approximation erstellen

Um eine Padé-Approximation zu erstellen, startet man immer mit der Potenzreihe der zu approximierenden Funktion. Diese Potenzreihe der Form

$$f(x) = \sum_{n=0}^{\infty} c_n z^n \tag{18.6}$$

möchte man dann durch einen rationalen Bruch

$$R_{[L/M]} = [L/M] = \frac{a_0 + a_1 z + \dots + a_L z^L}{b_0 + b_1 z + \dots + b_M z^M} + O(z^{L+M+1}) \tag{18.7}$$

beschreiben. Wir verwenden nun für den Approximanten die Notation $R_{[L/M]}$. Der erste Koeffizient des Nennerpolynoms wird mit $b_0 = 1$ festgelegt. Dank dieser Definition haben wir $L+1$ zu bestimmende Zähler- und M zu bestimmende Nennerkoeffizienten. Dies resultiert in $L+M+1$ unbekannten Koeffizienten, welche mit der Nummerierung der Terme $z^0, z^1, z^2, \dots, z^{L+M}$ der Potenzreihe übereinstimmt.

Um die einzelnen a_k (mit $0 \leq k \leq L$) und b_k (mit $1 \leq k \leq M$) Koeffizienten zu berechnen, wir die Approximation mit der Potenzreihe gleichgesetzt

$$\sum_{n=0}^{\infty} c_n z^n = \frac{a_0 + a_1 z + \dots + a_L z^L}{b_0 + b_1 z + \dots + b_M z^M} + O(z^{L+M+1}). \tag{18.8}$$

Das Ganze wird umgestellt und als lineares Gleichungssystem

$$\left(\sum_{n=0}^{\infty} c_n (z)^n \right) \cdot (b_0 + b_1 z + \dots + b_M z^M) = (a_0 + a_1 z + \dots + a_L z^L) + O(z^{L+M+1}) \tag{18.9}$$

für einen späteren Koeffizientenvergleich betrachtet. Diese linearen Gleichungen können in der Matrixform

$$\begin{pmatrix} c_{L-M+1} & c_{L-M+2} & c_{L-M+3} & \dots & c_L \\ c_{L-M+2} & c_{L-M+3} & c_{L-M+4} & \dots & c_{L+1} \\ c_{L-M+3} & c_{L-M+4} & c_{L-M+5} & \dots & c_{L+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_L & c_{L+1} & c_{L+2} & \dots & c_{L+M-1} \end{pmatrix} \cdot \begin{pmatrix} b_M \\ b_{M-1} \\ b_{M-2} \\ \vdots \\ b_1 \end{pmatrix} = - \begin{pmatrix} c_{L+1} \\ c_{L+2} \\ c_{L+3} \\ \vdots \\ c_{L+M} \end{pmatrix}$$

aufgeschrieben und nach den b_k -Koeffizienten aufgelöst werden. Die dazugehörigen a_k -Koeffizienten können dann aus den b_k und den c_n (mit $0 \leq n \leq M+L$) Koeffizienten mittels Koeffizientenvergleich

$$\begin{aligned} a_0 &= c_0 \\ a_1 &= c_1 + b_1 c_0 \\ a_2 &= c_2 + b_1 c_1 + b_2 c_0 \\ &\vdots \\ a_L &= c_L + \sum_{n=1}^{\min(L,M)} b_n c_{L-n} \end{aligned} \tag{18.10}$$

berechnet werden. Beim Lösen des Gleichungssystems können sich vor allem für grosse L und M numerische Fehler einschleichen, weil die Koeffizienten der Potenzreihe möglicherweise sehr klein werden. Gelingt die Lösung des Gleichungssystems, hat man einen $[L/M]$ -Approximanten konstruiert. Dieser Approximant stimmt mit

$$\sum_{n=0}^{\infty} c_n z^n \tag{18.11}$$

bis zu der Ordnung $z^{[L+M]}$ überein.

Beispiel. Nehmen wir das Beispiel der Einleitung,

$$f(x) = \left(\frac{1+2x}{1+x} \right)^{\frac{1}{2}} \approx 1 + \frac{1}{2}x - \frac{5}{8}x^2 + \frac{13}{16}x^3 - \frac{141}{128}x^4 + \frac{399}{256}x^5 - \frac{2353}{1024}x^6 + \frac{7205}{2048}x^7 \mp \dots \tag{18.12}$$

von welchem wir nun den Padé-Approximanten der dritten Ordnung $R_{[3/3]}$ berechnen möchten. Zuerst kreieren wir die Matrix, um die b_k Koeffizienten zu berechnen.

$$\begin{pmatrix} c_1 & c_2 & c_3 \\ c_2 & c_3 & c_4 \\ c_3 & c_4 & c_5 \end{pmatrix} \cdot \begin{pmatrix} b_3 \\ b_2 \\ b_1 \end{pmatrix} = - \begin{pmatrix} c_4 \\ c_5 \\ c_6 \end{pmatrix}$$

Hier können nun die Koeffizienten der Potenzreihe 18.12 eingesetzt werden.

$$\begin{pmatrix} \frac{1}{2} & -\frac{5}{8} & \frac{13}{16} \\ -\frac{5}{8} & \frac{13}{16} & -\frac{141}{128} \\ \frac{13}{16} & -\frac{141}{128} & \frac{399}{256} \end{pmatrix} \cdot \begin{pmatrix} b_3 \\ b_2 \\ b_1 \end{pmatrix} = - \begin{pmatrix} -\frac{141}{128} \\ \frac{399}{256} \\ -\frac{2353}{1024} \end{pmatrix}$$

Da es sich in diesem Fall um eine quadratische Matrix handelt, kann die Inverse der C -Matrix

$$C \cdot b = -c \quad \Rightarrow \quad b = C^{-1} \cdot -c \quad (18.13)$$

genommen werden, um die Gleichung

$$\begin{pmatrix} b_3 \\ b_2 \\ b_1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & -\frac{5}{8} & \frac{13}{16} \\ -\frac{5}{8} & \frac{13}{16} & -\frac{141}{128} \\ \frac{13}{16} & -\frac{141}{128} & \frac{399}{256} \end{pmatrix}^{-1} \cdot \begin{pmatrix} \frac{141}{128} \\ -\frac{399}{256} \\ \frac{2353}{1024} \end{pmatrix} = \begin{pmatrix} \frac{169}{64} \\ \frac{47}{8} \\ \frac{17}{4} \end{pmatrix}$$

zu lösen. Die a_k Koeffizienten können nun aus den b_k und den c_n Koeffizienten berechnet werden:

$$\left. \begin{array}{l} a_0 = c_0 \\ a_1 = c_1 + b_1 c_0 \\ a_2 = c_2 + b_1 c_1 + b_2 c_0 \\ a_3 = c_3 + b_1 c_2 + b_2 c_1 + b_3 c_0 \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} a_0 = 1 \\ a_1 = \frac{1}{2} + \frac{17}{4} \cdot 1 = \frac{19}{4} \\ a_2 = -\frac{5}{8} + \frac{17}{4} \cdot \frac{1}{2} + \frac{47}{8} \cdot 1 = \frac{59}{8} \\ a_3 = \frac{13}{16} + \frac{17}{4} \cdot -\frac{5}{8} + \frac{47}{8} \cdot \frac{1}{2} + \frac{169}{64} \cdot 1 = \frac{239}{64} \end{array} \right. \quad (18.14)$$

Daraus resultiert der gesamte Padé-Approximant der dritten Ordnung

$$R_{[3/3]}(x) = \frac{a_0 + a_1 x + a_2 x^2 + a_3 x^3}{1 + b_1 x + b_2 x^2 + b_3 x^3} = \frac{1 + \frac{19}{4}x + \frac{59}{8}x^2 + \frac{239}{64}x^3}{1 + \frac{17}{4}x + \frac{47}{8}x^2 + \frac{169}{64}x^3}, \quad (18.15)$$

welcher aus dem Taylor-Polynom sechster Ordnung gewonnen werden konnte.

Auffallend ist, dass dieser Approximant der dritten Ordnung aus komplett anderen Polynomen zusammengesetzt ist, als der Approximant zweiter Ordnung (18.2) des einführenden Beispiels. ○

Dieser Aufwand, um einen Approximanten zu berechnen, muss jedoch nicht immer betrieben werden. Für manche Funktionen können Formeln zur Konstruktion der Nenner- und Zählerpolynome der Padé-Approximanten gefunden werden. Wie man auf diese Formeln kommt, ist zum Beispiel in [1] beschrieben. Die Exponentialfunktion, welche für viele Anwendungen gebraucht wird, besitzt glücklicherweise eine solche Formel für die Konstruktion der Nenner- und Zählerpolynome [2]. Um eine Padé-Approximation beliebiger Ordnung

$$R_{[L/M]}(x) = \frac{P_{[L/M]}(x)}{Q_{[L,M]}(x)} \approx e^{-x} \quad (18.16)$$

für die Exponentialfunktion zu erhalten, können die zwei gegebenen Formeln aus [2] verwendet werden. Für $P_{[L/M]}(x)$, also das Zählerpolynom, gilt die Formel

$$P_{[L/M]}(x) = \sum_{n=0}^L \frac{(L+M-n)!L!}{(L+M)!n!(M-n)!} (-x)^n \quad (18.17)$$

und für $Q_{[L/M]}(x)$, das Nennerpolynom, gilt

$$Q_{[L/M]}(x) = \sum_{n=0}^M \frac{(L+M-n)!M!}{(L+M)!n!(M-n)!} x^n. \quad (18.18)$$

Es ist gängige Praxis, die Padé-Approximanten in einer Tabelle mit der Form

$[L/M]$	0	1	2	\dots
0	$[0/0]$	$[1/0]$	$[2/0]$	\dots
1	$[0/1]$	$[1/1]$	$[2/1]$	\dots
2	$[0/2]$	$[1/2]$	$[2/2]$	\dots
\vdots	\vdots	\vdots	\vdots	

darzustellen. Diese Darstellung dient zur Übersicht der einzelnen Approximanten und macht deutlich, dass alle Approximanten verschiedener Ordnung unabhängig von einander sind. Das heisst die Polynome sehen bei jeder Ordnung komplett verschieden aus und werden nicht wie bei der Taylor-Reihe einfach immer länger.

In diese Tabelle können nun die einzelnen Approximanten einer Funktion eingefüllt werden. Als Beispiel wurden die $[L/M]$ Approximanten der Exponentialfunktion bis zum Approximant $[2/2]$ mit Hilfe der Formeln 18.17 und 18.18 berechnet und in die Tabelle

$[L/M]$	0	1	2
0	$\frac{1}{1}$	$\frac{1+z}{1}$	$\frac{2+2z+z^2}{2}$
1	$\frac{1}{1-z}$	$\frac{2+z}{2-z}$	$\frac{6+4z+z^2}{6-2z}$
2	$\frac{2}{2-2z+z^2}$	$\frac{6+2z}{6-4z+z^2}$	$\frac{12+6z+z^2}{12-6z+z^2}$

eingefügt.

18.3 Anwendungen

In diesem Abschnitt werden ein paar Anwendungsmöglichkeiten der Padé-Approximation gezeigt. Wie in der Signalverarbeitung in der Elektrotechnik üblich, werden des öfteren mathematische Probleme im Frequenzbereich gelöst, da dies oft einfacher ist. Die Transformation in das geeignete Spektrum kann bei kontinuierlichen Signalen mit der Laplace-Transformation und bei diskreten Signalen mit der Z-Transformation erreicht werden. Unter dem Begriff Übertragungsfunktion, versteht man das Verhältnis von Ausgangssignal zu Eingangssignal. Die Abhängigkeit der verschiedenen Zustandsräumen wird in Abbildung 18.2 mit dem Beispiel der Übertragungsfunktion $h(t)$ dargestellt. Dabei gelten $s = \sigma + j\omega$ und $z = Ae^{j\omega} = \sigma + j\omega$. Der jeweilige Spektralbereich wird auch oft Bildbereich genannt.

18.3.1 Totzeit Approximation

Bei der Totzeit spricht man von einer Verzögerung im Zeitbereich, welche zum Beispiel in einem digitalen Regelkreis vorkommen kann. Lineare zeitinvariante (LZI)-Systeme oder besser bekannt unter dem englischen Begriff linear time-invariant (LTI) system, sind unabhängig von zeitlichen Verschiebungen. Die Eingangs-Ausgangsbeziehung von LTI-Systemen, ist durch die zeitdiskrete Faltung der Impulsantwort des Systems mit dem Eingangssignal gegeben. LTI-Systeme können in zwei Typen eingeteilt werden. Der erste Typ ist die finite impulse response (FIR) mit endlicher

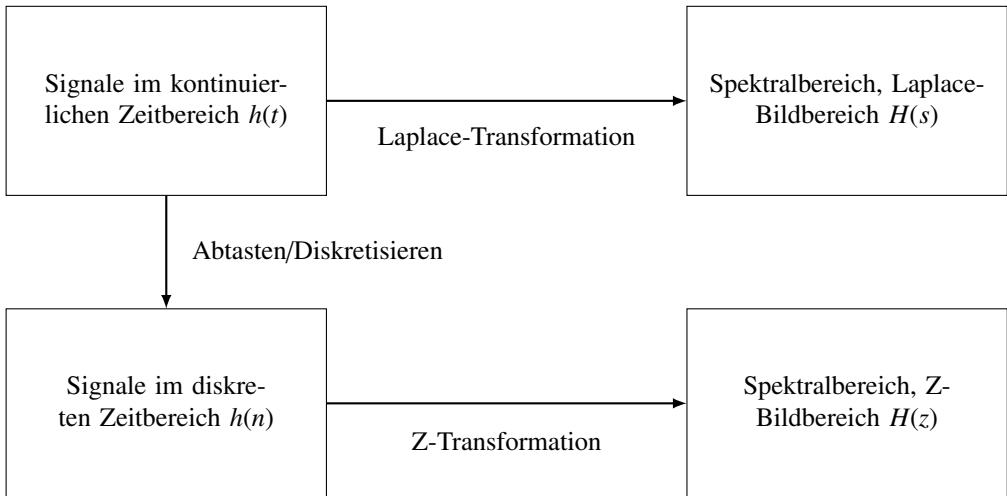


Abbildung 18.2: Zeit- und Spektralbereich für stetige und diskrete Signale

Impulsantwort. Typ 2 hat eine unendlich lange Impulsantwort, genannt die infinite impulse response (IIR). Diese zwei Typen unterteilen Impulsantworten anhand der Eigenschaft ob sie eine endliche oder unendliche Dauer hat. Wer sich mit LTI Systemen auseinander gesetzt hat, weiss, dass eine Totzeit T im Frequenzbereich durch eine Exponentialfunktion

$$H(s) = e^{-sT}$$

ausgedrückt wird.

Verschiedene Methoden wie z.B. die Wurzelortskurven können nicht mit einer Totzeit umgehen, sondern nur mit Übertragungsfunktionen, welche normalerweise als gebrochen rationale Funktionen vorliegen. Die Übertragungsfunktion besteht aus einem Nenner- und Zählerpolynom welches der Form

$$H(s) = \frac{P(s)}{Q(s)}$$

entspricht. Wir bezeichnen die Grade von $P(s)$ und $Q(s)$ mit p und q .

Beim Design von Filtern, Reglern oder anderen frequenzabhängigen Netzwerken, verwendet man die Laplace- oder Z-Transformierte der Übertragungsfunktion, um die Eigenschaften des zu entwerfenden Netzwerkes zu synthetisieren oder zu analysieren. Dazu benötigt man nun die Übertragungsfunktion in Form von gebrochenen Polynomen, mit welchen man zum Beispiel die Pol- und Nullstellen des Netzwerkes erkennt. Mit diesen Pol- und Nullstellen können Aussagen über den Betrags- und Phasenverlauf des Frequenzgangs, Impuls- und Sprungantwort, Stabilität eines Systems und weitere Eigenschaften gemacht werden. Mit diesen Informationen kann entschieden werden, welche Parameter geändert werden müssen um die gewünschten Eigenschaften zu erhalten.

Diese Netzwerke werden oft mit passiven Bauelementen wie Widerständen, Induktivitäten und Kapazitäten und einem Operationsverstärker realisiert. Ein PID-Glied kann so zum Beispiel in vereinfachter Form wie in Abbildung 18.3 konstruiert werden. Zu diesem Beispiel können nun die

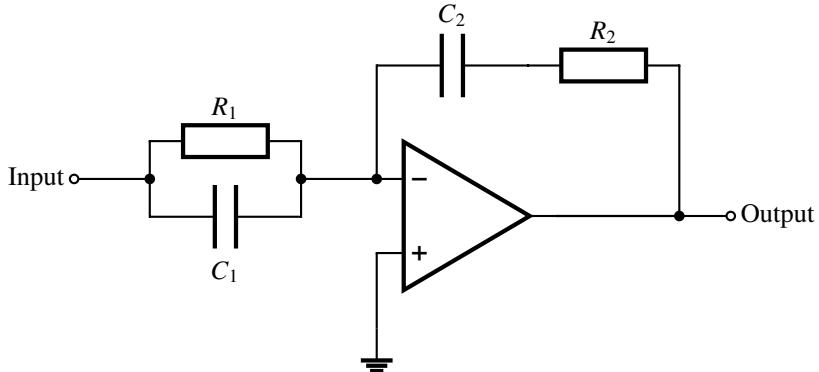


Abbildung 18.3: Realisierung eines PID-Gliedes mit Operationsverstärker, Widerständen und Kapazitäten.

dazugehörigen Eingangs- und Ausgangsimpedanzen

$$Z_0(s) = \frac{1 + sC_2R_2}{sC_2}$$

$$Z_1(s) = \frac{R_1}{1 + sC_1R_1}$$

berechnet werden. Für die Übertragungsfunktion erhält man damit

$$H(s) = \frac{Z_0(s)}{Z_1(s)} = \frac{1 - C_2R_2C_1R_1 + sC_2R_2 + sC_1R_1}{sC_2R_2}.$$

Die zwei Polstellen befinden sich somit bei $s = 0$ was einem DC entsprechen würde. Die Nullstellen befinden sich bei

$$s = \frac{C_1R_1C_2R_2 - 1}{C_1R_1 + C_2R_2},$$

abhängig von den Komponenten. Die Erkenntnisse kann man nun für die Dimensionierung der Bauteile oder für Aussagen darüber verwenden, für welche Anwendungen dieses PID-Glied nicht geeignet ist.

Leider kann mit einem passiven Bauteil keine Exponentialfunktion im Frequenzbereich nachgebaut werden und somit auch keine Totzeit. In der Praxis wird deshalb die Totzeit oft mit digitalen Bausteinen realisiert, da man digital einfach die Werte speichern kann und zur gewünschten Zeit wieder ausgeben. Mit e^{-sT} in einem analogen System haben wir jedoch keine brauchbaren Pol-Nullstellen welche mit passiven Bauteilen nachgebaut werden können. Dies kann jedoch gefordert sein wenn man eine zeit-kontinuierliche Analyse macht oder einfach keine Digitalbausteine in seiner Schaltung verwenden will. Eine Lösung dafür könnte nun sein, e^{-sT} zu approximieren und die Approximation für weitere Rechnungen zu verwenden.

Die Padé-Approximation der Exponentialfunktion ist glücklicherweise schon bekannt und kann mit den Formeln (18.17) und (18.18) für eine beliebig hohe Ordnung ermittelt werden. Die Qualität dieser Approximation wird ersichtlich, wenn die Padé-Approximanten in einer logarithmischen Skala mit der originalen Funktion verglichen. In der Grafik 18.4 zeigt sich, dass die Padé-Approximation mit steigendem Grad bessere Resultate liefert und sich der Exponentialfunktion mehr und mehr annähert.

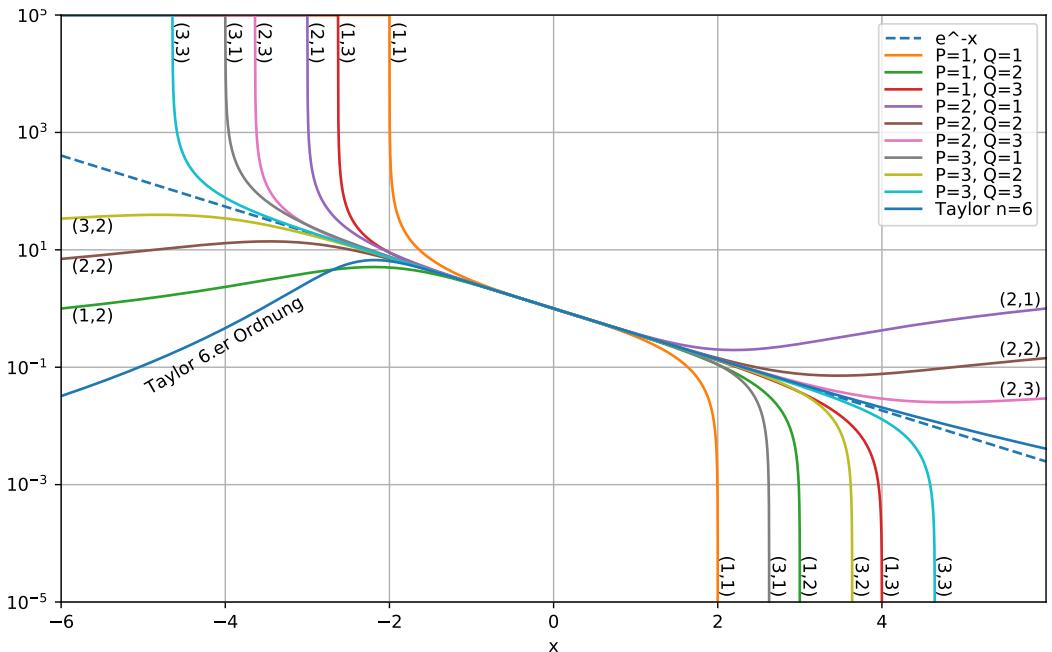


Abbildung 18.4: Plot der Padé-Approximanten der Exponentialfunktion im Vergleich zu der Originalfunktion in einer logarithmischen Skala vergleichend mit der Taylor-Polynom sechster Ordnung der Exponentialfunktion.

Im Beispiel 18.4 mit der Exponentialfunktion erweisen sich die Padé-Approximanten des selben Grades im Nenner und Zähler als die genauesten Approximationen. Es wird dabei mit steigendem Grad der Approximation eine stetige Verbesserung erzielt. Zum Vergleich ist noch das Taylor-Polynom sechster Ordnung von e^{-x} in der Grafik 18.4 dargestellt. Diese ist im negativen Bereich der Funktionsachse deutlich schlechter als die Padé-Approximanten, jedoch verhält sie sich im positiven Bereich besser.

Transformieren wir nun diese gebrochenen Polynome, welche sich noch im Frequenzbereich befinden zurück in den Zeitbereich, erhalten wir die in der Grafik 18.9 dargestellten Kurven. Die Polynome der Ordnung [3/3] sind dabei noch weit von einem brauchbaren Ergebnis des gesuchten verzögerten Einheitssprunges entfernt. Man kann nun die Ordnung der Padé-Approximanten weiter erhöhen bis eine zufriedenstellende Approximation vorliegt. Dabei muss jedoch aufgepasst werden, welche Methode für die Rücktransformation verwendet wird. Nicht alle Implementierungen der Rücktransformation von Übertragungsfunktionen in ein Zustandsraummodell sind numerisch stabil. Bei Polynomen grosser Ordnung oder wenn das Nenner- und Zähler-Polynom nicht gleicher Ordnung sind, können bei einer Rücktransformation Fehler auftauchen. Dies ist jedoch ein anderes Thema, welches den Rahmen dieses Papers sprengen würde.

In den Abbildungen 18.5 und 18.6 sind nun die Pol- und Nullstellen von höheren Ordnungen dargestellt. Auffallend ist die schöne Symmetrie der Pol- und Nullstellen, welche bei gleicher Ordnung der Nenner- und Zählerpolynome um den Nullpunkt verteilt sind. Diese Symmetrie verschwindet wenn der Grad des Nenner- oder des Zählerpolynoms grösser gewählt wird. Dies kann man soweit treiben, bis eine Polstelle in den rechten Bereich der komplexen Halbebene reicht 18.6, was ein

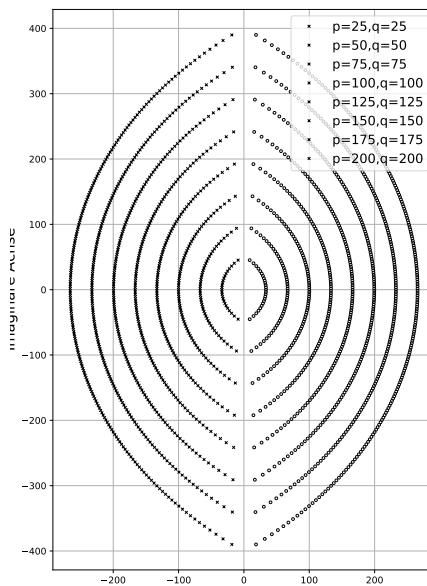


Abbildung 18.5: Pol- und Nullstellen in der komplexen Ebene wenn $p = q$ ist.

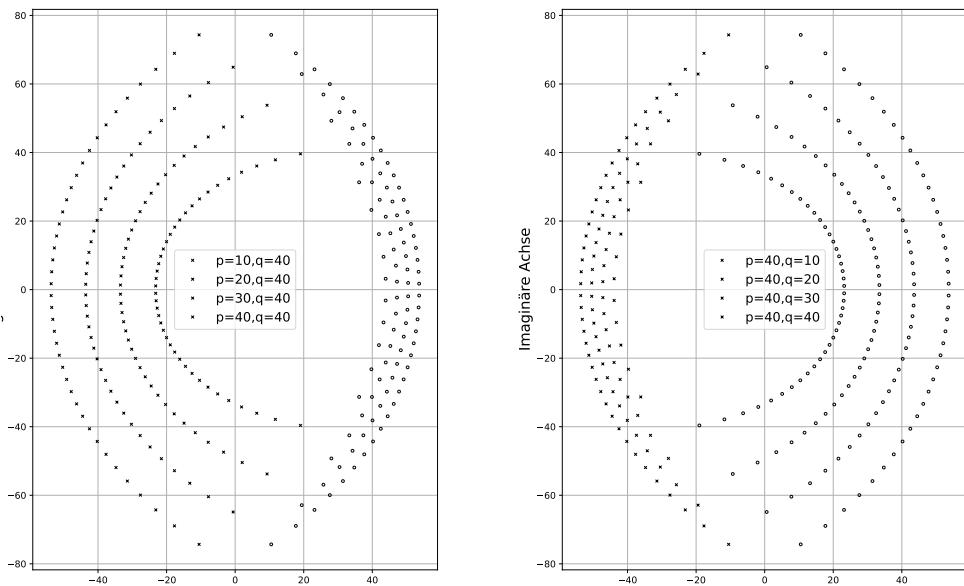


Abbildung 18.6: Pol- und Nullstellen in der komplexen Ebene wenn $p < q$ (links) und $p > q$ (rechts) ist.

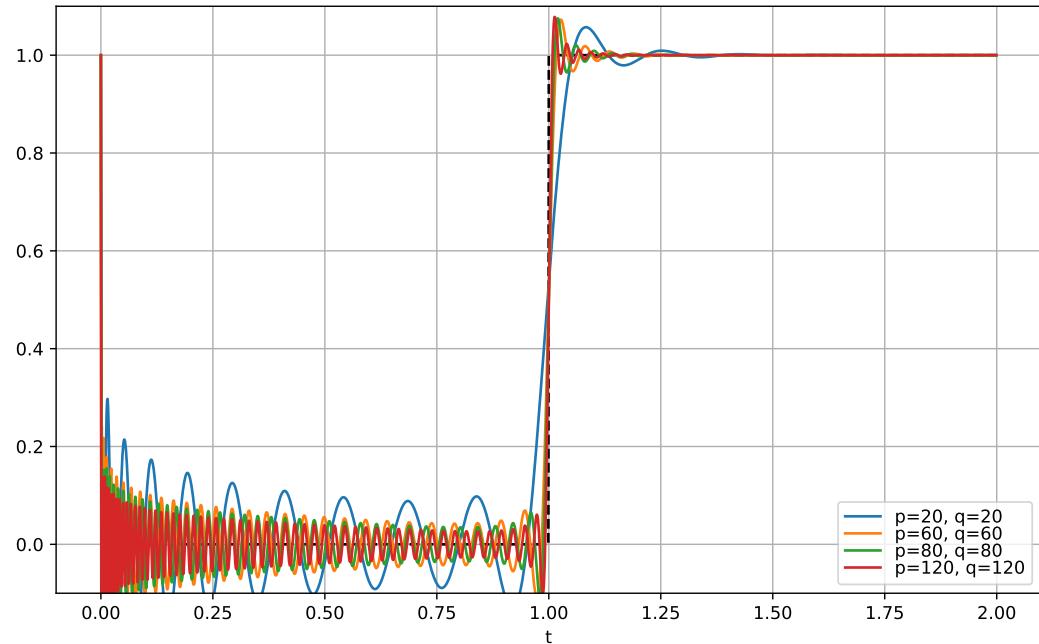


Abbildung 18.7: Rücktransformation der Padé-Approximanten im Zeitbereich geplottet wenn $p = q$ ist.

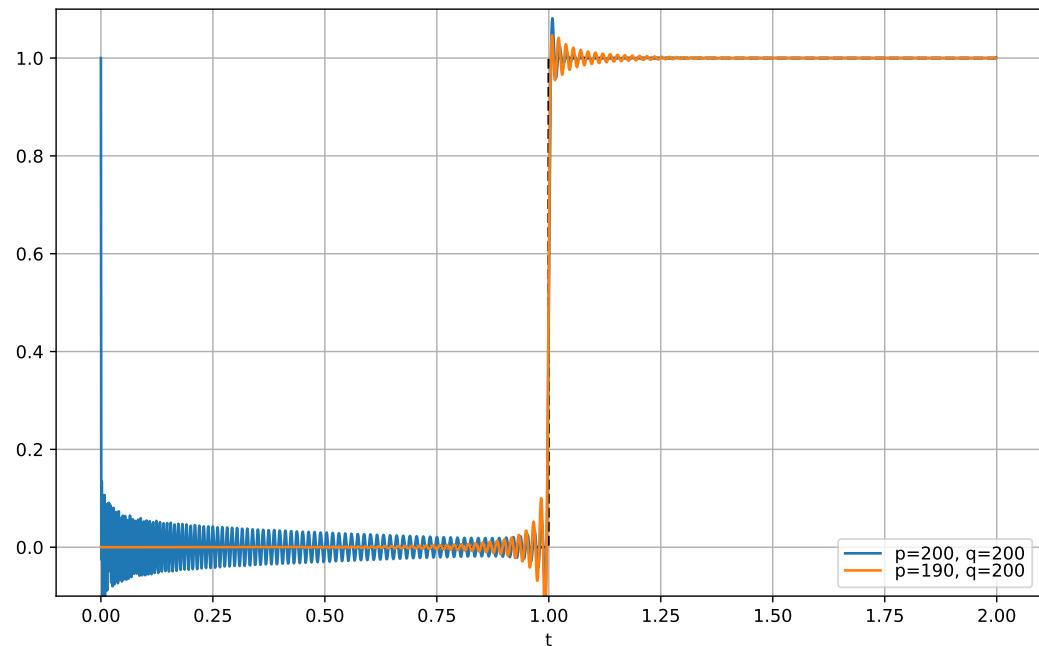


Abbildung 18.8: Rücktransformation der Padé-Approximanten im Zeitbereich geplottet wenn $p = q - 10$ ist.

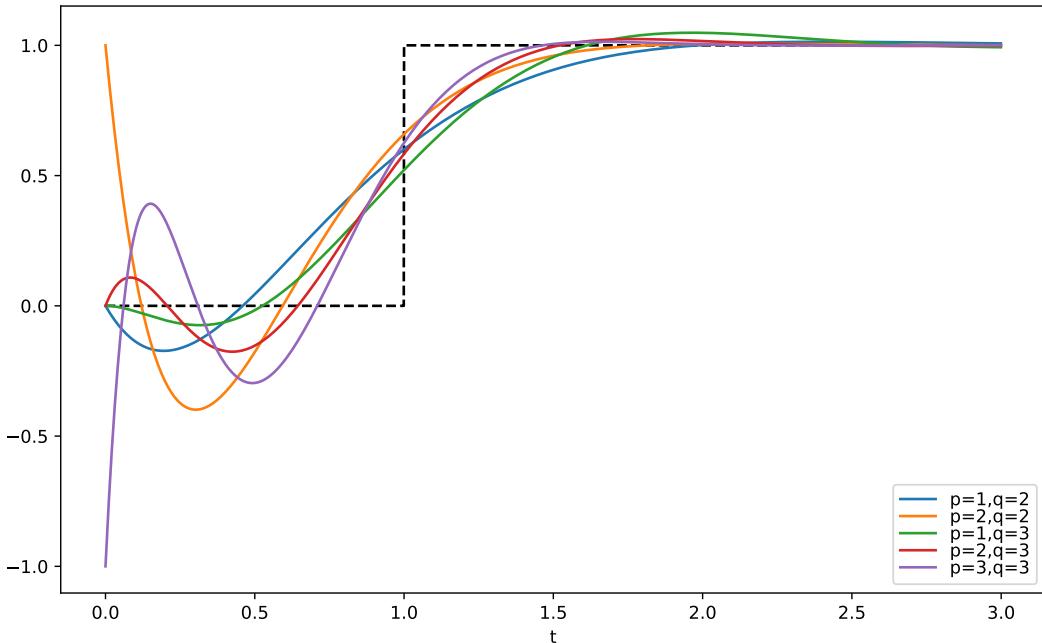


Abbildung 18.9: In den Zeitbereich rücktransformierte Padé-Approximanten tieferer Ordnungen.

instabiles System zur Folge hat. Ein instabiles System bedeutet hier, dass der rücktransformierte Einheitssprung eine anwachsende Schwingung ist.

Verwendet man nun die Approximanten höherer und gleicher Ordnung für die Rücktransformation in Grafik 18.7, erhält man einen immer besser werdenden Einheitssprung mit einem kleinen Überschwinger vor und nach dem Sprung. Auffallend ist hier, dass der Padé-Approximant höchster Ordnung zu Beginn den grössten Fehlerpeak hat. Dieser ist sogar so hoch wie der Einheitssprung selbst.

Um diesen Fehlerpeak zu Beginn ein wenig zu vermindern, können nun die Pole in der komplexen Ebene ein wenig näher zur positiven Halbebene gebracht werden. Wie schon in dem Pol-Nullstellendiagramm 18.6 erkennbar, ist das mit einer Differenz von der Ordnung des Zählerpolynoms p und Nennerpolynom q möglich. Mit einer Differenz von 190 zu 200 der Polynom Ordnung wurde die Rücktransformierte des Padé-Approximanten der Grafik 18.8 dargestellt. Deutlich zu sehen ist dabei, dass der Überschwinger, welcher bei dem Padé-Approximanten gleicher Ordnung in derselben Grafik 18.8 noch vorhanden ist, verschwindet. Jedoch besitzt der Approximant ungleicher Ordnung eine längere Einschwingzeit als der mit gleicher Ordnung. Mit diesen Werten der Ordnungen kann in der Praxis solange optimiert werden, bis man die beste Lösung für das gegebene Problem gefunden hat.

18.3.2 Digitale Signalmodellierung

Das Ziel einer Signalmodellierung ist, eine parametrische Beschreibung eines Signales zu finden. Dies kann für die Filterentwicklung, Interpolation, Extrapolation oder Kompression verwendet werden. Beispielsweise soll ein Audiosignal $x(n)$, welches N Werte $[x(0), \dots, x(N-1)]$ lang ist, soll gespeichert oder übertragen werden. Eine Möglichkeit ist es das komplette diskrete Signal, in einem

Übertragungskanal zu übertragen. Wenn es jedoch möglich ist, dieses Signal mit einer kleineren Anzahl k von Parametern zu modellieren, wobei $k \ll N$, ist es effizienter, diese Parameter zu senden statt das komplette diskrete Signal.

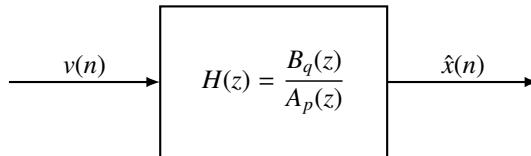
Als ein einfaches Beispiel kann ein Sinuston verwendet werden. Das Audiosignal $x(n)$ wäre in diesem Fall $x(n) = a \cos(n\omega_0 + \phi)$. Würde man nun dieses Signal speichern oder über einen Übertragungskanal versenden, und würde einfach alle Samples $x(n)$ verwenden, wäre dies sehr ineffizient. Eine bessere Methode ist die Amplitude a , die Frequenz ω_0 und die Phase ϕ aus dem Audiosignal zu messen und diese Parameter für die Speicherung oder Übertragung zu verwenden. Mit diesen Parametern kann das Signal $x(n)$ später perfekt rekonstruiert werden.

Bei komplizierteren Signalen verwendet man einfach mehr Parameter.

$$x(n) \approx \sum_{k=1}^L a_k \cos(n\omega_k + \phi_k) \quad (18.19)$$

Die dazugehörigen Modellparameter ($a_k, \omega_k, \phi_k, k = 1, \dots, L$) können gemessen und übertragen werden. Die Schwierigkeit ist ein gutes Verhältnis zwischen Effizienz und Genauigkeit zu finden.

In diesem Abschnitt verwenden wir immer das gleiche Modell, welches den Ausgang $x(n)$ eines linearen verschiebungsinvarianten Filters zu einem Eingang $v(n)$ darstellt.



Dieses Modell wird mit einer rationalen Funktion

$$H(z) = \frac{B_q(z)}{A_p(z)} = \frac{\sum_{k=0}^q b_q(k)z^{-k}}{1 + \sum_{k=1}^p a_p(k)z^{-k}} \quad (18.20)$$

beschrieben. Der Fehler e' dieses Modells, ist durch die Differenz des originalen Signales und dem Ausgang des approximierenden Modells

$$e'(n) = x(n) - \hat{x}(n) \quad (18.21)$$

definiert. Das Ziel ist, einen Filter $H(z)$ und einen Input $v(n)$ zu finden, welche $\hat{x}(n)$ so nahe wie möglich an $x(n)$ bringen.

Wenn man ein Modell gewählt hat, möchte man die besten Parameter oder auch die beste Approximation für sein Modell finden. Es gibt jedoch sehr viele Arten, die beste Approximation zu definieren und damit auch unzählige Methoden um die Modellparameter zu erhalten. Deshalb ist es wichtig, wenn man eine Signalmodellierung angeht, eine Methode zu finden, welche nicht nur funktioniert und gute Werte liefert, sondern auch effizient berechenbar ist. Dies ist besonders bei Systemen zur Echtzeitdatenverarbeitung ausschlaggebend.

Um eine solche Approximation zu finden ist die erste Intuition oft den Fehler minimieren zu wollen. Dies nennt sich auch die Least-Squares-Methode. Ziel ist es dabei den quadratischen Fehler

$$E_{LS} = \sum_{n=0}^{\infty} |e'(n)|^2 \quad (18.22)$$

des Systems zu minimieren.

Aus den Bedienungen der Least Squares Method resultiert eine Menge von nichtlinearen Gleichungen welche möglicherweise sehr schwierig lösbar sind. Da die Lösungsverfahren, welche für die Lösung solcher Gleichungen verwendet wird, wie die Newtonsche Methode und steepest descent, iterative Methoden sind, sind diese rechnerisch sehr kostspielig und langsam. Aus diesem Grund wird in der Praxis der Ansatz des kleinsten quadratischen Fehlers vermieden.

Um diese nichtlinearen Gleichungen zu vermeiden und dennoch eine Lösung zu finden, welche uns eine passende Anzahl von $x[n]$ Werten mit der Einheitssprungantwort verbindet, kann ein eleganter Trick, der zur Padé-Approximation führt verwendet werden. Aus (18.20) bekommt man zunächst

$$H(z)A_p(z) = B_q(z). \quad (18.23)$$

In der Zeitebene betrachtet handelt es sich bei dieser Gleichung um die Faltung

$$h(n) + \sum_{k=1}^p a_p(k)h(n-k) = b_q(n). \quad (18.24)$$

Die rechte Seite der Gleichung für $n > q$ ist dabei gleich null. Die Fallunterscheidung

$$x(n) + \sum_{k=1}^p a_p(k)x(n-k) = \begin{cases} b_q(n) & \text{mit } n = 0, 1, \dots, q \\ 0 & \text{mit } n = q+1, \dots, q+p \end{cases} \quad (18.25)$$

kann in die Matrixform

$$\left[\begin{array}{cccc} x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ x(2) & x(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x(q) & x(q-1) & \cdots & x(q-p) \\ x(q+1) & x(q) & \cdots & x(q-p+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(q+p) & x(q+p-1) & \cdots & x(q) \end{array} \right] \left[\begin{array}{c} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{array} \right] = \left[\begin{array}{c} b_q(0) \\ b_q(1) \\ b_q(2) \\ \vdots \\ b_q(q) \\ 0 \\ \vdots \\ 0 \end{array} \right] \quad (18.26)$$

gebracht werden. Diese Matrix ist uns schon aus dem Abschnitt 18.2.2 bekannt. Es handelt sich nämlich um eine Padé-Approximation, welche wir nun für das Signal mit einer vorgegebener Ordnung berechnen können.

Beispiel. Die ersten sechs Werte

$$x = \begin{bmatrix} 1.0000 \\ 1.5000 \\ 0.7500 \\ 0.3750 \\ 0.1875 \\ 0.0938 \end{bmatrix}$$

eines Signales sind vorgegeben. Das Ziel ist, drei verschiedene Padé-Approximanten mit drei verschiedenen Freiheitsgraden zu finden. Die Grade von Zähler und Nenner dieser Approximationen sind in der folgenden Tabelle gegeben.

Modell	Zählergrad q	Nennergrad p
all pole	0	2
finite impulse response (FIR)	2	0
infinite impulse responses (IIR)	1	1

1. Das *All pole* Modell ($q = 0, p = 2$) wird in die Matrixform

$$\begin{bmatrix} x(0) & 0 & 0 \\ x(1) & x(0) & 0 \\ x(2) & x(1) & x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ 0 \\ 0 \end{bmatrix} \quad (18.27)$$

gebracht. Von der ersten Gleichung folgt, dass der Wert $b_0 = x(0) = 1$ ergeben muss. Weiter können nun die letzten zwei Gleichungen

$$\begin{bmatrix} x(0) & 0 \\ x(1) & x(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} x(1) \\ x(2) \end{bmatrix} \quad (18.28)$$

eingesetzt und gelöst werden.

$$\begin{bmatrix} 1 & 0 \\ 1.5 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} 1.50 \\ 0.75 \end{bmatrix} \quad (18.29)$$

Die resultierenden Koeffizienten

$$a_1 = -1.50 \quad a_2 = 1.50 \quad (18.30)$$

können nun zusammen mit dem b_0 -Koeffizienten in die Übertragungsfunktion eingesetzt werden. Somit erhalten wir als Ergebnis für die Approximation unseres $x(n)$

$$H(z) = \frac{1}{1 - 1.50z^{-1} + 1.50z^{-2}}. \quad (18.31)$$

Man sollte beachten, dass sich die komplexen Pole nicht im Inneren des Einheitskreises befinden und somit das Resultat kein stabiles Modell sein wird. Evaluiert man die Impulsantwort des Systems, erhalten wir die Approximation

$$\hat{x} = [1, 1.500, 0.750, -1.125, -2.8125, -2.5312]. \quad (18.32)$$

für $x(n)$. Daraus geht hervor das $x(n) = \hat{x}$ für $n = 0, 1, 2$ gleich ist, jedoch bei höheren n keine genaue Approximation liefert.

2. Für das FIR-Modell ($q = 2, P = 0$) ist das Resultat einfach mit $A(z) = 1$ berechenbar. Da alle weiteren a -Koeffizienten wegfallen werden die b -Koeffizienten mit dem Eingangssignal gleich gesetzt

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}. \quad (18.33)$$

Somit erhalten wir

$$H(z) = 1 + 1.50z^{-1} + 0.75z^{-2} \quad (18.34)$$

als unser Modell.

3. Für das IIR-Modell mit ($q = 1, p = 1$) welches die Form

$$H(z) = \frac{b(0) + b(1)z^{-1}}{1 + a(1)z^{-1}} \quad (18.35)$$

erhalten wird, sind die Padé-Gleichungen

$$\begin{bmatrix} x(0) & 0 \\ x(1) & x(0) \\ x(2) & x(1) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ 0 \end{bmatrix}. \quad (18.36)$$

Der Pol in der letzten Gleichung

$$x(2) + a_1 x(1) = 0 \quad (18.37)$$

kann damit berechnet werden. Eingesetzt und nach a_1 aufgelöst erhalten wir

$$a_1 = -\frac{x(2)}{x(1)} = -\frac{0.75}{1.5} = -0.5. \quad (18.38)$$

Mit den bekannten Polen können nun die Nullstellen mit den oberen zwei Gleichungen

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1.5 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (18.39)$$

berechnet werden. Die Koeffizienten können nun in das Modell eingesetzt werden, womit wir

$$H(z) = \frac{1 + z^{-1}}{1 - 0.5z^{-1}} \quad (18.40)$$

erhalten. Dieses Modell hat eine identische Impulsantwort

$$h(n) = (0.5)^n u(n) + (0.5)^{n-1} u(n-1) = \delta(n) + 1.5(0.5)^{n-1} u(n-1) \quad (18.41)$$

wie $x(n)$. Erkennbar wenn man $n = 0$ bis $n = 5$ in $h(n)$ einsetzt.

$$\begin{aligned} h(0) &= \delta(0) + 1.5(0.5)^{0-1} u(0-1) = 1 \\ h(1) &= \delta(1) + 1.5(0.5)^{1-1} u(1-1) = 1.5 \\ h(2) &= \delta(2) + 1.5(0.5)^{2-1} u(2-1) = 0.75 \\ h(3) &= \delta(3) + 1.5(0.5)^{3-1} u(3-1) = 0.375 \\ h(4) &= \delta(4) + 1.5(0.5)^{4-1} u(4-1) = 0.1875 \\ h(5) &= \delta(5) + 1.5(0.5)^{5-1} u(5-1) = 0.09375 \end{aligned}$$

Im Vergleich noch einmal \mathbf{x} welches Padé-Approximiert wurde

$$\mathbf{x} = [1.0000, 1.5000, 0.7500, 0.3750, 0.1875, 0.0938]^T.$$

Dies ist jedoch ein Spezialfall und kommt nicht immer vor. ○

18.4 Wann nun eine Padé-Approximation?

Die Padé-Approximation hat sich ergänzend zur Taylor-Reihe als ein sehr nützliches Werkzeug gezeigt. Es ist jedoch nicht immer die beste Lösung. Bei vielen Approximationen reicht eine Taylor-Reihe aus und die Padé-Approximation ist dann nur ein Mehraufwand. Hört man jedoch, dass eine Funktion als ein Bruch von rationalen Funktionen dargestellt werden soll oder eine Funktion, welche bei grossen Werten konvergiert, approximiert werden soll, dann sollte man als erstes an die Padé-Approximation denken.

Literatur

- [1] George A. Baker und Peter Graves-Morris. *Padé Approximants* - 2. Aufl. Cambridge: Cambridge University Press, 2009. ISBN: 978-0-511-53007-4.
- [2] Cleve Moler und Charles Van Loan. “Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later”. In: *SIAM Review* 45.1 (2003), 9–10. doi: 10.1137/s00361445024180. URL: <http://www.cs.cornell.edu/cv/researchpdf/19ways.pdf#page=9>.

19

Kettenbrüche und Padé-Approximation

Andreas Müller

Kettenbrüche versprechen eine bestmögliche Approximation einer Zahl durch rationale Zahlen, die Padé-Approximation verspricht eine bestmögliche Approximation einer Potenzreihe durch eine gebrochen rationale Funktion. Besteht ein Zusammenhang?

In Kapitel 17 wurde ohne Beweis eine Approximation der Taylor-Reihe der Arkustangens-Funktion

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11} + \dots \quad (19.1)$$

als Kettenbruch präsentiert. Doch genau genommen ist das gar kein Kettenbruch in dem Sinne, wie er im Rest von Kapitel 17 betrachtet wurde, denn bestimmt man die Approximationsbrüche, entstehen nicht Brüche von ganzen Zahlen sondern eine gebrochen rationale Funktion. In diesem Kapitel soll gezeigt werden, dass der Kettenbruch eigentlich nur eine Schreibweise ist, mit der verschiedene Padé-Approximanten von $\arctan x$ effizient berechnet werden können.

19.1 Kettenbrüche und Matrizen

Wir verwenden wieder die schon in Kapitel 17 eingeführte Notation

$$\frac{p_n}{q_n} = a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{\dots + \cfrac{\dots}{\dots + \cfrac{b_{n-1}}{a_{n-1} + \cfrac{b_n}{a_n}}}}}}$$

Für einen Kettenbruch hatten verlangt, dass alle Zahlen a_k und b_k wie auch die Zähler p_n und Nenner q_n der Näherungsbrüche ganze Zahlen sind. Für die Approximation der Funktion $\arctan x$ müssen wir diese Voraussetzung etwas aufweichen. Wir verlangen nur noch, dass alle diese Ausdrücke Polynome in x sind.

Wir brauchen eine Methode zur Berechnung der Approximationssähler und -nennen. Dazu berechnen wir, wie sich ein Bruch ändert, wenn ein zusätzlicher Bruchterm $\frac{p}{q}$ im Nenner hinzugefügt wird:

$$\frac{\tilde{p}}{\tilde{q}} = \frac{b}{a + \frac{p}{q}} = \frac{bq}{ab + p}. \quad (19.2)$$

Schreibt man die Brüche als Spaltenvektoren mit Zähler und Nenner als Komponenten, dann kann man die Operation (19.2) in Matrixform als

$$\begin{pmatrix} \tilde{p} \\ \tilde{q} \end{pmatrix} = \begin{pmatrix} b & q \\ a & ab + p \end{pmatrix} = \begin{pmatrix} 0 & b \\ 1 & a \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}$$

schreiben. Durch Iteration dieser Idee kann man daher den Näherungsbruch p_n/q_n ebenfalls in Matrixform als

$$\begin{pmatrix} p_n \\ q_n \end{pmatrix} = \begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} \begin{pmatrix} 0 & b_2 \\ 1 & a_2 \end{pmatrix} \begin{pmatrix} 0 & b_3 \\ 1 & a_3 \end{pmatrix} \cdots \begin{pmatrix} 0 & b_{n-1} \\ 1 & a_{n-1} \end{pmatrix} \begin{pmatrix} b_n \\ a_n \end{pmatrix}$$

schreiben. Will man den letzten Bruch b_n/a_n in der Kettenbruchentwicklung weglassen, also den Kettenbruch verkürzen, dann ist das gleichbedeutend damit, $b_n = 0$ und $a_n = 1$ zu setzen. Damit erhält man aber die Approximation p_{n-1}/q_{n-1} . Kombiniert man beide Approximationen in eine Matrix, erhält man

$$\begin{pmatrix} p_{n-1} & p_n \\ q_{n-1} & q_n \end{pmatrix} = \begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} \begin{pmatrix} 0 & b_2 \\ 1 & a_2 \end{pmatrix} \begin{pmatrix} 0 & b_3 \\ 1 & a_3 \end{pmatrix} \cdots \begin{pmatrix} 0 & b_{n-1} \\ 1 & a_{n-1} \end{pmatrix} \begin{pmatrix} 0 & b_n \\ 1 & a_n \end{pmatrix}. \quad (19.3)$$

Das Produkt (19.3) wurde oben zunächst von rechts nach links abgeleitet, es wurde der "letzte" Ausdruck in Matrixform gebracht und dann nach links fortschreitend der ganze Kettenbruch aufgebaut. Die Matrixmultiplikation ist jedoch assoziativ, so dass man ausgerüstet mit Formel (19.3) die Quotienten p_n/q_n jetzt auch von links nach rechts berechnen kann.

19.2 Padé-Approximation von $\arctan x$

Wir versuchen jetzt zu verstehen, wie man die Taylor-Reihe 19.1 als Kettenbruch deuten kann. Nach dem Prinzip der Padé-Approximation suchen wir einen Bruch p_n/q_n , der mit der Taylor-Reihe bis zu einem gewissen Grad k übereinstimmt. Wir verlangen daher, dass p_n und q_n Polynome sind, für die gilt

$$\frac{p_n(x)}{q_n(x)} = \arctan x + O(x^{k+1}).$$

Wir werden später sehen, dass wir $k = 2n + 1$ verlangen können, was zum Beispiel bedeutet, dass $p_2(x)/q_2(x)$ mit der Reihe 19.1 bis und mit Termen fünfter Ordnung übereinstimmt. Wie bei der Herleitung der Padé-Approximation in Kapitel 18 multiplizieren wir mit dem Nenner und erhalten die Bedingung

$$p_n(x) - q_n(x) \arctan x = O(x^{k+1}). \quad (19.4)$$

Auch diese Bedingung lässt sich in Vektorform fassen:

$$(1 - \arctan x) \begin{pmatrix} p_n(x) \\ q_n(x) \end{pmatrix} = O(x^{k+1})$$

oder $(1 - \arctan x) \begin{pmatrix} p_{n-1}(x) & p_n(x) \\ q_{n-1}(x) & q_n(x) \end{pmatrix} = (O(x^k) \quad O(x^{k+1})).$ (19.5)

Die Rekursionsfolge startet mit der Matrix

$$(1 - \arctan x) \begin{pmatrix} p_0(x) & p_1(x) \\ q_0(x) & q_1(x) \end{pmatrix} = (1 - \arctan x) \begin{pmatrix} 0 & p_1(x) \\ 1 & q_1(x) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & p_1(x) - q_1(x) \left(x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots \right) \end{pmatrix}.$$

Damit auf der rechten Seite in der zweiten Komponente die Terme erster Ordnung wegfallen, müssen p_1 und q_1 als Polynome möglichst niedrigen Grades so gewählt werden, dass

$$p_1(x) - xq_1(x) = O(x^2),$$

dies kann zum Beispiel $q_1(x) = 1$ und $p_1(x) = x$ geschehen. Damit ist

$$\begin{pmatrix} p_0(x) & p_1(x) \\ q_0(x) & q_1(x) \end{pmatrix} = \begin{pmatrix} 0 & x \\ 1 & 1 \end{pmatrix}$$

der Start der Rekursion.

19.2.1 Approximation bis zur Ordnung 3

Die Bedingung (19.5) kann jetzt dazu verwendet werden, auch die weiteren a_k und b_k zu bestimmen. Für a_1 und b_1 rechnen wir dazu zunächst das Produkt

$$\begin{pmatrix} p_0(x) & p_1(x) \\ q_0(x) & q_1(x) \end{pmatrix} \begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} = \begin{pmatrix} 0 & x \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & b_1 \\ 1 & a_1 \end{pmatrix} = \begin{pmatrix} x & a_1 x \\ 1 & a_1 + b_1 \end{pmatrix}$$

aus. Multiplikation von links mit dem Zeilenvektor ergibt

$$(1 - \arctan x) \begin{pmatrix} x & a_1 x \\ 1 & a_1 + b_1 \end{pmatrix} = \left(O(x^3) \quad a_1 x - (a_1 + b_1) \left(x - \frac{x^3}{3} + O(x^5) \right) \right).$$

Die Polynome a_1 und b_1 müssen so gewählt werden, dass keine Terme dritter Ordnung mehr stehen bleiben. Dies ist natürlich nicht eindeutig möglich sondern nur bis auf ein Vielfaches. Den Term erster Ordnung kann man mit konstantem a_1 zum verschwinden bringen, dann muss aber b_1 mindestens den Grad 2 haben. Es muss also

$$a_1 x - (a_1 + c x^2) \left(x - \frac{x^3}{3} + O(x^5) \right) = a_1 \frac{x^3}{3} - c x^3 + O(x^5)$$

sein. Dies lässt sich zum Beispiel mit $a_1 = 3$ und $c = 1$ erreichen. Der Näherungsbruch ist dann

$$\begin{pmatrix} p_1(x) & p_2(x) \\ q_1(x) & q_2(x) \end{pmatrix} = \begin{pmatrix} x & 3x \\ 1 & x^2 + 3 \end{pmatrix} \quad \Rightarrow \quad \frac{p_2(x)}{q_2(x)} = \frac{3x}{3 + x^2}.$$

19.2.2 Approximation bis zur Ordnung 5

Wir führen auch noch den Schritt $k = 2$ durch, bei dem die Terme fünfter Ordnung wegfallen sollten. Das Matrizenprodukt ist

$$\begin{pmatrix} x & 3x \\ 1 & x^2 + 3 \end{pmatrix} \begin{pmatrix} 0 & b_2 \\ 1 & a_2 \end{pmatrix} = \begin{pmatrix} 3x & b_2x + 3xa_2 \\ x^2 + 3 & b_2 + a_2(x^2 + 3) \end{pmatrix}.$$

Nach Linksmultiplikation mit dem Zeilenvektor ergibt sich wieder

$$\begin{aligned} b_2 p_1(x) + a_2 p_2(x) - (b_2 q_1(x) + a_2 q_2(x)) \arctan x \\ = b_2(p_1(x) - q_1(x) \arctan x) + a_2(p_2(x) - q_2(x) \arctan x) = O(x^7). \end{aligned}$$

Im ersten Term wissen wir, dass wir, dass bereits die Terme der Ordnung ≤ 1 wegfallen. In der zweiten Klammer fallen die Terme bis zur dritten Ordnung weg. Es folgt daher wieder, dass man für b_2 einen Term zweiten Grades der Form cx^2 wählen muss und folglich für a_2 eine Konstante. Setzt man alles ein, erhält man

$$\begin{aligned} cx^2 \left(x - x + \frac{x^3}{3} - \frac{x^5}{5} + O(x^7) \right) + a_2 \left(3x - (x^2 + 3) \left(x - \frac{x^3}{3} + \frac{x^5}{5} + O(x^7) \right) \right) \\ = c \frac{x^5}{3} + O(x^7) + a_2 \left(\frac{x^5}{3} - \frac{3}{5}x^5 + O(x^7) \right) \\ = x^5 \left(\frac{c}{3} - a_2 \frac{4}{15} \right) + O(x^7), \end{aligned}$$

woraus als mögliche Lösung

$$b_2 = 4x^2, \quad a_2 = 5$$

folgt. Der zugehörige Näherungsbruch ist

$$\arctan x \approx \frac{p_3(x)}{q_3(x)} = \frac{xb_2 + 3xa_2}{b_2 + a_2(x^2 + 3)} = \frac{4x^3 + 15x}{9x^2 + 15}.$$

Es lässt sich zum Beispiel mit einem Computer-Algebra-Programm wie Maxima verifizieren, dass die Matrizen

$$\begin{pmatrix} 0 & b_n \\ 1 & a_n \end{pmatrix} = \begin{pmatrix} 0 & n^2 x^2 \\ 1 & 2n + 1 \end{pmatrix} \quad \text{also} \quad b_n = n^2 x^2 \quad \text{und} \quad a_n = 2n + 1 \quad (19.6)$$

tatsächlich dafür sorgen, dass die Kettenbruchentwicklung immer eine Approximation bis zur Ordnung $2n + 1$ ist. Damit erhalten wir den Kettenbruch

$$\begin{aligned} \arctan x = & \frac{x}{1 + \frac{x^2}{3 + \frac{4x^2}{5 + \frac{9x^2}{7 + \frac{16x^2}{9 + \dots}}}}}, \end{aligned}$$

den wir bereits in Kapitel 17 gefunden haben.

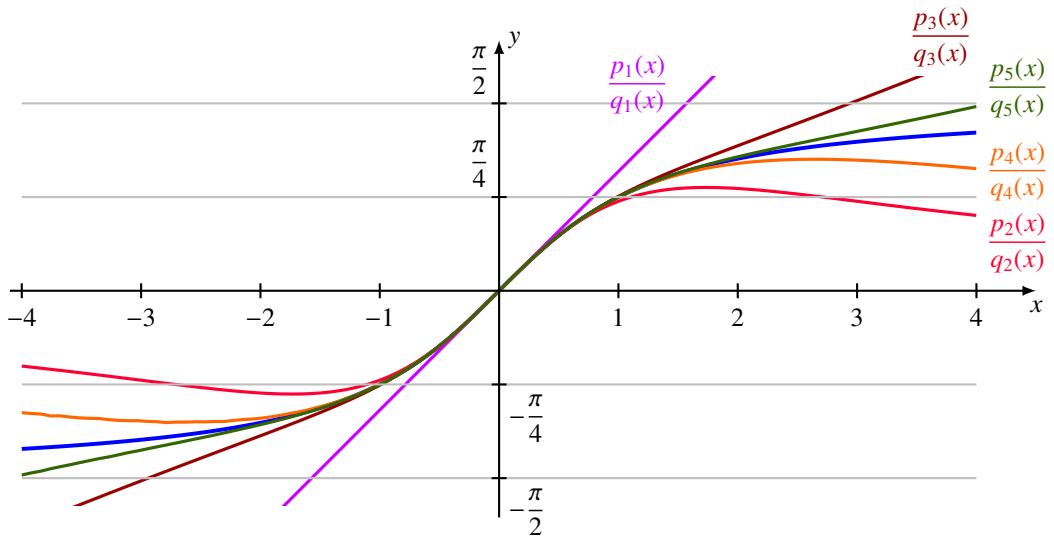


Abbildung 19.1: Graphen der Näherungsbrüche p_n/q_n für die Funktion $\arctan x$ (blau) für $n = 1, \dots, 5$. Die Näherungsbrüche approximieren $\arctan x$ abwechselnd von oben und unten.

Mit Computerhilfe kann man jetzt die folgenden Näherungsbrüche ausrechnen:

$$\begin{aligned}\frac{p_1(x)}{q_1(x)} &= \frac{x}{1}, \\ \frac{p_2(x)}{q_2(x)} &= \frac{3x}{x^2 + 3}, \\ \frac{p_3(x)}{q_3(x)} &= \frac{4x^3 + 15x}{9x^2 + 15}, \\ \frac{p_4(x)}{q_4(x)} &= \frac{55x^3 + 105x}{9x^4 + 90x^2 + 105}, \\ \frac{p_5(x)}{q_5(x)} &= \frac{6x^5 + 735x^3 + 945x}{225x^4 + 1050x^2 + 945}.\end{aligned}$$

Ihre Graphen sind in Abbildung 19.1 dargestellt.

19.2.3 Der Grad von Zähler und Nenner

Wir ermitteln noch die Grade der Polynome p_n und q_n . Für den Grad $\deg p(x)$ eines Polynoms gelten die Rechenregeln

$$\begin{aligned}\deg 0 &= -\infty, \\ \det x^k &= k, \\ \deg p(x)q(x) &= \deg p(x) + \deg q(x), \\ \deg(p(x) + q(x)) &\leq \max(\deg p(x), \deg q(x)).\end{aligned}$$

n	1	2	3	4	5	6	7
$\deg p_n(x)$	1	3	3	5	5	7	7
$\deg q_n(x)$	2	2	4	4	6	6	8
Ordnung	3	5	7	9	11	13	15

Tabelle 19.1: Grade von Zähler und Nenner-Polynomen des Kettenbruchs für $\arctan x$. Die letzte Zeile enthält die Ordnung, bis zu der $p_n(x) - q_n(x) \arctan x$ verschwindet.

Damit lässt sich die Veränderung der Grade der Matrixeinträge bei Matrizenprodukten nachvollziehen, wir nenne dies die *Grad-Multiplikation*. Schreiben wir eckige Klammern für eine Matrix aus Graden, denn gelten dafür die Rechenregeln

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} \max(a_{11} + b_{11}, a_{12} + b_{21}) & \max(a_{11} + b_{12}, a_{12} + b_{22}) \\ \max(a_{21} + b_{11}, a_{22} + b_{21}) & \max(a_{21} + b_{12}, a_{22} + b_{22}) \end{bmatrix},$$

die man aus der Definition des Matrizenproduktes erhält, indem man Produkte durch Summen und Summen durch $\max(\cdot, \cdot)$ ersetzt.

Für den Spezialfall der Matrix (19.6) gilt

$$\deg \begin{pmatrix} 0 & n^2 x^2 \\ 1 & 2n+1 \end{pmatrix} = \begin{bmatrix} -\infty & 2 \\ 0 & 0 \end{bmatrix}.$$

Wir berechnen jetzt die Gradmatrizen für p_n und q_n . Zu Beginn gilt

$$\deg \begin{pmatrix} p_1(x) & p_2(x) \\ q_1(x) & q_2(x) \end{pmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}.$$

Für die späteren Näherungsbrüche berechnen wir gleich allgemeiner

$$\begin{bmatrix} k & k \\ k-1 & k+1 \end{bmatrix} \begin{bmatrix} -\infty & 2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} k & k+2 \\ k+1 & k+1 \end{bmatrix}$$

$$\begin{bmatrix} k & k+2 \\ k+1 & k+1 \end{bmatrix} \begin{bmatrix} -\infty & 2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} k+2 & k+2 \\ k+1 & k+3 \end{bmatrix}.$$

Daraus lesen wir ab, dass sich die Grade von Zähler und Nenner wie in Tabelle 19.1 entwickeln. In Kapitel 18 wurde gezeigt, dass diese Ordnungen auch die Padé-Approximanten charakterisieren. Der Näherungsbruch p_n/q_n ist also ein Padé-Approximant von $\arctan x$, und zwar ist

$$\frac{p_n(x)}{q_n(x)} = \begin{cases} [n+1/n] & n \text{ gerade} \\ [n/n+1] & n \text{ ungerade.} \end{cases}$$

20

Die Gleichung von Burgers

Michael Schmid

20.1 Einleitung

Die Gleichung von Burgers,

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad (20.1)$$

ist eine nichtlineare partielle Differentialgleichung. Das Definitionsgebiet

$$\Omega = \{(x, t) \in \mathbb{R} \times \mathbb{R}^t\} \quad (20.2)$$

besteht aus der Dimension x und der Zeit t . Die Lösung der Gleichung von Burgers $u(t, x)$ ist in Ω definiert, wobei die Anfangswerte $u(0, x)$ bekannt sein müssen. In diesem Paper wird die reibungsfreie Burgersgleichung

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0 \quad (20.3)$$

betrachtet, wobei der Diffusionsterm ν von (20.1) auf 0 gesetzt ist.

Das Aussergewöhnliche an der Gleichung ist die Nichtlinearität. Diese Eigenschaft erschwert die numerische Lösung erheblich.

Als Beispiel kann für die Randbedingung bei $u(0, x)$ eine Normalverteilung verwendet werden. Die gelöste Gleichung ist in Abbildung 20.1 abgebildet.

20.1.1 Beziehung zu den Navier-Stokes-Gleichungen

Die Gleichung wird häufig für die Modellierung von Fluiden verwendet. Sie wird gebraucht um die Navier-Stokes-Gleichungen zu vereinfachen. Die Navier-Stokes-Gleichungen beschreiben die Strömung von Fluiden. Die Gleichung von Burgers kann aus der Navier-Stokes-Gleichung

$$\rho \left(\frac{\partial u}{\partial t} + u \nabla u \right) = -\nabla p + \mu \nabla^2 u + F \quad (20.4)$$

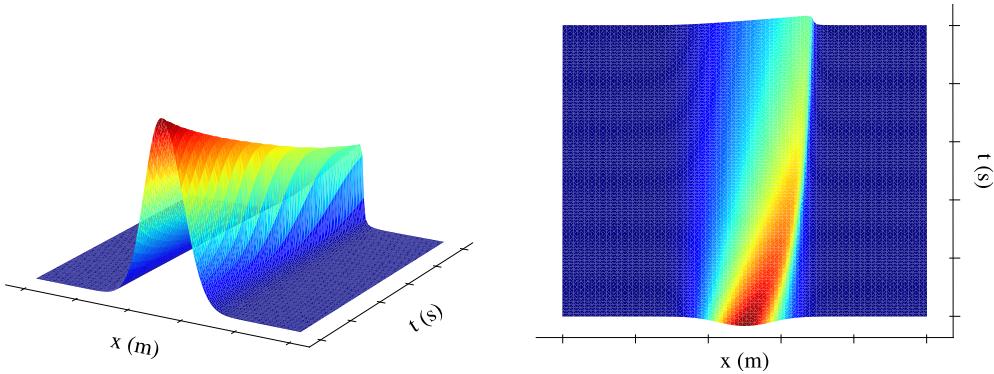


Abbildung 20.1: Gelöste Burgersgleichung

für newtonische inkompressible Fluide hergeleitet werden [2]. Wenn der Druck p und die externe Kraft F vernachlässigt wird, ergibt sich

$$\rho \left(\frac{\partial u}{\partial t} + u \nabla u \right) = \mu \nabla^2 u \quad (20.5)$$

Da die Dichte ρ für inkompressible Fluide konstant ist, kann sie mit der Viskosität μ zur Konstante der kinematischen Viskosität $\nu = \frac{\mu}{\rho}$

$$\frac{\partial u}{\partial t} + u \nabla u = \nu \nabla^2 u \quad (20.6)$$

umgeschrieben werden. Wie bereits erwähnt, wird die reibungsfreie Gleichung betrachtet ($\nu = 0$). Weiter werden die kommenden Berechnungen im eindimensionalen Raum durchgeführt. Somit kann der Ableitungsoperator ∇ als die Ableitung nach x umgeschrieben werden. Mit diesen Vereinfachungen der Navier-Stokes-Gleichung für newtonische inkompressible Fluide ist man bei der ursprünglichen Gleichung (20.3) angelangt.

20.2 Problemstellung

Das Ziel dieses Papers ist die numerische Berechnung der Lösung für die Gleichung von Burgers mit ihren Schwierigkeiten aufzuzeigen.

20.2.1 Numerische Wettervorhersage

Der numerische Lösungsansatz von nichtlinearen partiellen Differentialgleichungen entpuppte sich bereits in den 30er Jahren als eine grosse Herausforderung. Lewis Fry Richardson (1881 – 1953) war ein britischer Mathematiker und Physiker, welcher als einer der Ersten versuchte, das Wetter mittels numerischer Lösung der fundamentalen Differentialgleichungen zu prognostizieren. Für eine Vorhersage von sechs Stunden benötigte er mit seinem Team gegen sechs Wochen. Das Resultat war ernüchternd da schlichtweg falsch. Es stellte sich heraus, dass Richardson einen Algorithmus verwendete welcher numerisch instabil war. Im Verlauf des Papers wird aufgezeigt, welcher Fehler dabei begangen wurde.

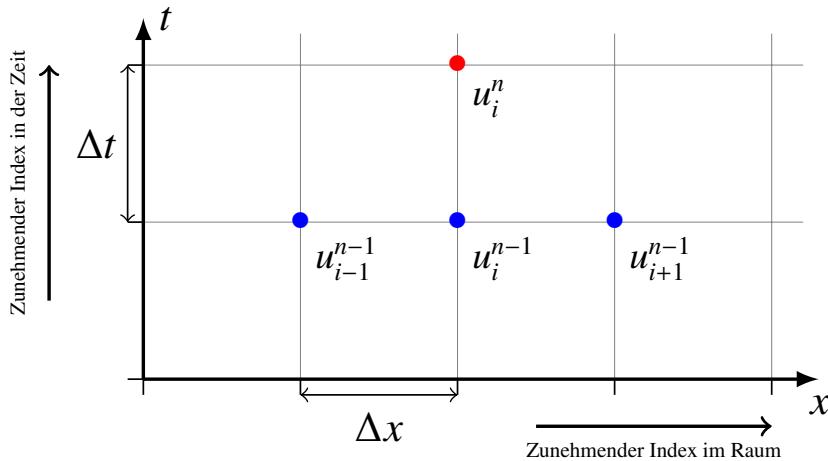


Abbildung 20.2: Notation Diskretisierung

20.2.2 Notation

Wie es bei der numerischen Lösung von partiellen Differentialgleichungen üblich ist, wird die Raum-Zeit-Ebene diskretisiert. Die gewählte Notation für die Diskretisierung kann Abbildung 20.2 entnommen werden, wobei u_i^n der zu berechnende Punkt ist. Weiter ist $i - 1$ Schritt zurück im Raum und $n - 1$ ein Schritt zurück in der Zeit. Δx und Δt sind die Schrittgrößen in der entsprechenden Dimension.

20.3 Lösungsmethoden

Das Kapitel 7 beschreibt den theoretischen Aspekt der Numerik partieller Differentialgleichungen. In diesem Abschnitt wird auf die numerischen Lösungsmethoden der Gleichung von Burgers eingegangen.

20.3.1 Explizit

Bei der Lösung der Gleichung mit dem expliziten Ansatz werden für die Berechnung des gewünschten Punktes u_i^n , auf Gitterpunkte einen Zeitschritt in der Vergangenheit u^{n-1} zurückgegriffen. Nachfolgend werden zwei grundlegende Varianten mit jeweils einer kleinen Variation aufgezeigt.

Lineare Methoden

Die Ableitung in der Zeit wird mit der Differenz der beiden Punkte $u_i^n - u_i^{n-1}$ realisiert. Die Ableitung im Raum wird mit der Differenz $u_i^{n-1} - u_{i-1}^{n-1}$ abstrahiert. Sie wird grafisch in Abbildung 20.3 dargestellt.

Die grundlegende Gleichung (20.3) kann mit der gewählten Methode

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \frac{u_i^{n-1} - u_{i-1}^{n-1}}{\Delta x} = 0 \quad (20.7)$$

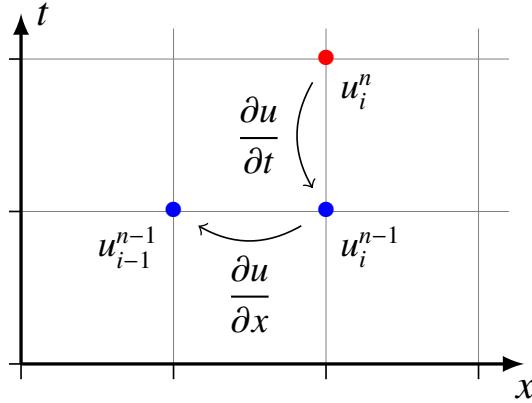
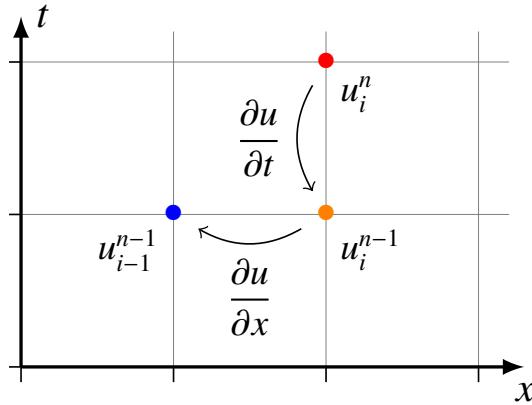


Abbildung 20.3: Lineare explizite Methode

Abbildung 20.4: Lineare explizite Methode mit u_i^{n-1} für die Multiplikation

umgeschrieben werden. Für die Lösung muss die Gleichung (20.7) nach

$$u_i^n = \frac{\Delta x u_i^{n-1}}{-\Delta t u_{i-1}^{n-1} + \Delta t u_i^{n-1} + \Delta x} \quad (20.8)$$

aufgelöst werden.

Für die Multiplikation des zweiten Bruches muss nicht unbedingt der gesuchte Punkt u_i^n verwendet werden (Siehe dazu die grafische Darstellung in Abbildung 20.4). Es kann sowohl der Punkt auf gleicher Stelle im Raum, als auch einen Zeitschritt in der Vergangenheit verwendet werden.

Die Gleichung

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1} \frac{u_i^{n-1} - u_{i-1}^{n-1}}{\Delta x} = 0 \quad (20.9)$$

ändert sich nur minimal. Auch kann wieder eine Lösung für den Wert im Punkt

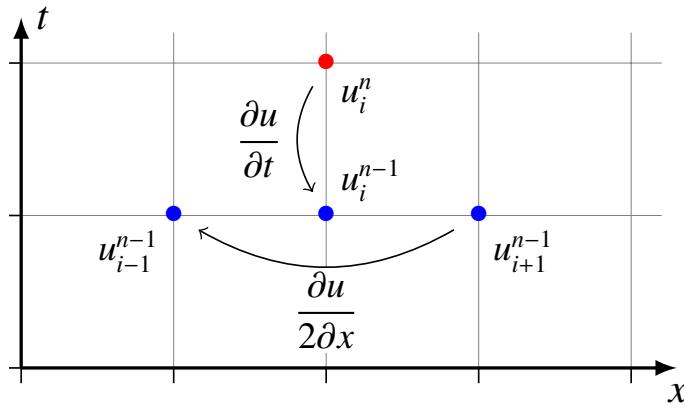


Abbildung 20.5: Explizite Leap-Frog Methode

$$u_i^n = \frac{u_i^{n-1} (\Delta t u_{i-1}^{n-1} - \Delta t u_i^{n-1} + \Delta x)}{\Delta x} \quad (20.10)$$

gefunden werden.

Stabilitätskriterium lineare Methoden

Die Stabilität kann für diese Lösungsmethode nicht garantiert werden. Sie folgt dem Courant-Friedrichs-Lowy-Kriterium

$$c = \frac{u \Delta t}{\Delta x} < c_{\text{krit.}} \quad (20.11)$$

Δt und Δx beschreiben die diskrete Schrittgrösse. Das Kriterium c_{krit} ist von der gewählten Lösungsmethode abhängig. Beim expliziten Verfahren beträgt $c_{\text{krit}} = 1$. Falls $c_{\text{krit}} > 1$ bedeutet dies, dass das Verhältnis zwischen Δt und Δx zu gross ist und die Berechnungen instabil werden. Für diesen Fall und mit der Annahme $u = 1$ müsste Δt grösser sein als Δx . Dies macht intuitiv Sinn, da der Raum nicht feiner abgetastet werden sollte als die Zeit.

Leap-Frog Methode

Bei genauer Betrachtung der Ableitungen in Abbildung 20.3 und Abbildung 20.4 erkennt man, dass sich die berechnete Ableitung eigentlich zwischen den beiden Punkten befindet. Grundsätzlich sollte man für ein optimales Resultat die Ableitung genau am Punkt u_i^n erhalten.

Die Leap-Frog Diskretisierung umgeht diese Problematik, indem sie für die Ableitung einen Punkt überspringt. Die daraus folgende Lösung für die Ableitung befindet sich auf der gleichen Ebene im Raum wie der Punkt u_i^n , allerdings noch einen Zeitschritt zurück. In Abbildung 20.5 ist die Leap-Frog Methode grafisch dargestellt.

Wiederum kann die Gleichung

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \frac{u_{i+1}^{n-1} - u_{i-1}^{n-1}}{2 \Delta x} = 0 \quad (20.12)$$

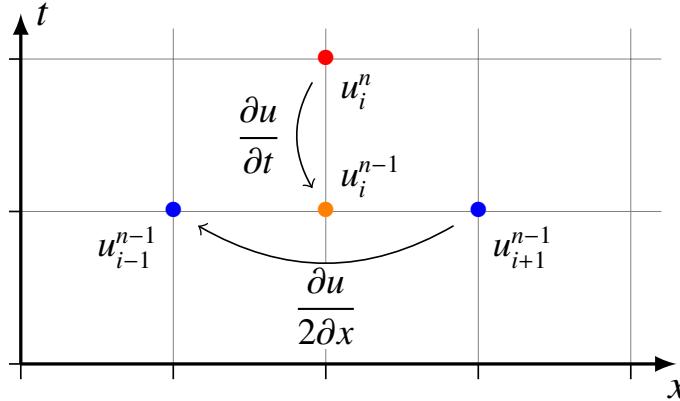


Abbildung 20.6: Explizite Leap-Frog Methode mit u_i^{n-1} für die Multiplikation

diskretisiert und nach dem gesuchten Punkt

$$u_i^n = \frac{2\Delta x u_i^{n-1}}{\Delta t u_{i+1}^{n-1} - \Delta t u_{i-1}^{n-1} + 2\Delta x} \quad (20.13)$$

aufgelöst werden.

Wie bereits in den gezeigten Varianten kann für die Multiplikation der Punkt u_i^{n-1} verwendet werden (Abbildung 20.6). Die Diskretisierung ändert sich zu

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1} \frac{u_{i+1}^{n-1} - u_{i-1}^{n-1}}{2\Delta x} = 0 \quad (20.14)$$

und die Lösung kann berechnet werden als

$$u_i^n = \frac{u_i^{n-1} (-\Delta t u_{i+1}^{n-1} + \Delta t u_{i-1}^{n-1} + 2\Delta x)}{2\Delta x}. \quad (20.15)$$

Stabilitätskriterium Leap-Frog Verfahren

Mit der Einführung des Leap-Frog Verfahrens wurde eine unerwünschte Instabilität in das System gebracht. Die Problematik ist, dass im Raum über zwei Raumschritte abgetastet wird. Dabei kann schon in einem Abtastschritt eine Änderung vorkommen. Von einer technischen Blickrichtung kann von einer Verletzung des Nyquist-Shannon-Abtasttheorems gesprochen werden. Die Abtastrate beträgt nicht das Doppelte der höchsten Frequenz im System.

Die erwähnte Instabilität im Abschnitt 20.2.1, auf die Richardson gestossen ist, ist die hier beschriebe. Ein analytisches Beispiel aus der Biografie von Richardson [3] mit der Differentialgleichung der Reibung:

$$\frac{dQ}{dt} = -\kappa Q, \quad \text{wobei} \quad Q = Q^0 \quad \text{bei} \quad t = 0. \quad (20.16)$$

Mit dem expliziten Verfahren würde die Diskretisierung wie folgt aussehen:

$$\frac{Q^{n+1} - Q^n}{\Delta t} = -\kappa Q^n. \quad (20.17)$$

Die Lösung

$$Q^n = Q^0(1 - \kappa\Delta t)^n \quad (20.18)$$

ist eine monoton fallende geometrische Reihe, falls $|\kappa\Delta t| < 1$.

Wird nun das Leap-Frog Verfahren angewendet ändert sich die Diskretisierung zu

$$\frac{Q^{n+1} - Q^{n-1}}{2\Delta t} = -\kappa Q^n. \quad (20.19)$$

Die Gleichung kann weiter nach

$$-\frac{Q^{n+1} - Q^{n-1}}{\kappa 2\Delta t} = Q^n \quad (20.20)$$

aufgelöst werden. Die Lösung dieser linearen Differentialgleichung kann mit einem Potenzansatz der Form

$$Q^n = Q^0 \xi^n. \quad (20.21)$$

gefunden werden. Dies ist wiederum eine monoton fallende Lösung gegen Null, wenn $0 < \xi < 1$. Setzen wir für $Q^n = \xi$, $Q^{n+1} = \xi^2$ und $Q^{n-1} = \xi^0$, erhält man

$$\frac{\xi^2 - 1}{2\Delta t} = -\kappa\xi. \quad (20.22)$$

Diese quadratische Gleichung kann mit den folgenden Schritten aufgelöst werden:

$$\xi^2 + (\kappa 2\Delta t)\xi - 1 = 0 \quad (20.23a)$$

$$\xi = \frac{-\kappa 2\Delta t \pm \sqrt{(\kappa 2\Delta t)^2 + 4}}{2} \quad (20.23b)$$

$$\xi_+ = -\kappa\Delta t + \sqrt{\kappa^2\Delta t^2 + 1} \quad (20.23c)$$

$$\xi_- = -\kappa\Delta t - \sqrt{\kappa^2\Delta t^2 + 1}. \quad (20.23d)$$

Man sieht, dass $|\xi_+| < 1$ und somit eine abfallende, gegen Null gehende Lösung entsteht. Allerdings ist $|\xi_-| > 1$ und damit wächst die zu ξ_- gehörige Lösung ohne Limit mit n . Dieser Anteil wird der *computational mode* genannt und entsteht bei der Verwendung des expliziten Leap-Frog Verfahrens. In Abbildung 20.7 kann der Verlauf der Lösungen für ξ betrachtet werden.

In Abbildung 20.8 kann diese Instabilität mit der Gleichung von Burgers beobachtet werden. Mit fortschreitender Zeit würde die Lösung gegen ∞ gehen.

Unterdrückung des computational mode Dieser hochfrequente Fehler kann mittels eines digitalen Tiefpassfilters eliminiert werden. Das sogenannte *Robert-Asselin-Time-Filter* [1] kann hierfür eingesetzt werden. Das Filter

$$\hat{u}(t_n, x_i) = u(t_n, x_i) + \frac{\alpha}{2} [u(t_n, x_{i-1}) - 2u(t_n, x_i) + u(t_n, x_{i+1})] \quad (20.24)$$

wird in dieser Anwendung allerdings nicht auf die Zeit, sondern auf die Raum-Domäne angewendet. Mit dem Dämpfungsfaktor α kann das Filter auf die Applikation angepasst werden.

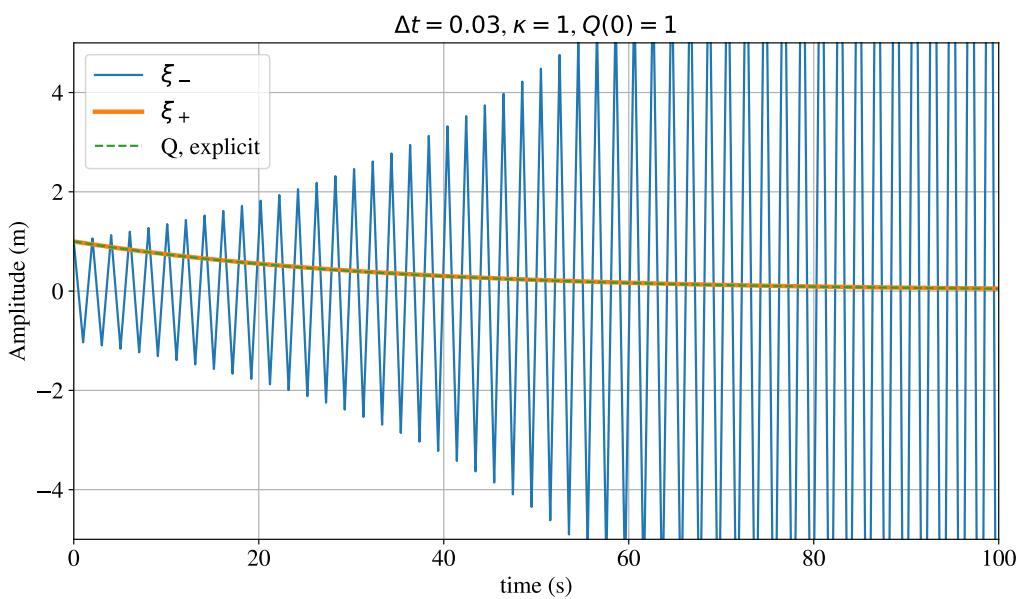


Abbildung 20.7: Computational mode für die Differentialgleichung der Reibung

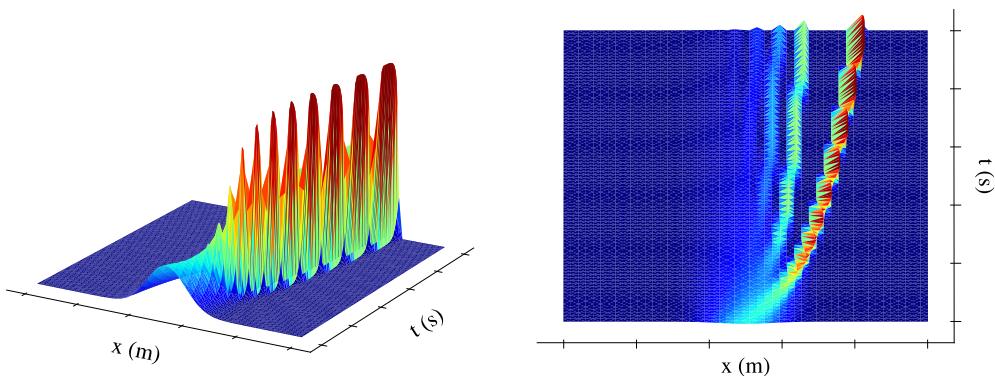


Abbildung 20.8: Computational mode für die Burgers Gleichung

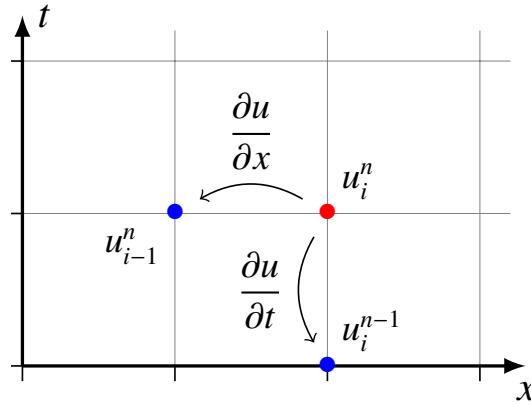


Abbildung 20.9: Implizite quadratische Methode

20.3.2 Implizit

Beim impliziten Verfahren werden Datenpunkte der gleichen Zeitebene wie der gesuchte Punkt u_i^n verwendet.

Quadratisch

Die direkteste Art der Diskretisierung, Abbildung 20.9, kann wie folgt beschrieben werden. Da der Punkt u_i^n für die Multiplikation verwendet wird, entsteht

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \frac{u_i^n - u_{i-1}^n}{\Delta x} = 0. \quad (20.25)$$

Die beiden Lösungen dieser quadratischen Gleichung sind

$$u_i^n = \frac{\Delta t u_{i-1}^n - \Delta x \pm \sqrt{\Delta t^2 (u_{i-1}^n)^2 + 4 \Delta t \Delta x u_i^{n-1} - 2 \Delta t \Delta x u_{i-1}^n + \Delta x^2}}{2 \Delta t}, \quad (20.26)$$

dabei führt der negative Teil der Lösung zu einer Instabilität.

Linear

Wird wie gewohnt der Wert u_i^{n-1} für die Multiplikation verwendet, entsteht eine simplere lineare Gleichung

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1} \frac{u_i^n - u_{i-1}^n}{\Delta x} = 0, \quad (20.27)$$

die in Abbildung 20.10 dargestellt ist. Die Lösung kann mit

$$u_i^n = \frac{u_i^{n-1} (\Delta t u_{i-1}^n + \Delta x)}{\Delta t u_i^{n-1} + \Delta x} \quad (20.28)$$

gefunden werden.

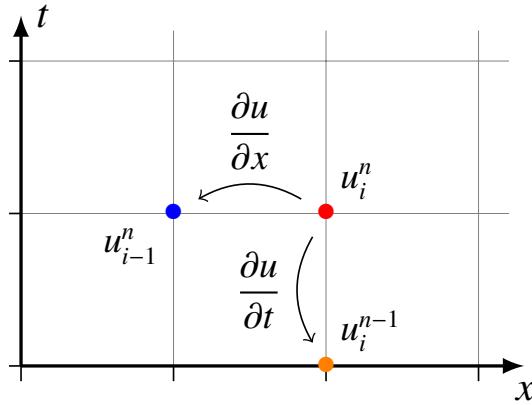


Abbildung 20.10: Implizite lineare Methode

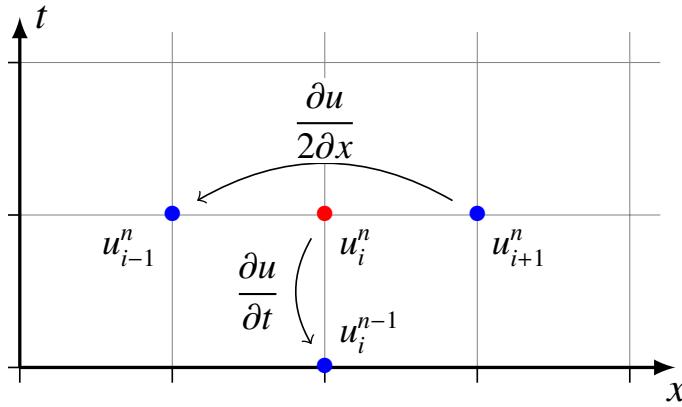


Abbildung 20.11: Implizite leap-frog Methode

Leap-Frog Verfahren

Mit dem impliziten Leap-Frog Verfahren (Abbildung 20.11) kann für die Ableitung im Raum der Wert an der korrekten Stelle berechnet werden. Nun wird allerdings der Punkt u_{i+1}^n für die Berechnung verwendet, welcher bei der Berechnung des Punktes u_i^n noch unbekannt ist. Somit entsteht ein lineares Gleichungssystem.

Um das lineare Gleichungssystem in gewohnter Notation zu schreiben, werden alle Werte auf der Zeitebene u^n als gesuchte Variablen angenommen. Bei Diskretisierung mit dem Leap-Frog Verfahren entsteht

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \cong \frac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1} \frac{u_{i+1}^n - u_{i-1}^n}{2 \Delta x} = 0. \quad (20.29)$$

Die Gleichung kann nach den Unbekannten

$$-u_i^{n-1} \Delta t u_{i-1}^n + 2 \Delta x u_i^n + u_i^{n-1} \Delta t u_{i+1}^n = 2 \Delta x u_i^{n-1} \quad (20.30)$$

sortiert werden.

Für die Randpunkte kann das Leap-Frog Verfahren nicht angewendet werden. Nachfolgend wurde für diese Punkte das Verfahren in Abschnitt 20.3.2 verwendet.

Das Gleichungssystem kann in Matrix-Notation umgeschrieben werden als

$$\begin{bmatrix} dx - u_1^{n-1} \Delta t & u_1^{n-1} \Delta t & 0 & \dots & 0 \\ -u_2^{n-1} \Delta t & 2 \Delta x & u_2^{n-1} \Delta t & \dots & 0 \\ 0 & -u_3^{n-1} \Delta t & 2 \Delta x & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & u_{M-1}^{n-1} \Delta t \\ 0 & 0 & \dots & -u_M^{n-1} \Delta t & \Delta x + u_M^{n-1} \Delta t \end{bmatrix} \begin{bmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_M^n \end{bmatrix} = \begin{bmatrix} \Delta x u_1^{n-1} \\ 2 \Delta x u_2^{n-1} \\ 2 \Delta x u_3^{n-1} \\ \vdots \\ \Delta x u_M^{n-1} \end{bmatrix}. \quad (20.31)$$

Matrizen mit dieser Anordnung werden tridiagonale Matrizen genannt. Die Gleichungen haben die allgemeine Form

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \quad (20.32)$$

oder in Matrixform

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & \ddots & \ddots & c_{n-1} & \\ 0 & & a_n & b_n & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix} \quad (20.33)$$

und können mit dem Thomas-Algorithmus [4] mit der Geschwindigkeit $O(n)$ gelöst werden. Mehr zum Algorithmus im Abschnitt 6.1.1.

Stabilitätskriterium implizite Verfahren

Eine Instabilität wie in den expliziten Verfahren konnte nicht beobachtet werden. All gezeigten Varianten können als stabil betrachtet werden. Mehr dazu im Abschnitt 7.2.5

20.4 Resultate

Im Folgenden werden die Berechnungen für eine normalverteilte Startbedingung mit der expliziten und impliziten Methode gezeigt. Um den Vergleich zwischen den verschiedenen Methoden besser ersichtlich zu machen, werden die Resultate in zwei Dimensionen aufgezeigt. Für jeden Zeitschritt wurde ein neuer Subplot gezeichnet.

20.4.1 Explizit

Beim expliziten Verfahren werden die beiden linearen Varianten aufgezeigt, wobei sich nur der gewählte Punkt für die Multiplikation unterscheidet. In Abbildung 20.12 zeigen sich die Unterschiede, welche durch die Änderung des gewählten Punktes für die Multiplikation hervorgerufen werden.

Algorithm 1 Tridiagonal matrix algorithm (Thomas algorithm)

```

1: function THOMAS(a, b, c, d) ▷ Vectors
2:    $\hat{c}_1 \leftarrow \frac{c_1}{b_1}$ 
3:    $\hat{d}_1 \leftarrow \frac{d_1}{b_1}$ 
4:   for  $i = 2, 3, \dots, n - 1$  do ▷ Forward sweep
5:      $\hat{c}_i \leftarrow \frac{c_i}{b_i - a_i \hat{c}_{i-1}}$ 
6:   end for
7:   for  $i = 2, 3, \dots, n$  do ▷ Forward sweep
8:      $\hat{d}_i \leftarrow \frac{d_i - a_i \hat{d}_{i-1}}{b_i - a_i \hat{c}_{i-1}}$ 
9:   end for
10:   $x_n \leftarrow \hat{d}_n$ 
11:  for  $i = n - 1, n - 2, \dots, 1$  do ▷ Backwards substitution
12:     $x_i \leftarrow \hat{d}_i - \hat{c}_i x_{i+1}$ 
13:  end for
14:  return x
15: end function

```

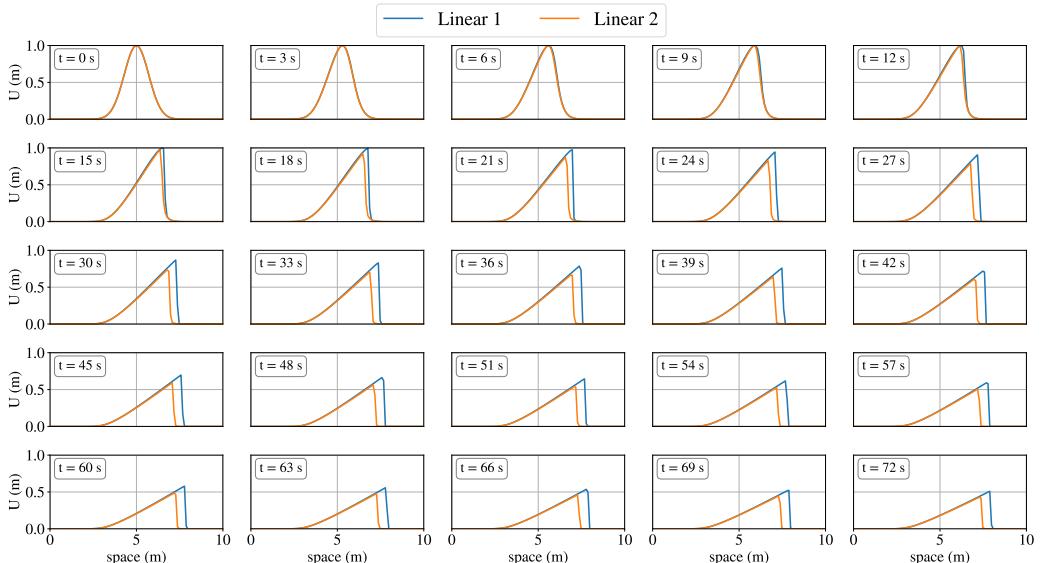


Abbildung 20.12: Lösung explizit

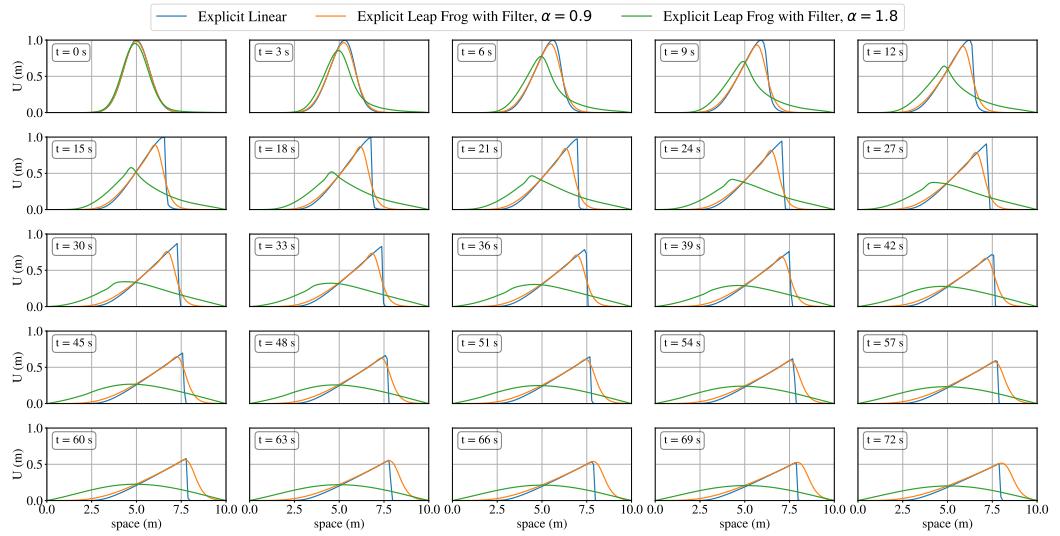


Abbildung 20.13: Lösung explizit mit linearem und Leap-Frog Ansatz, Tiefpassfilterung der Leap-Frog Lösung

Robert-Asselin-Time-Filter

In Abbildung 20.13 kann die Lösung mit dem beschriebenen Tiefpass-Filters aus Abschnitt 20.3.1 entnommen werden. Wird der Dämpfungsfaktor α zu gross gewählt, wird die Lösung zu stark gefiltert.

20.4.2 Implizit

Die Unterschiede bei den impliziten Verfahren sind in Abbildung 20.14 etwas besser ersichtlich. Es zeigt, dass der gewählte Punkt für die Ableitung bei der Leap-Frog Methode einen grösseren Einfluss hat. Die Welle ist etwas träger, als die der Lösung der quadratischen und linearen impliziten Varianten.

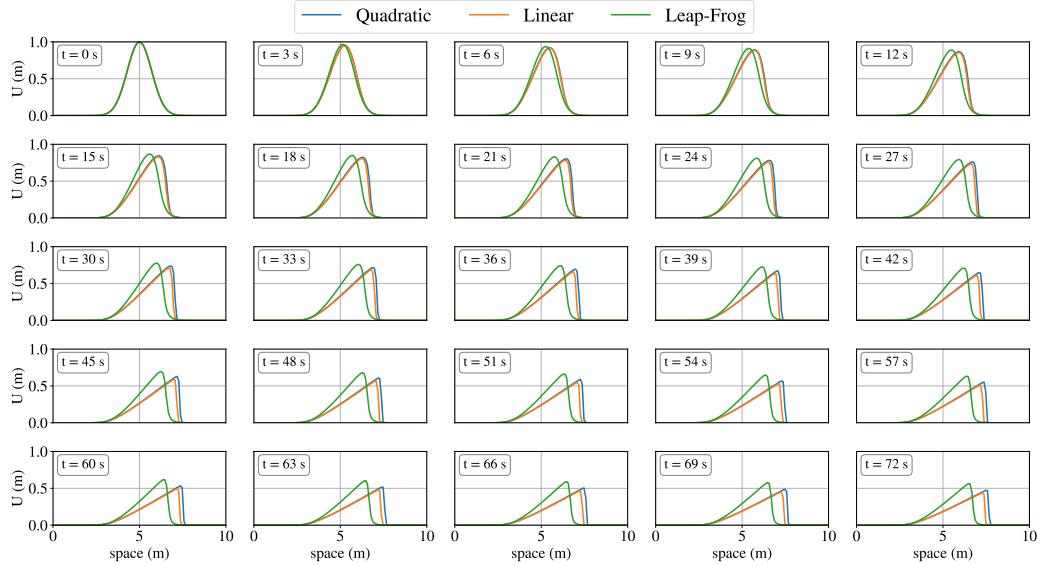


Abbildung 20.14: Lösung implizit

Literatur

- [1] Richard Asselin. “Frequency Filter for Time Integrations”. In: *Monthly Weather Review* 100.6 (Juni 1972), S. 487–490. issn: 0027-0644. url: [https://doi.org/10.1175/1520-0493\(1972\)100<0487:FFTI>2.3.CO;2](https://doi.org/10.1175/1520-0493(1972)100<0487:FFTI>2.3.CO;2).
- [2] *Burgers' equation*. 15. Mai 2020. url: <https://www.azimuthproject.org/azimuth/show/Burgers%27+equation>.
- [3] Peter Lynch. *The emergence of numerical weather prediction: Richardson's dream*. Cambridge University Press, 2014. isbn: 978-1-1074-1483-9.
- [4] *Thomas Algorithmus*. 29. Juni 2020. url: https://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm.

21

Finite Elemente in der Ebene

Joël Rechsteiner

21.1 Einleitung

Das Grundkonzept der Finiten Element Methode ist im Kapitel 7 beschrieben. Das aktuelle Kapitel beschränkt sich auf zwei Dimensionen bzw. auf FEM in der Ebene. Im Unterschied zur FEM in einer Dimension, wird es bei FEM in der Ebene keine perfekte Lösung geben wie z. B. die Splines. Das Verstehen im zweidimensionalen Fall bringt den Vorteil mit sich, dass die Vorgehensweise zum Lösen der FEM im dreidimensionalen Raum identisch ist. Dies gilt auch für noch höhere FEM Dimensionen. Auch hier ist das Ziel ein partielle Differentialgleichung (PDE) numerisch zu lösen. Es wird ein allgemeines Vorgehen beschrieben, um die umständlichen Ableitungen in Integralterme zu verwandeln und diese komplizierten Integrale wiederum in ein lineares Gleichungssystem zu überführen. Die Gleichungssysteme können dann mit einem Rechner gelöst werden. Solche numerische Approximationen werden z. B. in der Simulation von Feldlinien benötigt. In dem folgenden Abschnitt wird drauf eingegangen, wie die einzelnen Schritte umgesetzt werden und insbesondere werden die jeweiligen Nutzen dargestellt.

21.2 Problemstellung

Für die Erklärung der Finiten Element Methode in der Ebene wird hier als Beispiel die folgende partielle Differentialgleichung (Poisson-Gleichung) mit dem Eigenwertproblem der Form

$$\Delta u = \lambda u \quad (21.1)$$

verwendet, wobei der Laplace-Operator die zweiten Ableitungen nach den beiden Variablen darstellt:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \quad (21.2)$$

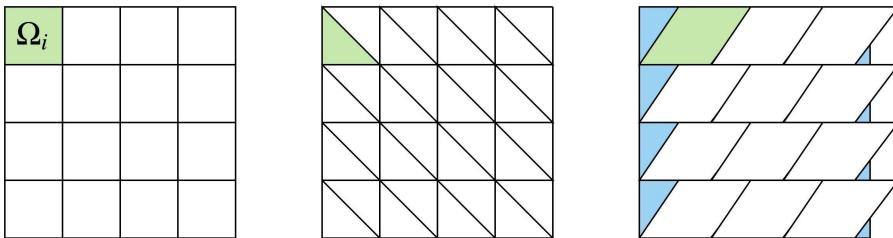


Abbildung 21.1: Approximationsmöglichkeiten eines Quadratischen Gebietes

Diese PDE muss nach Abschnitt 7.1.1 auf ein definiertes Gebiet Ω z. B. ein Rechteck mit Randwerten $= 0$ angewendet werden. Andere mögliche Randbedingungen sind in Abschnitt 7.1.3 zusammengestellt. Wie im Unterkapitel 7.4 behandelt, gibt es zu gewissen partiellen Differentialgleichungen der Form (21.1) für ein Rechteck ein äquivalentes Minimalproblem

$$I(u) = \int_a^b \int_c^d \nabla u(x)^2 + \lambda u(x)^2 dy dx. \quad (21.3)$$

Die Lösungsfunktion u muss so gewählt werden, dass $I(u)$ minimal wird. Das Gebiet Ω muss nicht zwingend ein Rechteck sein. Für ein beliebiges Gebiet $\Omega \subset \mathbb{R}^2$ wird das zu minimierende Integral

$$I(u) = \int_{\Omega} \nabla u(x)^2 + \lambda u(x)^2 dy dx \quad (21.4)$$

geschrieben.

21.2.1 Was sind Ansatzfunktionen?

Um eine gesuchte Funktion $u(x, y)$ als Lösung für die PDE (21.1) zu finden, wird diese mit einfacheren Funktionen approximiert. Diese einfachen Approximationen werden in diesem Kapitel als Ansatzfunktion $u(x, y)$ bezeichnet.

Als einfachere Funktionen bieten sich Polynome mit niedrigem Grad an. Diese sind einfach integrier- und differenzierbar. Während der Polynomgrad 1 sehr leicht zu berechnen ist, müssen beim Polynomgrad 2 aufwändiger, aber nicht komplexere, Berechnungen durchgeführt werden, da quadratische Ausdrücke und davon Kombinationen entstehen. Polynomgrade 3 und 4 sind mit noch mehr Fleissarbeit verbunden, bieten dafür eine bessere Approximation an. Wie im Kapitel 7 ist auch hier das Ziel, Gleichungen mit wenigen Unbekannten aufzustellen. Ist es möglich auch in der Ebene Ansatzfunktionen mit wenigen Koeffizienten zu finden, obwohl über ein Gebiet integriert wird? Es ist möglich, allerdings gibt es besondere Herausforderungen zu beachten, die in den nächsten Abschnitten genauer beschrieben werden.

21.2.2 Herausforderung FEM in der Ebene

Die einzelne lineare Ansatzfunktion kann nicht die globale Lösung sein. Ansonsten würde gemäß der Formel (21.4), die zweite Differenzierbarkeit fordert, als einzige Lösung 0 zustande kommen, da

lineare Terme als zweite Ableitung 0 aufweisen. Diese Lösung ist nicht sehr spannend. Daher muss das Gebiet Ω in Teilgebiete Ω_i unterteilt werden. Dadurch wird die Formel (21.4) zu

$$\int_{\Omega} \nabla u(x)^2 + \lambda u(x)^2 dx dy = \sum_{i=1}^n \int_{\Omega_i} \nabla u(x)^2 + \lambda u(x)^2 dx dy. \quad (21.5)$$

Die Lösung verwendet das äquivalente Minimalproblem, also Integrale statt Ableitungen der Unbekannten. Entsprechend dem Kapitel 7 sind die Integrale von quadratischen oder linearen Ausdrücken. Das Prinzip der FEM in zwei Dimensionen wird hier anhand von linearen Ausdrücken erläutert. Das Prinzip ist für quadratische Ausdrücke genau gleich, lediglich die Berechnung ist mit mehr Aufwand verbunden. Die Integrale werden dann auf jedes Teilgebiet angewendet. Damit eine Lösung gefunden werden kann, müssen die Teilgebiete einfach sein. Eine Approximation eines Gebietes kann beispielsweise mit Dreiecken, Rechtecken oder Parallelogrammen vorgenommen werden. Die Teilgebiete sind einfache geometrische Formen wie z. B. in Abbildung 21.1. Unterschiedliche geometrische Formen approximieren gut das gewünschte Gebiet unterschiedlich. Die Approximation mit Parallelogrammen z. B. deckt im Unterschied zu den anderen Approximationen das Gebiet weniger genau ab. Die Vierecke haben im Gegensatz zu den Dreiecken eine Unbekannte mehr bzw. einen Koeffizienten, mehr der bestimmt werden muss.

In einem nächsten Schritt muss die die approximierende Funktion betrachtet werden. Aus der Approximation resultieren zwei wesentliche Herausforderungen:

1. Ansatzfunktion muss stetig differenzierbar in den Stützstellen (Steigung) sein wie z. B. im Dreieck in den Ecken.
2. Sie muss stetig differenzierbar sein an den Übergängen entlang den Rändern eines Elements zum anliegenden Rand eines benachbarten Elements (Krümmung).

Um diese beiden Herausforderungen etwas besser verständlich zu machen, wurde in der Abbildung 21.1 und 21.2 eine Approximation der Funktion $u(x, y) = 1 - x^2 - y^2$ dargestellt, die eine Lösung der PDE $\Delta u = -4$ ist mit Randbedingungen $u = 0$ auf dem Einheitskreis (violett). Wie in der Abbildung 21.2 zu erkennen ist, passt die Approximation mit Dreiecken auf einem quadratischen Gitter nicht gut, da sich die Randbedingungen nicht ausnutzen lassen. Es benötigt also eine Unterteilung der Kreisscheibe die bis an den Rand kommt. Eine bessere Approximation ist in der Abbildung 21.3 zu sehen, die eine bessere Triangulation zeigt. Aus der Abbildung 21.3 lässt sich erkennen, dass die Ansatzfunktion für jedes Dreieck unterschiedlich ist. Auch müssen die Ansatzfunktionen flexibel auf die Grösse des Flächenelements anpassbar sein. Die Knicke von zwei benachbarten Elementen wird in der folgenden Beschreibung der Einfachheit halber ignoriert, um das Lösungsvorgehen verständlicher aufzuzeigen. Bessere Ansatzfunktionen ergeben bessere Lösungen, aber die Vorgehensweise ist analog zu den linearen Ansatzfunktionen. Aus den Integralen der Teilgebiete muss dann die Lösung gefunden werden, um die Werte in den Ecken zu bestimmen. Um den Aufwand der Berechnung über das Flächenelement zu minimieren, wird in Abschnitt 21.3 auch eine Lösung aufgezeigt, wie mit Hilfe einer Transformation das Dreiecksflächenelement in ein weniger aufwändig berechenbares Flächenelement überführt werden kann. Nochmals kurz zusammengefasst was bis hier hin aufgezeigt wurde:

- Ebene wird in Teilgebiete unterteilt
- Teilgebiete können Dreiecke, Rechtecke oder Parallelogramme sein
- Ansatzfunktion soll auf jedes Teilgebiet ein Polynom niedrigen Grades sein

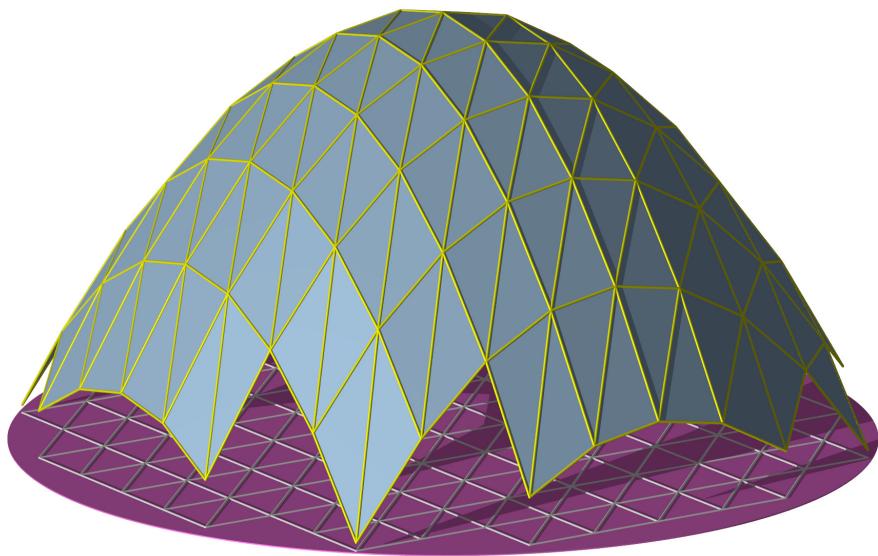


Abbildung 21.2: Approximation mit Dreiecken auf einem quadratischen Gitter

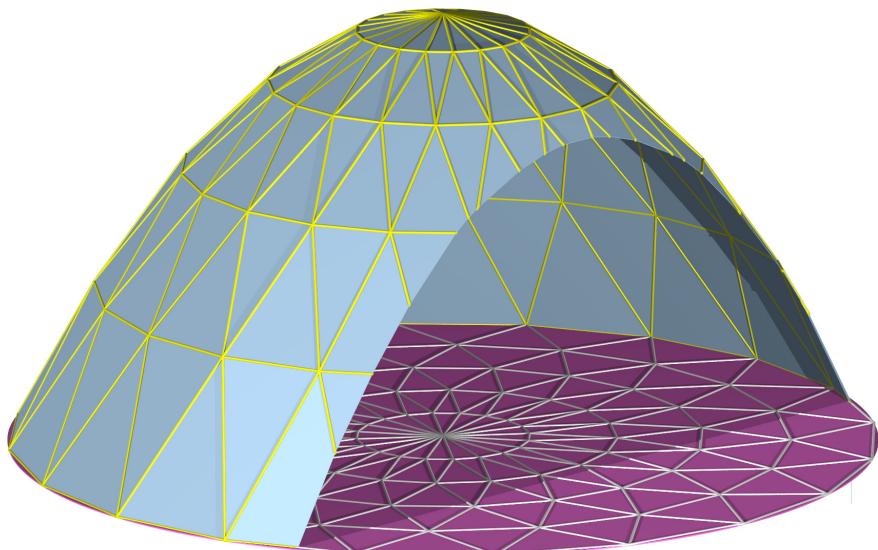


Abbildung 21.3: bessere Triangulation

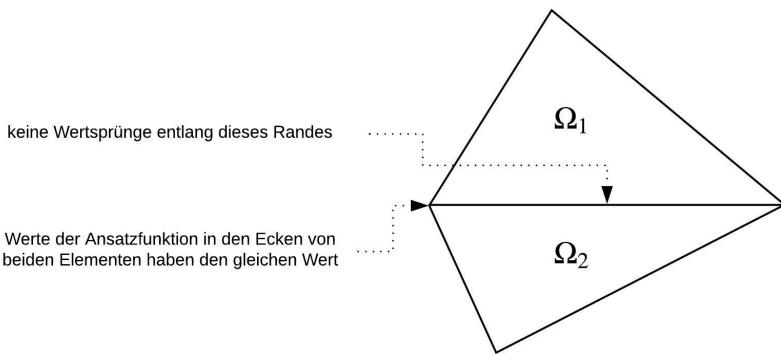


Abbildung 21.4: differenzierbar entlang eines Randes

- auf jedes Teilgebiet wird eine Ansatzfunktion angewendet

Was bis jetzt noch nicht klar ist, wie die Ansatzfunktion sich zusammenstellt. Dies wird im folgenden Abschnitt beschrieben.

21.3 Lösung

Wie im vorherigen Kapitel erwähnt, sind Herausforderungen bzw. Ansprüche an die Ansatzfunktionen zu beachten. Nun stellt sich die Frage, wie die Ansatzfunktion gewählt werden müssen, so dass entlang eines Randes nur "Knicke" entstehen und keine Sprünge. Und wie ist gewährleistet, dass in jeder Stützstelle alle Elemente den gleichen Wert haben? Als Lösung bietet sich ein lineare Funktion an. Für ein Dreieck kann der Term $u(x) = c_1 + c_2x + c_3y$ verwendet werden. Diese Zusammensetzung ergibt sich, da der lineare Ansatz durch die Funktionswerte in den Ecken des Dreiecks bestimmt wird. In anderen Worten sind die Funktionswerte in den Ecken des Teilgebiets, hier im Dreieck, die Unbekannten, die bestimmt werden müssen. Wie diese Koeffizienten in einer Matrix bestimmt werden ist in 21.3.3 aufgezeigt.

Nochmals zur Verdeutlichung warum ein linearer Ansatzfunktion verwendet wird:

- eine einfache Formel gibt es nur auf einem kleinen Teilgebiet
- in dem Teilgebiet ist die Integralformel sehr einfach zu berechnen wie z. B. eine lineare Funktion die durch Koeffizienten bestimmt wird.

Um möglichst einfach darzustellen, wie das Verfahren der finiten Elemente vonstatten geht, wurde auf eine einfache Approximation mit Einheitsdreiecken und einem linearen Ansatz gewählt.

21.3.1 Vorbereitung

Damit die Integration über das gewählte Teilgebiet einfacher fällt, wird als Vorbereitung im entsprechenden Fall das gewählte Gebiet in ein Einheitsdreieck oder in ein Einheitsparallelogramm

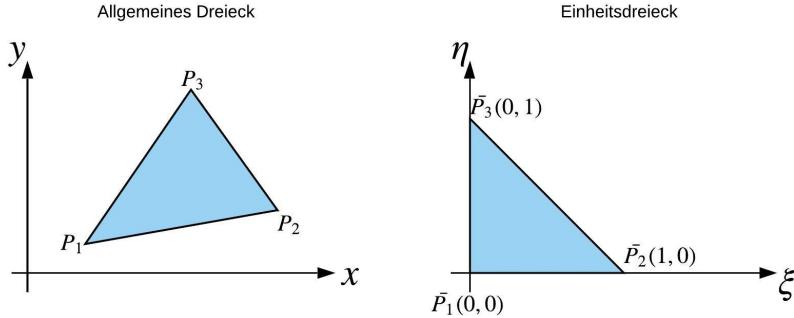


Abbildung 21.5: Transformation in ein Einheitsdreieck

transformiert. So können alle Gebiete gleich behandelt und berechnet werden. Das allgemeine Dreieck in allgemeiner Lage mit den Eckpunkten $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ und $P_3(x_3, y_3)$ kann mit Hilfe der linearen Transformation

$$\begin{aligned} x &= x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta \\ y &= y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta \end{aligned} \quad (21.6)$$

in das einfachere Gebiet nämlich das gleichschenklig rechtwinklige Einheitsdreieck mit Kathetenlänge 1 überführt werden, wie in der Abbildung 21.5 dargestellt ist.

Die Variablen ξ und η werden die neuen Integrationsvariablen. Damit die Berechnung korrekt ist, muss noch auf der rechten Seite die Determinante der Jacobi-Matrix

$$dy dy = \det(J) d\xi d\eta \quad (21.7)$$

dazu multipliziert werden. Die Jacobi-Matrix

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}. \quad (21.8)$$

besteht aus den partiellen Ableitungen aus (21.6). Aus der Transformation (21.6) erhält man

$$\det(J) = x_1 \cdot y_2 - 2 \cdot x_1 \cdot y_1 + x_2 \cdot y_1 + x_1 \cdot y_3 + x_3 \cdot y_1 - x_2 \cdot y_3 - x_3 \cdot y_2. \quad (21.9)$$

Die Jacobi-Determinante (21.9) entspricht der doppelten Fläche des Dreiecks. Die partiellen Ableitungen werden gemäss der Kettenregel zu

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{\partial u}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial u}{\partial \eta} \frac{\partial \eta}{\partial x} \\ \frac{\partial u}{\partial y} &= \frac{\partial u}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial u}{\partial \eta} \frac{\partial \eta}{\partial y}. \end{aligned} \quad (21.10)$$

Auf Grund der linearen Transformation (21.6) ergeben sich nach partieller Differentiation der beiden Beziehungen nach x zunächst

$$\begin{aligned} 1 &= (x_2 - x_1) \frac{\partial \xi}{\partial x} + (x_3 - x_1) \frac{\partial \eta}{\partial x} \\ 0 &= (y_2 - y_1) \frac{\partial \xi}{\partial x} + (y_3 - y_1) \frac{\partial \eta}{\partial x}. \end{aligned} \quad (21.11)$$

Nach Auflösung der beiden Gleichungen nach $\frac{\partial \xi}{\partial x}$ und $\frac{\partial \eta}{\partial x}$ ergibt sich

$$\frac{\partial \xi}{\partial x} = \frac{y_3 - y_1}{J} \quad \text{und} \quad \frac{\partial \eta}{\partial x} = -\frac{y_2 - y_1}{J}. \quad (21.12)$$

Analog nach der partiellen Ableitung nach y erhält man

$$\frac{\partial \xi}{\partial y} = \frac{x_3 - x_1}{J} \quad \text{und} \quad \frac{\partial \eta}{\partial y} = -\frac{x_2 - x_1}{J}. \quad (21.13)$$

Somit vereinfacht sich die Integration über das Einheitsdreieck zu

$$\int_{\Delta} u \, dy \, dx = \int_0^1 \int_0^{1-\xi} u \det(J) \, d\eta \, d\xi. \quad (21.14)$$

21.3.2 Schritt 1: Minimalproblem bilden

Analog zum Kapitel 7.4.1 muss auch die DGL in der Ebene zuerst in ein äquivalentes Minimalproblem übersetzt werden. Das Minimalproblem für FEM in einer Dimension hatte die Form:

$$\int_0^1 u'(x)^2 - \lambda u(x)^2 \, dx \, dy. \quad (21.15)$$

Im Unterschied zu den mehrdimensionale Form wird anstatt der einfachen Ableitung der Laplace-Operator verwendet, der zugleich zweifache Differenzierbarkeit von der Ansatzfunktion $u(x)$ fordert. Zudem wird nicht mehr über eine Strecke integriert sondern über ein Teilgebiet Ω_i :

$$\int_{\Omega_i} (\nabla u)^2 - \lambda u^2 \, dx \, dy. \quad (21.16)$$

Der Integrationsterm (21.16) wird gemäss der Summenregel in zwei Terme zerlegt nämlich

$$\int_{\Omega_i} (\nabla u)^2 \, dx \, dy - \lambda \int_{\Omega_i} u^2 \, dx \, dy. \quad (21.17)$$

In einem nächsten Schritt wird das Minimalproblem auf das Element eingeschränkt. Dies wird im folgenden Abschnitt erläutert.

21.3.3 Schritt 2: Lösung durch Ansatzfunktionen approximieren

Die lineare Ansatzfunktion hat wie schon erwähnt die Form

$$u(\xi, \eta) = c_0 + c_1 \xi + c_2 \eta. \quad (21.18)$$

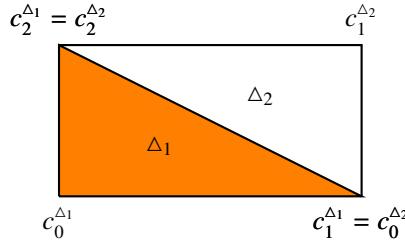
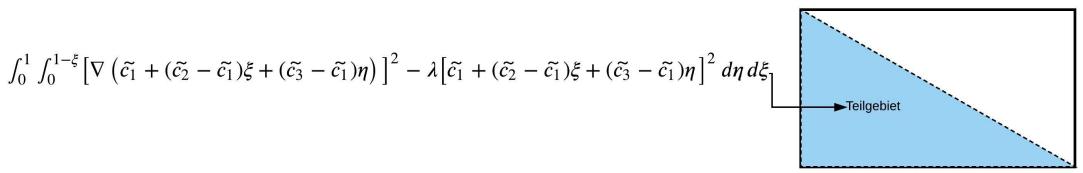
Abbildung 21.6: Stützstellen der Elemente \int_{Ω_1} und \int_{Ω_2} 

Abbildung 21.7: Anwenden der Formel (21.20) auf jedes Teilgebiet

Bei näherer Betrachtung fällt jedoch auf, dass in (21.18) die Parameter c_1, c_2 und c_3 von einem Dreieck zu einem anderen Dreieck unterscheiden. Die Ursache liegt insbesondere darin, dass c_2 und c_3 von den Kanten abhängig sind und die Kanten wiederum können von Dreieck zu Dreieck variieren (Ω_i, Ω_j). Daher ist die lineare Ansatzfunktion

$$u(\xi, \eta) = \tilde{c}_1 + (\tilde{c}_2 - \tilde{c}_1)\xi + (\tilde{c}_3 - \tilde{c}_1)\eta \quad (21.19)$$

besser geeignet. In (21.19) sind die c_i^N die Funktionswerte in den Ecken und eignen sich somit viel besser, da nur die Eckwerte ohne Einfluss der Kanten berücksichtigt werden wie in Abbildung 21.6 dargestellt. Die Ansatzfunktion (21.19) kann somit direkt in die Formel (21.16) für ein Teilgebiet eingesetzt werden. Ziel ist, dass die unbekannten Koeffizienten c_1, c_2 und c_3 anhand dieses Verfahrens bestimmt werden. Das zu lösende Integral für ein Teilgebiet hat die Form

$$\int_0^1 \int_0^{1-\xi} [\nabla(\tilde{c}_1 + (\tilde{c}_2 - \tilde{c}_1)\xi + (\tilde{c}_3 - \tilde{c}_1)\eta)]^2 - \lambda[\tilde{c}_1 + (\tilde{c}_2 - \tilde{c}_1)\xi + (\tilde{c}_3 - \tilde{c}_1)\eta]^2 d\eta d\xi. \quad (21.20)$$

Allerdings sollte gemäss (21.17) die Formel (21.20) aufgeteilt werden in

$$\int_0^1 \int_0^{1-\xi} [\nabla(\tilde{c}_1 + (\tilde{c}_2 - \tilde{c}_1)\xi + (\tilde{c}_3 - \tilde{c}_1)\eta)]^2 d\eta d\xi - \lambda \int_0^1 \int_0^{1-\xi} [\tilde{c}_1 + (\tilde{c}_2 - \tilde{c}_1)\xi + (\tilde{c}_3 - \tilde{c}_1)\eta]^2 d\eta d\xi. \quad (21.21)$$

Berechnung der Integrale

Im ersten Integralterm von (21.21) wird der Gradienten von $u(\xi, \eta)$ quadriert benötigt. Der Gradient von $u(\xi, \eta)$ ist

$$\nabla u = \begin{bmatrix} \tilde{c}_2 - \tilde{c}_1 \\ \tilde{c}_3 - \tilde{c}_1 \end{bmatrix}. \quad (21.22)$$

Der Koeffizient \tilde{c}_1 im ersten Integralterm von (21.21) verschwindet in (21.22). Der Gradient (21.22) quadriert ergibt

$$(\nabla u)^2 = \tilde{c}_2^2 - 2\tilde{c}_1\tilde{c}_2 + \tilde{c}_1^2 + \tilde{c}_3^2 - 2\tilde{c}_1\tilde{c}_3 + \tilde{c}_1^2 = 2\tilde{c}_1^2 + \tilde{c}_2^2 + \tilde{c}_3^2 - 2\tilde{c}_1\tilde{c}_2 - 2\tilde{c}_1\tilde{c}_3. \quad (21.23)$$

Im zweite Integralterm wird

$$\begin{aligned} u(\xi, \eta)^2 &= \tilde{c}_1^2\xi^2 - 2\tilde{c}_1\tilde{c}_2\xi^2 + \tilde{c}_2^2\xi^2 + 2\tilde{c}_1^2\xi\eta - 2\tilde{c}_1\tilde{c}_2\xi\eta - 2\tilde{c}_1\tilde{c}_3\xi\eta + 2\tilde{c}_2\tilde{c}_3\xi\eta - 2\tilde{c}_1^2\xi + \\ &\quad 2\tilde{c}_1\tilde{c}_2\xi + \tilde{c}_1^2\eta^2 - 2\tilde{c}_1\tilde{c}_3\eta^2 + \tilde{c}_3^2\eta^2 - 2\tilde{c}_1^2\eta + 2\tilde{c}_1\tilde{c}_3\eta + \tilde{c}_1^2. \end{aligned} \quad (21.24)$$

Wie in (21.24) zusehen ist, ist die Berechnung nicht anspruchsvoll, aber aufwändig. Nun ist aber noch die Frage offen wie die Koeffizienten in Matrixform zustande kommen. In folgenden Abschnitt wird dies anhand eines Beispiels beschrieben.

Matrixform der Integrale

Da die Ansatzfunktion (21.19) zwar unabhängig von den Kanten ist, aber die Berechnung des Quadrates von u (21.24) unübersichtlich, wird in diesem Beispiel in diesem Abschnitt die lineare Ansatzfunktion (21.18) verwendet. Somit ist

$$\nabla u = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \Rightarrow (\nabla u)^2 = c_1^2 + c_2^2 \quad (21.25)$$

und

$$u^2 = \color{blue}{c_0^2} + \color{red}{c_1^2x^2} + \color{blue}{c_2^2y^2} + \color{red}{2c_0c_1x} + \color{red}{2c_0c_2y} + \color{red}{2c_1c_2xy}. \quad (21.26)$$

Bekannt ist nun, wie die Integrale für die einzelne Teilgebiete aussehen. Um eine Matrixnotation der Koeffizienten c_i zu erhalten, um dann ein Gleichungssystem für deren Berechnung aufzubauen, werden die Koeffizienten in Vektorform

$$\begin{bmatrix} c_0^{(\Delta_i)} \\ c_1^{(\Delta_i)} \\ c_2^{(\Delta_i)} \end{bmatrix} \quad (21.27)$$

aufgeschrieben. In (21.27) ist die Vektornotation der Koeffizienten von (21.18). Nun wird der Integralterm

$$\underbrace{\int_{\Omega_i} (\nabla u)^2 d\xi d\eta}_{\text{1. Term}} - \underbrace{\lambda \int_{\Omega_i} u^2 d\xi d\eta}_{\text{2. Term}} \quad (21.28)$$

eines Teilgebietes in die lineare Notation

$$\underbrace{c^t A c}_{\text{1. Term}} - \underbrace{\lambda c^t B c}_{\text{2. Term}} \quad (21.29)$$

gebracht. Gemäss (21.28) sind Quadrate der Koeffizienten c_i gefordert. Um nun quadratische Koeffizienten in der Matrixschreibweise zu erhalten wird dies wie folgt notiert:

$$\begin{bmatrix} c_0^{(\Delta_i)} & c_1^{(\Delta_i)} & c_2^{(\Delta_i)} \end{bmatrix} \begin{bmatrix} \ddots & \cdots & \cdots \\ \vdots & \ddots & \ddots \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} c_0^{(\Delta_i)} \\ c_1^{(\Delta_i)} \\ c_2^{(\Delta_i)} \end{bmatrix} + \lambda \begin{bmatrix} c_0^{(\Delta_i)} & c_1^{(\Delta_i)} & c_2^{(\Delta_i)} \end{bmatrix} \begin{bmatrix} \ddots & \cdots & \cdots \\ \vdots & \ddots & \ddots \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} c_0^{(\Delta_i)} \\ c_1^{(\Delta_i)} \\ c_2^{(\Delta_i)} \end{bmatrix}. \quad (21.30)$$

Dies ist auch die Begründung, warum Formel (21.29) verwendet wird. Die Matrix A wird gebildet durch die Teilmatrizen eines jeden Teilgebietes. Die Teilmatrix eines Teilgebiet besteht aus dem Integral des 1. Terms von (21.28)

$$\int_0^1 \int_0^{1-\xi} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}^2 d\eta d\xi \quad (21.31)$$

was dann zu der entsprechenden Teilmatrix

$$A_{\Delta_i} = \begin{pmatrix} \int_{\Omega_i} 1 d\eta d\xi & 0 \\ 0 & \int_{\Omega_i} 1 d\eta d\xi \end{pmatrix} \quad (21.32)$$

Hier gibt es ein Problem. Der Koeffizient c_0 ist hier nicht berücksichtigt. Es wird eine Matrix A_{Δ_i} derart benötigt so, dass

$$\int_{\Delta_i} (\nabla u)^2 d\xi d\eta = c^t A c \quad (21.33)$$

Daher wird die Matrix

$$A_{\Delta_i} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \int_{\Omega_i} 1 d\eta d\xi & 0 \\ 0 & 0 & \int_{\Omega_i} 1 d\eta d\xi \end{pmatrix} \quad (21.34)$$

mit Nullen entsprechend angepasst. Die Teilmatrix ausgerechnet ergibt

$$A_{\Delta_i} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \quad (21.35)$$

und wird dann in die grosse Matrix A eingefügt. Wie genau die Teilmatrizen A_{Δ_i} in eine gesamte Matrix A zusammengesetzt werden, wird weiter unten am Beispiel der Matrix B erläutert.

Die Matrix A ist nun bekannt. Es fehlt jedoch noch die Matrix B , um das Gleichungssystem zu komplettieren. Die Matrix B wird nun wie folgt definiert. Um die folgende Form in (21.36) zu erhalten muss lediglich der zweite Term von (21.28) für die einzelnen Koeffizienten berechnet werden, was dann den folgenden Ausdruck

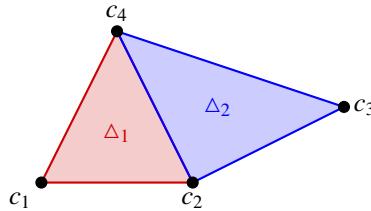
$$\int_0^1 \int_0^{1-\xi} c_0^2 + c_1^2 x^2 + c_2^2 y^2 + 2c_0 c_1 x + 2c_0 c_2 y + 2c_1 c_2 xy d\eta d\xi \quad (21.36)$$

ergibt. Nun soll dies wieder in die Matrixschreibweise übertragen werden was dann einer Teilmatrix

$$B_{\Delta_i} = \begin{pmatrix} -\lambda \int_{\Omega_i} 1 & -\lambda \int_{\Omega_i} x & -\lambda \int_{\Omega_i} y \\ -\lambda \int_{\Omega_i} x & -\lambda \int_{\Omega_i} x^2 & -\lambda \int_{\Omega_i} xy \\ -\lambda \int_{\Omega_i} y & -\lambda \int_{\Omega_i} xy & -\lambda \int_{\Omega_i} y^2 \end{pmatrix} \quad (21.37)$$

eines Dreiecks entspricht und ausgerechnet

$$B_{\Delta_i} = -\lambda \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{3} \\ \frac{1}{6} & \frac{1}{12} & \frac{1}{24} \\ \frac{1}{3} & \frac{1}{24} & \frac{1}{12} \end{pmatrix} \quad (21.38)$$

Abbildung 21.8: Zwei Dreiecke mit gemeinsamen Stützstellen c_2 und c_4

ergibt.

Zusammensetzen der Teilmatrizen A und B

Anhand eines Beispieles soll gezeigt werden wie sich die B Matrix aus den Teilmatrizen B_{Δ_i} zusammenstellt. Als Beispiel dient eine Approximation mit den beiden Dreiecken Δ_1 und Δ_2 . Das Dreieckgebiet Δ_1 hat

$$\int_{\Delta_1} u^2 d\eta d\xi = [c_1 \quad c_2 \quad c_4] \begin{bmatrix} & B_1 & \\ & & \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_4 \end{bmatrix} \quad (21.39)$$

als Matrixform und das Gebiet Δ_2

$$\int_{\Delta_2} u^2 d\eta d\xi = [c_2 \quad c_3 \quad c_4] \begin{bmatrix} & B_2 & \\ & & \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \\ c_4 \end{bmatrix}. \quad (21.40)$$

Beide Gebiete zusammen entsprechen

$$\int_{\Delta_1 \text{ und } \Delta_2} u^2 d\eta d\xi = \int_{\Delta_1} u^2 d\eta d\xi + \int_{\Delta_2} u^2 d\eta d\xi = c'_{\Delta_1} B_1 c_{\Delta_1} + c'_{\Delta_2} B_2 c_{\Delta_2}. \quad (21.41)$$

Die Koeffizienten von Δ_1 sind

$$\int_{\Delta_1} \underbrace{c_1^2}_{\frac{1}{2}} + \underbrace{c_2^2 \xi^2}_{\frac{1}{12}} + \underbrace{c_4^2 \eta^2}_{\frac{1}{12}} + \underbrace{2c_1 c_2 \xi}_{\frac{1}{6}} + \underbrace{2c_1 c_4 \eta}_{\frac{1}{3}} + \underbrace{2c_2 c_4}_{\frac{1}{24}} d\eta d\xi \quad (21.42)$$

und von Δ_2

$$\int_{\Delta_2} \underbrace{c_2^2}_{\frac{1}{2}} + \underbrace{c_3^2 \xi^2}_{\frac{1}{12}} + \underbrace{c_4^2 \eta^2}_{\frac{1}{12}} + \underbrace{2c_2 c_3 \xi}_{\frac{1}{6}} + \underbrace{2c_2 c_4 \eta}_{\frac{1}{3}} + \underbrace{2c_3 c_4}_{\frac{1}{24}} d\eta d\xi. \quad (21.43)$$

Bevor die roten und blauen Koeffizienten von (21.42) und (21.43) der jeweiligen Gebiete zusammengesetzt werden können, gemäss der Abbildung 21.9, müssen zwei Punkte berücksichtigt werden:

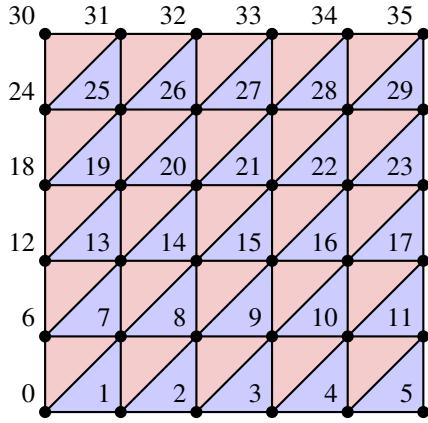
- Jeder Koeffizient muss nach (21.7) mit dem dazugehörigen Jacobi-Determinante (21.9) des eigenen Dreiecks multipliziert werden.
- Es müssen die \tilde{c} -Variablen verwendet werden. Die Umrechnung ist in 21.3.1 beschrieben.

$$B_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{3} \\ \frac{1}{6} & \frac{1}{12} & \frac{1}{24} \\ \frac{1}{3} & \frac{1}{24} & \frac{1}{12} \end{pmatrix}$$

$$B_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{3} \\ \frac{1}{6} & \frac{1}{12} & \frac{1}{24} \\ \frac{1}{3} & \frac{1}{24} & \frac{1}{12} \end{pmatrix}$$

↓ ↓

$$B = \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & & \frac{1}{3} \\ \frac{1}{6} & \frac{1}{12} + \frac{1}{2} & \frac{1}{6} & \frac{1}{24} + \frac{1}{3} \\ & \frac{1}{6} & \frac{1}{12} & \frac{1}{24} \\ \frac{1}{3} & \frac{1}{24} + \frac{1}{3} & \frac{1}{24} & \frac{1}{12} + \frac{1}{12} \end{pmatrix}$$

Abbildung 21.9: Matrix B zusammenstellen

$$B = \left(\begin{array}{cccccc} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{array} \right)$$

Abbildung 21.10: 5×5

Sind die beiden Schritte durchgeführt, können die Koeffizienten gemäss der Abbildung 21.9 die Matrix B gebildet werden. Das Vorgehen für die Matrix A ist genau gleich. Wie sieht die Matrix aus wenn mit mehr Dreiecken approximiert wird? Bei einem 5×5 Gitter mit 36 Variablen hätte die Matrix B die Form wie in Abbildung 21.10.

21.3.4 Schritt 3: Minimalprinzip anwenden auf Approximation

Nun folgt der nächste Schritt nämlich das Minimieren des quadratischen Ausdrucks

$$c^t A c - c^t \lambda B c. \quad (21.44)$$

Der Ausdruck (21.44) nach den Koeffizienten c_k abgeleitet ergibt den für 1. Term

$$\nabla c^t A c = 2 A c \quad (21.45)$$

und für den 2. Term den Ausdruck

$$\nabla c^t \lambda B c = 2\lambda B c. \quad (21.46)$$

Diese Ableitungen erscheinen nicht gerade offensichtlich, insbesondere der Faktor 2. Das Ableiten im eindimensionalen Raum ergibt nach der klassischen Analysis

$$\frac{d}{dx} ax^2 = 2ax. \quad (21.47)$$

Die Ableitung von $x^t A x$, $x \in \mathbb{R}^n$ nach x_n ergibt nach der Produktregel

$$\frac{\partial}{\partial x_n} x^t A x = \sum_{i,j} \frac{\partial x_i}{\partial x_n} a_{ij} x_j + \sum_{i,j} x_i a_{ij} \frac{\partial x_j}{\partial x_n}. \quad (21.48)$$

Dies kann vereinfacht werden anhand der Kürzungsberechnung

$$\sum_j a_{nj} x_j + \sum_{i,j} x_i a_{in} = \sum_j a_{nj} x_j + \sum_{j \neq i} a_{nj} x_{ji} = 2(Ax)_n. \quad (21.49)$$

21.3.5 Schritt 4: Gleichungssystem aufstellen

Nun kann das Gleichungssystem aufgestellt werden:

$$2Ac - 2\lambda Bc = 0 \Rightarrow (A - \lambda B)c = 0. \quad (21.50)$$

Durch das Umschreiben der Gleichung (21.50) in

$$B^{-1}Ac = \lambda c \quad (21.51)$$

ist ersichtlich, dass es sich um ein Eigenwertproblem für die Matrix $B^{-1}A$ handelt. Dieses kann z. B. mit dem Francis-Algorithmus, beschrieben im Kapitel 22, gelöst werden.

21.3.6 Andere lineare Ansatzfunktionen

Neben dem durchgerechneten Beispiel des Einheitsdreiecks kann auch eine andere Gebietsform verwendet werden. Die Funktion

$$u(x, y) = c_1 + c_2x + c_3y + c_4xy \quad (21.52)$$

gilt für einen möglichen Ansatz auf einem Parallelogramm.

Quadratischer Ansatz

Der Vorteil des quadratischen Ansatzes liegt darin, dass die Zahl der Freiheitsgrade erhöht wird. So mit kann unter Umständen eine bessere Approximation erreicht werden. Zu beachten ist allerdings, dass diese die Matrix enorm aufblasen bzw. vergrößern und dann mehr Ressourcen in Anspruch nehmen, um die Matrix bzw. die Eigenwerte zu berechnen.

In dem Ausdruck

$$u(x, y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 \quad (21.53)$$

ist der Ansatz für das Einheitsdreieck gegeben. Und in der Formel

$$u(x, y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 + c_7x^2y + c_8xy^2 \quad (21.54)$$

der quadratische Ansatz für ein Einheitsparallelogramm.

21.4 Folgerungen

In diesem Kapitel wurde die Bildung von Ansatzfunktionen in der Ebene eingeführt sowie die Eigenschaften einiger Typen aufgezeigt. Es wurde dabei gezeigt,

- dass die Wahl der Ansatzfunktionen bestimmte Voraussetzungen erfüllen müssen
- dass es unterschiedliche Arten von Ansatzfunktionen gibt
- was der Sinn dieser Ansatzfunktionen ist
- wie die Matrizen mit den Koeffizienten aufgestellt werden

Auch wurde gezeigt wie die Berechnungen vereinfacht werden durch die Transformation in eine Einheitsform wie dem Einheitsdreieck.

22

QR-Zerlegung mit Givens-Rotationen

Manuel Tischhauser

22.1 Einleitung

Wie in Abschnitt 6.4 schon genauer betrachtet, kann eine Matrix A mit l Zeilen und n linear unabhängigen Spalten ($l \geq n$) in zwei Matrizen zerlegt werden:

$$A = QR.$$

Q ist dabei wieder eine $l \times n$ -Matrix, diesmal aber mit orthonormierten Spalten. R ist eine obere Dreiecksmatrix der Grösse $n \times n$.

22.1.1 Anwendungsbeispiel Least-Squares

Die QR -Zerlegung kann unter anderem beim Lösen von Gleichungssystemen angewendet werden. Ein lineares Gleichungssystem

$$Ax = b \tag{22.1}$$

hat in praktischen Fällen keine Lösung, wenn die $l \times n$ -Matrix A mehr Zeilen als Spalten hat, bzw. überdefiniert ist.

Mit der Methode der kleinsten Quadrate kann aber ein Vektor \hat{x} gefunden werden welcher die L^2 -Norm

$$\|A\hat{x} - b\|_2$$

minimiert. Dies ist das Gleiche, wie wenn man einen Lösungsvektor \hat{x} sucht, sodass

$$A\hat{x} = b - b_{\perp}.$$

b_{\perp} ist hier die zu den Spalten in A orthogonale Komponente von b und ist folglich der Teil von b , welcher nicht linear abhängig von den Spalten von A ist.

Werden beide Seiten mit A^T multipliziert, erhält man

$$A^T A \hat{x} = A^T b - \underbrace{A^T b_{\perp}}_{= 0} \Leftrightarrow A^T A \hat{x} = A^T b, \quad (22.2)$$

wobei der letzte Term verschwindet, da alle Spalten von A und b_{\perp} orthogonal zueinander stehen und somit alle Skalarprodukte verschwinden. Das Matrixprodukt $A^T A$ ist unter Umständen aufwändig zu berechnen und muss auch noch invertiert werden, um \hat{x} zu erhalten.

Geht man nun aber den Umweg über die QR -Zerlegung, bzw. $A = QR$ kann die Gleichung (22.2) umgeschrieben werden:

$$(QR)^T QR \hat{x} = (QR)^T b \Leftrightarrow R^T \underbrace{Q^T Q}_{= E} R \hat{x} = R^T Q^T b \Leftrightarrow R^T R \hat{x} = R^T Q^T b,$$

wobei $Q^T Q = E$, da Q nach Definition orthogonal ist. Die Determinante von R^T ist

$$\det(R^T) = \det(R) = \det \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{pmatrix} = \prod_{k=1}^n r_{kk}.$$

Diese verschwindet nur, wenn eines der Diagonalelemente verschwindet. Dies würde aber bedeuten, dass die Spalten von Q und somit auch A linear abhängig voneinander sind, womit wiederum die Voraussetzungen für die QR -Zerlegung nicht gegeben wären. R^T ist somit immer regulär und kann invertiert werden, was eine weitere Vereinfachung ermöglicht:

$$(R^T)^{-1} R^T R \hat{x} = (R^T)^{-1} R^T Q b \Leftrightarrow R \hat{x} = Q^T b,$$

und weiter vereinfacht

$$\hat{x} = R^{-1} Q^T b, \quad (22.3)$$

was somit das Least-Squares-Problem mit dem Umweg über die QR -Zerlegung auf kompakte Weise löst.

22.2 Numerische Probleme des Gram-Schmidt-Verfahrens

In Abschnitt 6.4.1 wurde das Gram-Schmidt-Orthonormalisierungsverfahren vorgestellt. Es liefert zwar eine sehr anschauliche Methode zur QR -Zerlegung, ist aber numerisch nicht stabil wenn die die Spaltenvektoren von A nahezu parallel sind.

Dies soll nun Anhand der Matrix (von [1] übernommen)

$$A = \begin{pmatrix} 1 + \epsilon & 1 & \\ 1 & 1 + \epsilon & 1 \\ 1 & 1 & 1 + \epsilon \end{pmatrix}$$

genauer untersucht werden. ϵ ist dabei eine „kleine“ Zahl, sodass ϵ^2 vernachlässigt werden kann bzw. $\epsilon^2 \approx 0$. Nach Gram-Schmidt wird die erste Komponente von Q berechnet als

$$q_1 = \frac{a_1}{|a_1|} = \frac{1}{\sqrt{3 + 2\epsilon + \epsilon^2}} \begin{pmatrix} 1 + \epsilon \\ 1 \\ 1 \end{pmatrix} \approx \frac{1}{\sqrt{3 + 2\epsilon}} \begin{pmatrix} 1 + \epsilon \\ 1 \\ 1 \end{pmatrix}.$$

Schon hier kommt es also zu einem kleinen Rundungsfehler.

Die zweite Spalte von Q berechnet sich als

$$q_2 = \frac{a_2 - (q_1 \cdot a_2)q_1}{|a_2 - (q_1 \cdot a_2)q_1|} = \frac{z_1}{|z_1|}.$$

Dies kann in zwei Schritten ausgeführt werden. Zuerst wird z_1 ausgerechnet und dann nach dessen Betrag normiert, um auf q_2 zu kommen. Im ersten Schritt also:

$$z_1 = \begin{pmatrix} 1 \\ 1 + \epsilon \\ 1 \end{pmatrix} - \underbrace{\frac{1}{\sqrt{3+2\epsilon}}(3+2\epsilon)}_{=1} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 + \epsilon \\ \epsilon \\ 0 \end{pmatrix}$$

und

$$|z_1| = \sqrt{(-\epsilon)^2 + \epsilon^2 + 0^2} = \sqrt{2\epsilon^2} = \sqrt{2}\epsilon.$$

Im zweiten Schritt ergibt sich somit

$$q_2 = \frac{z_1}{|z_1|} = \frac{1}{\sqrt{2}\epsilon} \begin{pmatrix} -\epsilon \\ \epsilon \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}.$$

Die dritte Spalte von Q wird berechnet als

$$q_3 = \frac{a_3 - (q_1 \cdot a_3)q_1 - (q_2 \cdot a_3)q_2}{|a_3 - (q_1 \cdot a_3)q_1 - (q_2 \cdot a_3)q_2|} = \frac{z_2}{|z_2|}.$$

Dies ergibt

$$z_2 = \begin{pmatrix} 1 \\ 1 \\ 1 + \epsilon \end{pmatrix} - \underbrace{\frac{1}{\sqrt{3+2\epsilon}}(3+2\epsilon)}_{=1} \underbrace{\frac{1}{\sqrt{3+2\epsilon}} \begin{pmatrix} 1 + \epsilon \\ 1 \\ 1 \end{pmatrix}}_{=0} - \frac{1}{\sqrt{2}} \begin{pmatrix} -1 + 1 + 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \epsilon \\ 0 \\ -\epsilon \end{pmatrix}$$

und

$$|z_2| = \sqrt{\epsilon^2 + 0^2 + (-\epsilon)^2} = \sqrt{2}\epsilon.$$

Wiederum im zweiten Schritt ergibt sich

$$q_3 = \frac{z_2}{|z_2|} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

Nach diesem Vorgehen kommt man also auf

$$Q = \begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix} = \begin{pmatrix} \frac{1+\epsilon}{\sqrt{3+2\epsilon}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3+2\epsilon}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{3+2\epsilon}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}. \quad (22.4)$$

Q sollte nach dieser Konstruktion orthogonal sein. Die Spalten müssen also einen Betrag von jeweils 1 aufweisen. Dies stimmt immer, da ja im letzten Schritt eine Normierung durchgeführt

wird. Betrachtet man unter Vernachlässigung von ϵ die Winkel zwischen den Spalten (angegeben im jeweiligen Index), kommt man auf

$$\theta_{12} = \frac{\pi}{2} = 90^\circ, \quad \theta_{13} = \frac{\pi}{2} = 90^\circ, \quad \theta_{23} = \frac{2\pi}{3} = 120^\circ.$$

Q ist also nicht orthogonal und somit ist auch $Q^{-1} \neq Q^T$ was ja wie im Least-Squares-Beispiel im Abschnitt 22.1.1 eine Eigenschaft ist, welche zwingend nötig ist.

Woher die 120° kommen ist einfacher zu verstehen, wenn man die Abbildung 22.1, welche die ganze Orthonormalisierung visualisiert, betrachtet. In 22.1(a) ist ersichtlich, wie die drei Spalten

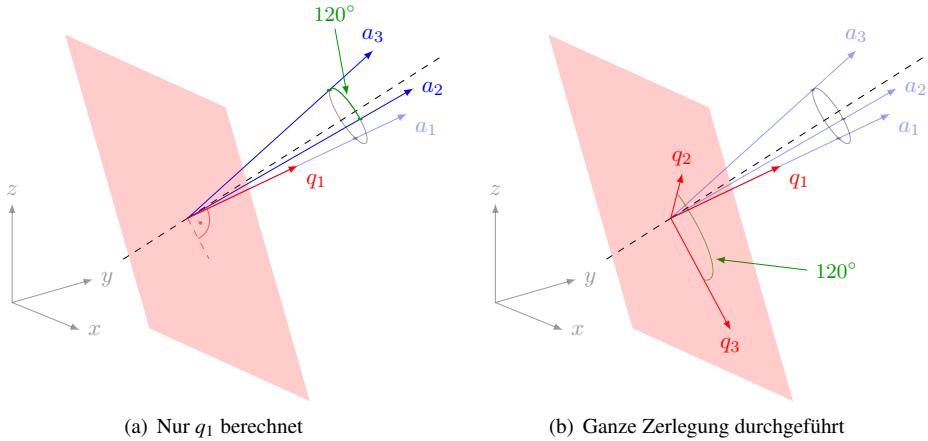


Abbildung 22.1: Visualisierung von A

von A jeweils einen Winkel von 120° bezüglich der gestrichelten Symmetrielinie einschliessen. Von den beiden Vektoren a_2 und a_3 wird jeweils das Skalarprodukt mit q_1 abgezogen, danach folgt die Normierung, womit man bei 22.1(b) ist. Der Winkel θ_{23} wird in diesem Prozess nie verändert.

22.3 Lösung mit Givens-Rotationen

Eine Drehung in der von x_i und x_j aufgespannten Ebene um den Winkel α kann mit der Matrix

$$D_{\alpha,i,j} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \quad (22.5)$$

beschrieben werden, wobei $c = \cos(\alpha)$ und $s = \sin(\alpha)$. Drehungen dieser Art heißen Givens-Rotationen.

Es ist nun möglich, Elemente unterhalb der Diagonalen einer $l \times n$ -Matrix A mit geeigneten Givens-Rotationen auf Null zu bringen, um so eine obere Dreiecksmatrix R zu erhalten. Das Produkt aller inversen Drehmatrizen ist dann eine orthogonale Matrix Q , da alle Drehmatrizen $D_{\alpha_k, i, j}$ orthogonal sind.

Das wird nun an dieser Stelle im Detail betrachtet. Zuerst wird mit Givens-Rotationen ($D_{\alpha_k, i, j}$ ist von der Grösse $l \times l$) die Matrix R berechnet:

$$R = D_{\alpha_k, l, n-1} \dots D_{\alpha_2, 3, 1} D_{\alpha_1, 2, 1} A.$$

Die erste Drehmatrix soll bewirken, dass das Element in der zweiten Zeile und ersten Spalte nach der Multiplikation

$$D_{\alpha_1, 2, 1} A = \begin{pmatrix} c_1 & -s_1 & \cdots & 0 \\ s_1 & c_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \color{red}{a_{11}} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \color{red}{a_{l1}} & a_{l2} & \cdots & a_{ln} \end{pmatrix}$$

verschwindet. Interessant sind dabei nur die rot eingefärbte Zeile und Spalte, deren Multiplikation Null ergeben muss:

$$s_1 a_{11} + c_1 a_{21} = 0,$$

und da $s_1 = \sin(\alpha_1)$ und $c_1 = \cos(\alpha_1)$, kennt man noch die Nebenbedingung

$$s_1^2 + c_1^2 = 1.$$

Dieses quadratische Gleichungssystem mit zwei Unbekannten und zwei Gleichungen kann somit nach s_1 und c_1 aufgelöst werden, wobei es wegen der quadratischen Nebenbedingung immer zwei Lösungen gibt. Welche genommen wird spielt aber keine Rolle, da nur die Auswirkung der Drehung und nicht der eigentliche Drehwinkel relevant ist.

Die Matrix $D_{\alpha_1, 2, 1}$ ist so vollständig bestimmt, ohne dass der Winkel α_1 explizit bestimmt werden musste. Dies kann so für alle weiteren Drehmatrizen $D_{\alpha_2, 3, 1}$ bis $D_{\alpha_k, l, n-1}$ durchgeführt werden.

Auf die Matrix Q kommt man dann mit

$$\begin{aligned} Q &= AR^{-1} \\ &= A(D_{\alpha_k, l, n-1} \dots D_{\alpha_2, 3, 1} D_{\alpha_1, 2, 1} A)^{-1} \\ &= \underbrace{AA^{-1}}_E D_{\alpha_1, 2, 1}^{-1} D_{\alpha_2, 3, 1}^{-1} \dots D_{\alpha_k, l, n-1}^{-1} = D_{\alpha_1, 2, 1}^T D_{\alpha_2, 3, 1}^T \dots D_{\alpha_k, l, n-1}^T, \end{aligned}$$

wobei im letzten Schritt ausgenutzt wurde, dass Drehmatrizen orthogonal sind.

Die Matrizen Q und R sind beide von der Grösse $l \times l$. Für den Spezialfall $l = n$ können sie so stehen gelassen werden. Für alle Fälle aber wo $l > n$ müssen die Dimensionen noch angepasst werden, um den Forderungen der QR-Zerlegung zu genügen. In der Matrix R stehen ab der n -ten Zeile nur noch Nullen. Diese Zeilen können somit weggelassen werden und folglich alle Einträge ab der n -ten Spalte von Q . Als Matrizen ausgeschrieben:

$$A = QR = \begin{pmatrix} q_{11} & \cdots & q_{1n} & q_{1(n+1)} & \cdots & q_{1l} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ q_{l1} & \cdots & q_{ln} & q_{l(n+1)} & \cdots & q_{ll} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$= \begin{pmatrix} q_{11} & \cdots & q_{1n} \\ \vdots & \ddots & \vdots \\ q_{l1} & \cdots & q_{ln} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{pmatrix}.$$

Q ist daraufhin also eine $l \times n$ -Matrix und R ist von der Grösse $n \times n$, wie verlangt.

22.4 Folgerungen

Um einen Vergleich anzustellen, wurden die beiden betrachteten Varianten wie beschrieben in einem Python-Skript implementiert. Der Datentyp der Matrixelemente ist dabei eine 32-Bit Floatingpoint-Zahl. Beiden Implementationen wurde dann die Matrix (welche schon vorher benutzt wurde)

$$A(\epsilon) = \begin{pmatrix} 1 + \epsilon & 1 & 1 \\ 1 & 1 + \epsilon & 1 \\ 1 & 1 & 1 + \epsilon \end{pmatrix}$$

übergeben, allerdings mit ϵ als Laufvariable zwischen 0 und 0.004 mit Schrittweite 10^{-5} .

In Abbildung 22.2 ist der Winkel von q_2 und q_3 resultierend aus beiden Implementationen geplotted. Klar ersichtlich ist, wie stabil die Implementation mit den Givens-Rotationen immer einen

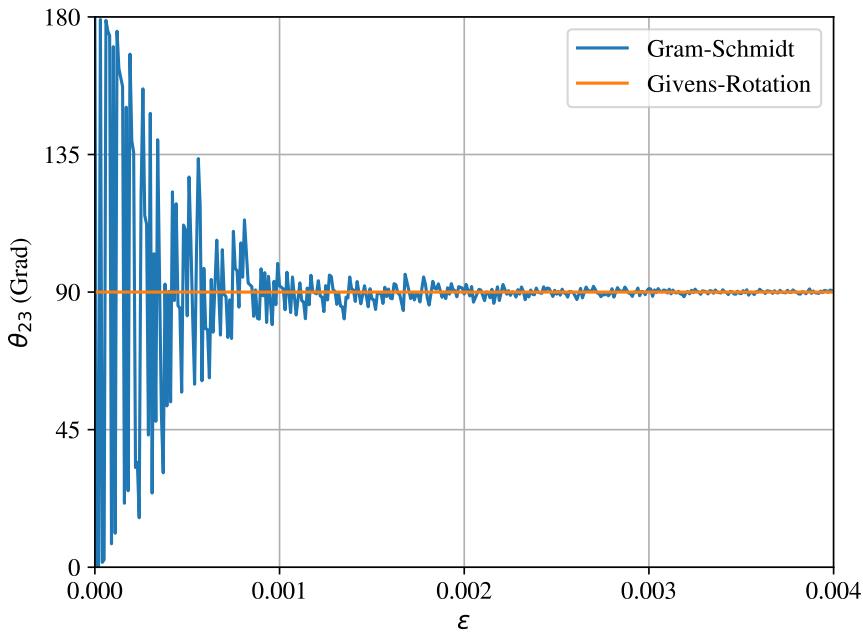


Abbildung 22.2: Winkel zwischen q_2 und q_3 .

Winkel von 90° liefert, die Implementation mit dem Gram-Schmidt-Orthonormalisierungsverfahren dagegen sehr instabil wirkt.

Die QR -Zerlegung über das Gram-Schmidt-Orthonormalisierungsverfahren liefert zwar eine sehr intuitive Methode, ist aber numerisch nicht stabil und sollte deshalb nicht so verwendet werden. Bessere numerische Eigenschaften bringt eine Implementation mit Givens-Rotationen mit sich.

Literatur

- [1] Tin-Yau Tam. *QR decomposition: History and its Applications*. 10. Dez. 2010.

23

Francis-Algorithmus

Tobias Grab

Die Berechnung der Eigenvektoren und Eigenwerte einer Matrix ist in vielen Bereichen der Wissenschaft ein oft zu lösendes Problem. Mittlerweile halten wir die Berechnung für selbstverständlich. Dank guter Hardware und vor allem guter Software ist das heutzutage meist kein Problem mehr, auch nicht von grossen Matrizen. Eine effiziente Berechnung ist in vielen Bibliotheken integriert, wie beispielsweise der MATLAB-Befehl `eig`. Vor rund 60 Jahren sah die Situation völlig anders aus. Zum einen war die zur Verfügung stehende Hardware langsam und unzuverlässig, aber vor allem konnte niemand einen zuverlässigen und effizienten Weg zur Berechnung. Dies änderte sich im Jahre 1959 als John Francis den implizit geshifteten QR-Algorithmus, heute auch bekannt unter dem Namen *Francis-Algorithmus*, entwickelt und verifiziert hat. Der Algorithmus bzw. verschiedene Versionen davon werden bis heute von diversen Bibliotheken benutzt [5].

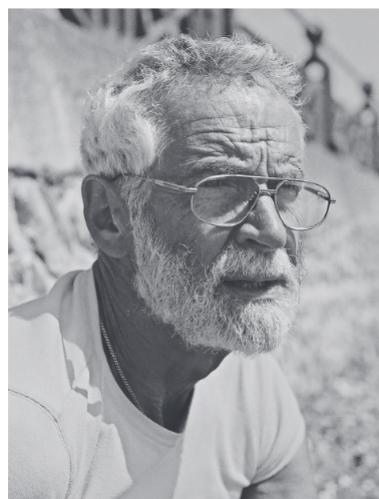


Abbildung 23.1: John Francis im Juli 2008 [1]

Wieso wird zur Berechnung der Eigenvektoren ein numerisches Verfahren benötigt? Zur Berechnung der Eigenwerte einer 3×3 Matrix kann man das charakteristische Polynom bilden und dessen Nullstellen finden, diese entsprechen den Eigenwerten der Matrix. Das grundlegende Problem ist also die Suche nach den Nullstellen eines Polynoms. Anfangs des 19. Jahrhunderts wurde von Niels Henrik Abel bewiesen, dass es keine Formel zur Berechnung der Nullstellen eines Polynoms der fünften Ordnung gibt. Folglich ist auch eine direkte Berechnung der Eigenwerte einer Matrix nicht möglich und die numerische Berechnung mit einem iterativen Verfahren der einzige Weg. Der Algorithmus fokussiert sich auf die Berechnung der Eigenwerte. Sind die Eigenwerte gefunden, können auch die dazugehörigen Eigenvektoren einfach berechnet werden.

23.1 Grundlagen

Um ein besseres Verständnis für den Algorithmus zu bekommen, wollen wir uns zuerst einige wichtige Resultate aus der linearen Algebra in Erinnerung rufen.

23.1.1 Ähnlichkeitstransformation

Wird eine Matrix von rechts mit einer Matrix C multipliziert und von links mit der Inversen C^{-1} , handelt es sich um eine Ähnlichkeitstransformation. Die Transformation ist auch bekannt als Basiswechsel bzw. Similarity Transformation

$$\hat{A} = C^{-1}AC. \quad (23.1)$$

Oft wird die Transformation auch in der Form $C\hat{A} = AC$ dargestellt, welche sich durch einfaches Umformen erreichen lässt. Es handelt sich bei einer solchen Transformation lediglich um einen Wechsel des Koordinatensystems. Daher sind einige Eigenschaften einer Matrix gegenüber einer solchen Transformation invariant. Zueinander ähnliche Matrizen haben:

1. die gleiche Determinante
2. die gleiche Spur
3. den gleichen Rang
4. das gleiche Minimalpolynom
5. die gleiche jordanische Normalform
- 6. die gleichen Eigenwerte (aber nicht notwendigerweise die gleichen Eigenvektoren)**

Eigenwerte und Eigenvektoren können gefunden werden, indem man eine Matrix durch Ähnlichkeitstransformation auf Diagonalform bringt. Der Francis-Algorithmus tut dies schrittweise unter Verwendung sehr spezieller Matrizen C .

23.1.2 Spezielle Matrizen

Im Folgenden werden wir immer wieder über einige spezielle Matrixformen sprechen. Reden wir von einer Matrix in der oberen Hessenberg-Form, meinen wir eine Matrix der Form:

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * \end{bmatrix}, \quad (23.2)$$

wobei die Sterne bedeuten, dass an dieser Stelle ein Wert in der Matrix steht, welcher nicht Null ist. Die Form entspricht also fast einer oberen Dreiecksmatrix, zusätzlich sind aber alle Elemente unmittelbar unter der Diagonalen von Null verschieden. Mit dem weiter unten im Abschnitt 23.3 vorgestellten Verfahren kann eine allgemeine Matrix in obere Hessenberg-Form gebracht werden. Wendet man das gleiche Verfahren aber auf eine symmetrische Matrix an, entsteht eine Spezialform, eine Tridiagonalmatrix:

$$\begin{bmatrix} * & * & & & \\ * & * & * & & \\ * & * & * & * & \\ * & * & * & * & * \\ * & * & * \end{bmatrix}. \quad (23.3)$$

23.1.3 Eigenwerte

Bei den Eigenvektoren einer Matrix handelt es sich um Vektoren, welche bei einer Multiplikation mit der Matrix die Richtung nicht ändern, sondern lediglich gestreckt werden.

Satz 23.1. Ein Vektor $v \in \mathbb{C}^n$ ist ein Eigenvektor von A , wenn $v \neq 0$ und Av ein Vielfaches von v ist. Der Skalar λ ist der zum Eigenvektor v gehörende Eigenwert von A

Es gilt also:

$$Av = \lambda v. \quad (23.4)$$

Gesucht sind die Nullstellen des charakteristischen Polynoms

$$\chi_A(\lambda) = \det(A - \lambda I) = 0. \quad (23.5)$$

Da ein Polynom vom Grad n höchstens n Nullstellen hat, gibt es auch höchstens n Eigenwerte.

Beispiel. Die Eigenwerte einer kleinen Matrix

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} \quad (23.6)$$

können wie folgt berechnet werden:

$$\det(A - \lambda I) = \det \left(\begin{bmatrix} 1 - \lambda & 3 & 2 \\ 0 & 2 - \lambda & 1 \\ 0 & 0 & 3 - \lambda \end{bmatrix} \right) = 0. \quad (23.7)$$

Daraus folgt:

$$(1 - \lambda)(2 - \lambda)(3 - \lambda) = 0 \quad (23.8)$$

und die Eigenwerte sind mit $\lambda_1 = 1$, $\lambda_2 = 2$ und $\lambda_3 = 3$ gefunden. \circ

Wie im Beispiel ersichtlich, sind die Eigenwerte einer Matrix in Dreiecks- oder Diagonalform durch die Diagonalelemente gegeben. Da bei Bestimmung der Determinante (bei dritten Ordnung z. B. mit der Regel von Sarrus) alles andere verschwindet.

23.1.4 Givens-Rotationen

Rotationsmatrizen sind attraktiv, da sie einfach konstruiert werden können und mit ihnen durch eine geeignete Wahl des Rotationswinkels an einer gewünschten Stelle in einem Vektor eine Null erzeugt werden kann. Dazu wird eine Rotations- bzw. Drehmatrix verwendet. Sie hat in zwei Dimensionen die folgende Form:

$$R = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix}. \quad (23.9)$$

Zur Drehung eines Punktes $P = (x, y)$ um den Winkel ϕ kann man einfach den dazugehörigen Ortsvektor mit der Rotationsmatrix multiplizieren und erhält so den Ortsvektor des gedrehten Punktes. Es handelt sich dabei um eine Givens-Rotationsmatrix, wenn $R^{-1}(\phi) = R_T(\phi) = R(-\phi)$. Durch Erweiterung der Rotationsmatrix können Drehungen natürlich auch in einem höherdimensionalem Raum verwendet werden.

Beispiel. Gegeben sei die Multiplikation einer Rotationsmatrix mit einem beliebigen Vektor:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos \phi & 0 & 0 & -\sin \phi & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \sin \phi & 0 & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \cos \phi - x_5 \sin \phi \\ x_3 \\ x_4 \\ x_2 \sin \phi + x_5 \cos \phi \\ x_6 \end{bmatrix}. \quad (23.10)$$

Wie dabei zu sehen ist, werden dabei nur die 2. und 5. Spalte des Vektors verändert. Eine Anwendung einer solchen Rotationsmatrix entspricht also einer Rotation in jener Ebene im 6-dimensionalen Raum, welche von Achse 2 und 5 aufgespannt wird. Durch eine geeignete Wahl des Rotationswinkels und ein systematisches Anwenden diverser Rotationsmatrizen kann man so einen Eintrag nach dem Andern auf Null setzen und beispielsweise eine QR-Zerlegung durchführen [3]. Weiteres dazu ist in den Kapiteln 6.4 und 22 zu finden. \circ

23.1.5 Householder-Transformation

Reflektoren (siehe Abschnitt 6.4.2) sind attraktiv, da sie einfach konstruiert werden können und mit ihnen alle Elemente eines Vektors bis auf ein Element auf Null gesetzt werden können. So kann jede beliebige Matrix relativ einfach in Hessenberg-Form gebracht werden. Ein beliebiger Vektor x und

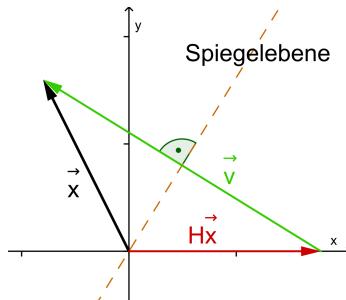


Abbildung 23.2: Illustration der Householder-Transformation in zwei Dimensionen, Abbildung von [2]

ein spezieller Vektor y sind gegeben:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{mit } y_1 \pm \|x\| \quad (23.11)$$

Da diese beiden Vektoren dieselbe Länge haben, wissen wir, dass ein Reflektor H existiert, sodass $Hx = y$. Mit dem Reflektor

$$H = I - 2vv^T, \quad \text{mit } v = (x - y)/\|x - y\| \quad (23.12)$$

können also alle Elemente eines Vektors bis auf ein Element auf null gesetzt werden. Das Ganze ist in Abbildung 23.2 in zwei Dimensionen dargestellt.

23.2 Strategie

Dieser Abschnitt dient dazu, dem Leser einen Überblick über den folgenden Algorithmus zu geben.

Das Ziel des Verfahrens ist es, die Eigenwerte einer Matrix zu bestimmen. Bei einer Dreiecksmatrix kann man diese auf der Diagonale ablesen. Ein direkte Transformation zu einer Dreiecksmatrix ist für eine allgemeine Matrix nicht möglich. Die Transformation zu einer oberen Hessenberg-Matrix, welche sich von der Dreiecksmatrix lediglich in der Subdiagonalen unterscheidet, ist aber machbar.

Folglich wird für die Bestimmung der Eigenwerte wie folgt vorgegangen:

1. Transformation in obere Hessenberg-Form.
2. Subdiagonale verkleinern.
3. Zurück zu Punkt 1, da bei Punkt 2 die Hessenberg-Form zerstört wird.

Nach genügend Iterationen sind die Subdiagonalelemente so klein, dass es sich bei der Matrix praktisch um eine Dreiecksmatrix handelt und sich die Eigenwerte auf der Diagonale befinden.

23.3 Vorbereitung der Matrix

Um den Francis-Algorithmus effizient ausführen zu können, muss die Matrix zuerst auf Hessenberg-Form gebracht werden.

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \quad (23.13)$$

Satz 23.2. Jede Matrix $A \in \mathbb{R}^{n \times n}$ ist orthogonal ähnlich zu einer Matrix H in der oberen Hessenberg-Form: $H = Q^{-1}AQ$, wobei Q ein Produkt von $n - 2$ Reflektoren ist.

Gegeben sei eine Matrix A mit einer Submatrix \hat{A} , einer ersten Spalte a_{11} und einem Vektor b .

$$A = \begin{bmatrix} a_{11} & * \\ b & \hat{A} \end{bmatrix}. \quad (23.14)$$

Ziel ist es, alle Einträge dieses Vektors bis auf den Ersten auf Null zu setzen. Dies kann mit einer Multiplikation mit einer geeigneten Matrix

$$Q_1 = \begin{bmatrix} 1 & * \\ 0 & \hat{Q}_1 \end{bmatrix} \quad (23.15)$$

gemacht werden. \hat{Q}_1 muss so gewählt werden, dass bei einer Multiplikation mit b alle Werte bis auf den Ersten verschwinden. Zusätzlich soll die Länge des neuen Vektors $\hat{Q}_1 b$, bzw. dieser einzelne Eintrag, der Länge des Vektors b entsprechen:

$$\hat{Q}_1 b = \begin{bmatrix} -y \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (23.16)$$

Wird \hat{Q}_1 so gewählt, entsteht bei einer Multiplikation von links mit der Matrix Q_1 der gewünschte Vektor in der ersten Spalte.

$$A_{\text{Half}} = Q_1 A = \begin{bmatrix} a_{11} & * & \dots & * \\ -y & & & \\ 0 & & & \\ \vdots & & \hat{Q}_1 \hat{A} & \\ 0 & & & \end{bmatrix} \quad (23.17)$$

\hat{Q}_1 kann dabei zum Beispiel mit Givens Rotationsmatrizen 23.1.4 oder einer Householder-Transformationsmatrix 23.1.5 gebildet werden.

Damit wir die Eigenwerte unserer Matrix nicht verändern, müssen wir die Matrix Q_1 nun auch von der rechten Seite hinzu multiplizieren. Da die erste Spalte von Q_1 nur eine Eins und sonst Nullen

hat, wird dabei die vorher erzeugte Spalte nicht beeinflusst. Das Resultat ist also eine ähnliche Matrix

$$A_1 = A_{\text{Half}} Q_1 = \begin{bmatrix} a_{11} & * & \dots & * \\ -y & 0 & & \\ 0 & & & \\ \vdots & & \hat{Q}_1 \hat{A} \hat{Q}_1^{-1} & \\ 0 & & & \end{bmatrix} = \begin{bmatrix} a_{11} & * & \dots & * \\ -y & 0 & & \\ 0 & & & \\ \vdots & & \hat{A}_{-1} & \\ 0 & & & \end{bmatrix}, \quad (23.18)$$

welche an den gewünschten Stellen in der ersten Spalte Nullen hat.

Das gleiche Prinzip wird jetzt auf die Submatrix angewendet. Durch den zweiten Reflektor entstehen Nullen in der zweiten Spalte:

$$Q_2 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & \hat{Q}_2 & & \\ 0 & 0 & & & \end{bmatrix}, \quad A_2 = \begin{bmatrix} a_{11} & * & \dots & * \\ -y_1 & * & \dots & * \\ 0 & -y_2 & & \\ \vdots & \vdots & \hat{Q}_2 \hat{A}_2 \hat{Q}_2^{-1} & \\ 0 & 0 & & \end{bmatrix}.$$

Mit einem weiteren Schritt ebenfalls in der dritten Spalte usw. Das Resultat ist eine Matrix in der Hessenberg-Form, welche ähnlich zu A ist. Das heisst für uns, sie besitzt dieselben Eigenwerte wie die Matrix A . Die ganzen Multiplikation der Reflektoren kann zusammengefasst werden, sodass sich eine einzige Ähnlichkeitstransformation

$$H = Q_{n-2} Q_{n-1} \dots Q_1 A Q_1^{-1} \dots Q_{n-1}^{-1} Q_{n-2}^{-1} \quad (23.19)$$

ergibt. Mit

$$Q = Q_1 \dots Q_{n-1} Q_{n-2} \quad (23.20)$$

erhalten wir

$$Q^{-1} = Q_{n-2}^{-1} Q_{n-1}^{-1} \dots Q_1^{-1} \quad (23.21)$$

und schlussendlich

$$H = Q^{-1} A Q. \quad (23.22)$$

Im Falle einer symmetrischen Matrix A werden durch die Multiplikation von rechts dieselben Elemente im oberen Teil der Matrix eliminiert und das Ergebnis der Transformation ist tridiagonal. Durch die Reduzierung der ursprünglichen Matrix auf Hessenberg-Form, werden die Kosten des anschliessenden Algorithmus stark reduziert.

23.4 Francis-Iteration erster Ordnung

Nun wollen wir uns mit dem eigentlichen Francis-Algorithmus [6] befassen. Es handelt sich dabei um ein iteratives Verfahren und lässt sich daher am besten durch die Analyse einer Iteration beschreiben. Die Ausgangssituation für den Algorithmus ist eine Matrix, welche sich in oberer Hessenberg-Form oder sogar tridiagonaler Form befindet.

Der Algorithmus beginnt mit der Wahl eines Shifts ρ_1 und der Berechnung der ersten Spalte von $A - \rho_1 I$. Diese Spalte hat die Form

$$p = \begin{bmatrix} a_{11} - \rho_1 \\ a_{21} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (23.23)$$

Anschliessend muss eine Matrix Q_0 gefunden werden, welche auf den Achsen 1 und 2 arbeitet und den Wert von a_{21} eliminiert. Diese Matrix kann zum Beispiel mit Givens-Rotationsmatrizen 23.1.4 oder einer Householder-Transformationsmatrix 23.1.5 gebildet werden.

$$Q_0^{-1} p = \begin{bmatrix} * \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (23.24)$$

Danach muss A mit Q_0 zu $Q_0^{-1} A Q_0$ transformiert werden. Die Transformation von A nach $Q_0^{-1} A$ kombiniert lediglich die Zeilen 1 und 2, die Matrix behält aber ihre Hessenberg-Form. Die anschliessende Transformation von $Q_0^{-1} A$ nach $Q_0^{-1} A Q_0$ kombiniert die Spalten 1 und 2, wodurch die Hessenberg-Form zerstört wird. Es entsteht an der Stelle (3,1) eine ‘‘Ausbuchtung’’. Der Rest der Iteration beschäftigt sich damit, die Matrix wieder in ihre Hessenberg-Form zu bringen. Dafür wählen wir eine Matrix, welche auf den Achsen 2 und 3 arbeitet und den Wert von a_{31} eliminiert. So wird die Ausbuchtung nach unten verschoben. Dieses Prozedere wird wiederholt, bis die Ausbuchtung schlussendlich unten aus der Matrix hinausgeschoben wird. Die Hessenberg-Form ist wiederhergestellt.

Bei einer 5×5 Matrix sieht eine Iteration also wie folgt aus:

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \quad (23.25)$$

Das Resultat einer Francis-Iteration ist also

$$\hat{A} = Q_{n-2} \dots Q_1 Q_0 A Q_0^{-1} Q_1^{-1} \dots Q_{n-2}^{-1}, \quad (23.26)$$

wobei Q_0 die Transformation bezeichnet, welche eine Ausbuchtung in der Matrix erzeugt und Q_1, \dots, Q_{n-2} die Transformationen sind, welche die Hessenberg-Form wiederherstellen. Dabei handelt es sich also nur um Ähnlichkeitstransformationen und \hat{A} besitzt dementsprechend die gleichen Eigenwerte wie A . Mit jeder Iteration konvergieren die Elemente unter der Diagonalen weiter gegen Null und die Eigenwerte erscheinen dementsprechend auf der Diagonalen von \hat{A} .

23.4.1 Wahl der Shifts

Die Shifts werden zur Konvergenzbeschleunigung gebraucht. Die Wahl eines guten Shifts kann den Algorithmus drastisch beschleunigen. Ein guter Shift ist dabei einer, welcher einen Eigenwert der Matrix gut approximiert.

Unser Algorithmus verkleinert bei jeder Iteration die Elemente auf der Subdiagonalen. Die Diagonalelemente werden daher eine immer bessere Approximationen der Eigenwerte. Daher wird oft ein Eigenwert auf der Diagonalen als Shift für die nächste Iteration verwendet. Zu Beginn mag dies noch eine schlechte Approximation sein, sie verbessert sich aber sehr schnell.

Wird das unterste Diagonalelement verwendet spricht man von einem Rayleigh-Quotienten-Shift. Eine andere, ähnliche Shift-Strategie ist der nach James Hardy Wilkinson benannte Wilkinson-Shift. Für diesen wird der näher am letzten Matrixelement liegende Eigenwert der untersten „ 2×2 Matrix“ als Shift benutzt.

Beispiel. Gegeben sei eine Matrix A , welche sich bereits in Hessenberg Form befindet

$$A = \begin{bmatrix} 17 & -28.941 & 1.847 & -4.46 & 2.257 \\ -27.677 & 33.94 & 26.188 & -2.228 & 1.268 \\ 0 & 25.096 & 20.687 & -6.606 & -0.197 \\ 0 & 0 & -5.963 & \textcolor{orange}{-16.816} & \textcolor{orange}{-12.445} \\ 0 & 0 & 0 & \textcolor{orange}{-9.01} & \textcolor{red}{10.189} \end{bmatrix} \quad \text{mit } \text{eig}(A) = \begin{bmatrix} 65 \\ -21.277 \\ -13.126 \\ 21.277 \\ 13.126 \end{bmatrix} \quad (23.27)$$

Für die Rayleigh-Quotienten-Shift Strategie würde man also $\textcolor{red}{10.189}$ als ersten Shift verwenden. Verfolgt man die Wilkinson-Shift Strategie, würde man die Eigenwerte der Matrix

$$\begin{bmatrix} \textcolor{orange}{-16.816} & \textcolor{orange}{-12.445} \\ \textcolor{orange}{-9.01} & \textcolor{red}{10.189} \end{bmatrix} \quad (23.28)$$

berechnen. Dabei findet man -21.279 und 13.126 . Da 13.126 näher als -21.279 am untersten Diagonalelement liegt, wird dies als erster Shift verwendet. Dabei handelt es sich schon um eine gute Annäherung für einen Eigenwert der Matrix A . \circ

23.5 Francis-Iteration der Ordnung n

Im Prinzip kann der Francis-Algorithmus für eine beliebige Ordnung n durchgeführt werden, praktisch sollte die Ordnung n aber klein gewählt werden. Francis hat die Ordnung $n = 2$ gewählt. Gegenüber der Ordnung $n = 1$ entsteht dabei der Vorteil, dass komplexe konjugierte Shifts gewählt werden können (Und somit auch komplexe konjugierte Eigenwerte approximiert) ohne dass komplexe gerechnet werden muss.

Berechnung einer Francis Iteration:

1. Wähle n verschiedene Shifts ρ_1, \dots, ρ_n .
2. Berechne $p = (A - \rho_n I) \cdots (A - \rho_1 I)e_1$.
3. Berechne den Reflektor Q_0 , welcher die erste Spalte wie gewünscht verändert.
4. Führe eine Ähnlichkeitstransformation mit dem berechneten Shift $Q_0^{-1}AQ_0$, was eine Ausbuchtung in der Matrix erzeugt.
5. Führe weitere Ähnlichkeitstransformationen durch, wobei sich die Ausbuchtung immer weiter nach unten verschiebt und sich schlussendlich wieder eine Hessenberg-Matrix ergibt.

Wichtig anzumerken ist, dass wir nach der Wahl von n Shifts nicht die gesamte Matrix $(A - \rho_n I) \cdots (A - \rho_1 I)$ berechnen, denn die Kosten dafür wären zu hoch. Es reicht die erste Spalte zu berechnen, welche bei einer vernünftigen Wahl von n einfach zu berechnen ist.

Der Francis-Algorithmus der 1. Ordnung kann auf allgemeine komplexe Matrizen angewendet werden, wobei dann die ganze Arithmetik ebenfalls komplex ist. Sind die Matrix und die Shifts aber reell, so kann die ganze Iteration in reeller Arithmetik ausgeführt werden. Beim Francis-Algorithmus der 2. Ordnung können sogar konjugiert komplexe Shifts verwendet werden, aber die ganze Berechnung kann in reeller Arithmetik ausgeführt werden. Grund dafür ist, dass beim Berechnen der ersten Spalte (Punkt 2) mit konjugiert komplexen Shifts

$$p = (A - \rho I)(A - \bar{\rho}I)e_1 \quad (23.29)$$

$$p = (A^2 - (\rho + \bar{\rho})A + \rho\bar{\rho}I)e_1 \quad (23.30)$$

der Imaginäranteil verschwindet.

Die totalen Kosten für den expliziten QR Algorithmus für k Iterationen sind $8kn^3$ Additionen und $12kn^3$ Multiplikationen. Im Vergleich dazu betragen die totalen Kosten für den impliziten QR-Algorithmus (Francis) für k Iterationen $2kn^3$ Additionen und $2kn^3$ Multiplikationen, wobei ein Schritt des impliziten Verfahrens zwei Schritten des expliziten Verfahrens entspricht [4].

23.6 Nachweis

Da ein formaler Beweis zu komplex wäre, verweise ich an dieser Stelle auf [6]. Stattdessen wollen wir aber anhand einer einfachen Implementierung zeigen, dass der Algorithmus funktioniert.

23.6.1 Implementation

Folgender MATLAB-Code von [6] ist eine Schritt-für-Schritt Implementation einer Francis-Iteration für eine symmetrische $n \times n$ Matrix. Zu Beginn muss eine Matrix generiert und auf Hessenberg-Form gebracht werden.

```

1 | rng(2)
2 | n=6;
3 | a=randn(n);
4 | a=a+a';
5 | a=hess(a);
6 | [Evec,Eval]=eig(a);
```

Anschliessend soll der Wilkinson-Shift berechnet werden. Dies geschieht im folgenden Code-Abschnitt.

```

1 | Eval;
2 | %%
3 | htr= .5*(a(n-1,n-1) + a(n,n));
4 | dscr= sqrt((.5*(a(n-1,n-1)-a(n,n)))^2 + a(n,n-1)^2);
5 | if htr < 0, dscr= -dscr; end
6 | root1 = htr + dscr;
7 | if root1 == 0
8 |     root2 = 0;
9 | else
10 |     det= a(n-1,n-1)*a(n,n) -a(n,n-1)^2;
11 |     root2= det/root1;
12 | end
13 | if abs(a(n,n)-root1) < abs(a(n,n)-root2)
```

```

14     shift= root1;
15   else      shift= root2;
16   end

```

Nun muss ein Rotator Q_0 gebildet werden. Die anschliessende Transformation (Multiplikation von rechts und links) erzeugt eine Ausbuchtung in der Matrix.

```

1  cs= a(1,1) -shift; sn=a(2,1);
2  r= norm([cs sn]);
3  cs= cs/r; sn=sn/r;
4  q0= [ cs -sn; sn cs];
5  a(:,1:2) = a(:,1:2) * q0;
6  a(1:2,:) = q0'*a(1:2,:);

```

Um die Francis-Iteration zu beenden, muss nun die Ausbuchtung aus der Matrix hinausgeschoben werden. Dies wird mit dem folgenden Loop erledigt.

```

1  for ii = 1:n-2
2    %Chase the bulge from position (ii+2, ii)
3    cs=a(ii+1,ii);sn=a(ii+2,ii);
4    r=norm([cs sn]);
5    cs= cs/r; sn= sn/r;
6    a(ii+1,ii) = r;
7    a(ii+2,ii)=0;
8    qj= [cs -sn; sn cs];
9
10   %Gives the rotator to chase the bulge
11   a(ii+1:ii+2,ii+1:n) =qj'*a(ii+1:ii+2,ii+1:n);
12   a(:,ii+1:ii+2) = a(:,ii+1:ii+2)*qj;
13 end

```

Zum Schluss kann man die wichtigsten Matrixelemente ausgeben.

```

1  format short e
2  subdiag = diag(a,-1);
3  format long
4  bottom_entry = a(n,n);

```

23.6.2 Resultate

Nun wollen wir den in 23.6.1 beschriebenen Code testen. Zuerst eine Bemerkung zur Darstellung der Matrizen. Kleine Zahlen, welche nicht null sind, werden im folgenden als 0.0000 oder -0.0000 geschrieben. Handelt es sich bei einer Zahl um eine exakte Null, so wird sie nicht geschrieben. Mit Samen¹ für den Zufallszahlengenerator ergibt sich folgende zufällige Matrix a :

$$\begin{bmatrix} -3.2289 & 1.6892 & & \\ 1.6892 & -1.6190 & 1.1276 & \\ & 1.1276 & 0.9950 & -2.6434 \\ & & -2.6434 & 0.7690 & -4.4160 \\ & & & -4.4160 & -3.7302 \end{bmatrix}. \quad (23.31)$$

¹Wird ein Samen für einen Zufallsgenerator definiert, produziert dieser Zufallsgenerator immer die gleiche Sequenz von Zahlen, welche von der definierten Verteilungsfunktion gesampelt wurden. So kann mit zufälligen Zahlen gerechnet werden und das Resultat ist dennoch reproduzierbar. In Matlab kann das durch Einfügen von `rng(1)` vor dem Zufallsgenerator gemacht werden.

Führt man nun mehrere Francis Iterationen gemäss obiger Implementation durch, ergibt sich nach 5 Iterationen die Matrix:

$$\begin{bmatrix} 1.9741 & 2.4170 & 0.0000 & 0.0000 & 0.0000 \\ 2.4170 & 2.6240 & 1.2488 & 0.0000 & 0.0000 \\ & 1.2488 & -0.3743 & 0.3716 & -0.0000 \\ & & 0.3716 & -4.3253 & -0.0000 \\ & & & -0.0000 & -6.7126 \end{bmatrix}. \quad (23.32)$$

Nach 10 Iterationen ergibt sich:

$$\begin{bmatrix} 4.8651 & 0.4449 & 0.0000 & 0.0000 & 0.0000 \\ 0.4449 & 0.4722 & 0.3417 & 0.0000 & -0.0000 \\ & 0.3417 & -1.0764 & 0.0040 & -0.0000 \\ & & 0.0040 & -4.3623 & -0.0000 \\ & & & 0.0000 & -6.7126 \end{bmatrix}. \quad (23.33)$$

Nach 50 Iterationen ergibt sich:

$$\begin{bmatrix} \textcolor{red}{4.9099} & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & \textcolor{red}{0.5008} & 0.0000 & 0.0000 & -0.0000 \\ & 0.0000 & \textcolor{red}{-1.1499} & 0.0000 & -0.0000 \\ & & 0.0000 & \textcolor{red}{-4.3623} & -0.0000 \\ & & & 0.0000 & \textcolor{red}{-6.7126} \end{bmatrix}. \quad (23.34)$$

Es ist ersichtlich, dass die Subdiagonalelemente von Iteration zu Iteration kleiner werden, bis sie schliesslich annähernd 0 sind und eine Dreiecksmatrix entstanden ist. Da wir zur Manipulation unserer Matrix lediglich Ähnlichkeitstransformationen verwendet haben, entsprechen die Diagonalelemente dieser Dreiecksmatrix den Eigenwerten unserer ursprünglichen Matrix.

Literatur

- [1] G. Golub und F. Uhlig. “The QR algorithm: 50 years later its genesis by John Francis and Vera Kublanovskaya and subsequent developments”. In: *IMA Journal of Numerical Analysis* 29.3 (2009), S. 467–485.
- [2] *Houdeholdertransformation*. 20. Juli 2020. URL: <https://de.wikipedia.org/wiki/Householdertransformation>.
- [3] *QR-Zerlegung mit Roatationsmatrizen*. 21. Dez. 2019. URL: <https://www.youtube.com/watch?v=JjRUKYwEYBQ>.
- [4] *The QR Algorithm*. 20. Juli 2020. URL: <https://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf>.
- [5] David S. Watkins. “Francis’ Algorithm”. In: *The American Mathematical Monthly* 118.5 (2011), S. 387–403. ISSN: 00029890, 19300972. URL: <http://www.jstor.org/stable/10.4169/amer.math.monthly.118.05.387>.
- [6] David S. Watkins. *Fundamentals of Matrix Computations*. USA: John Wiley und Sons, Inc., 1991. ISBN: 0471614149.

24

Die Methode der konjugierten Gradienten

Raphael Unterer

Die Methode der konjugierten Gradienten (kurz CG) bezeichnet einen schnellen Algorithmus zum Lösen grosser linearer Gleichungssysteme. Grosse lineare Gleichungssysteme zu lösen ist ein klassisches numerisches Problem. Diese treten in diversen Anwendungen auf, unter anderem beim Lösen von partiellen Differentialgleichungen.

Im Kapitel 6 wurden bereits einige numerische Algorithmen vorgestellt um lineare Gleichungssysteme approximativ zu lösen. Die Methode der konjugierten Gradienten löst im Gegensatz dazu ein lineares Gleichungssystem nicht approximativ sondern exakt. Dazu benötigt CG genau N Schritte, wobei N die Anzahl Gleichungen und Unbekannten bezeichnet.

Im Rahmen dieses Kapitels wird dieser CG-Algorithmus hergeleitet und einige Untersuchungen werden vorgenommen.

24.1 Voraussetzungen

Der CG-Algorithmus versucht ein (grosses) lineares Gleichungssystem der Form

$$Ax = b \quad x, b \in \mathbb{R}^N \tag{24.1}$$

zu lösen. Wir definieren folgende Voraussetzungen für die $N \times N$ Matrix A :

- A ist symmetrisch, d. h. $x^T A y = y^T A^T x = y^T A x$
- A ist positiv definit, d. h. $x^T A x > 0$

Diese Voraussetzungen sind die Bedingung, damit das CG-Verfahren erfolgreich funktionieren kann. Um eine schnelle Konvergenz zu erreichen, sind zudem gut konditionierte Matrizen von Vorteil. Dies sind zum Beispiel dünn besetzte Matrizen mit wenigen Einträgen abseits der Diagonalen, wie sie häufig bei der Lösung von partiellen Differentialgleichungen vorkommen.

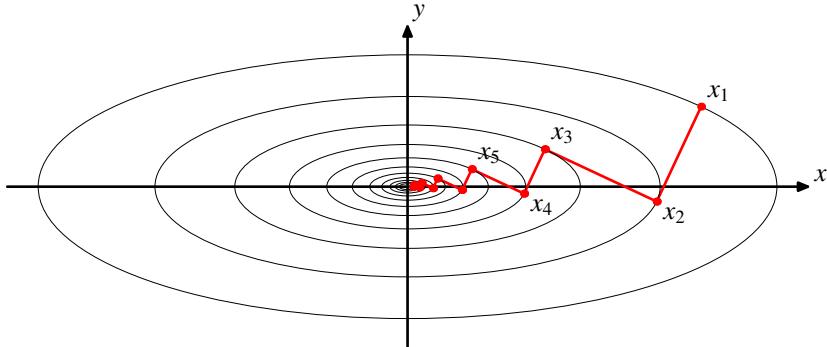


Abbildung 24.1: Gradient Descent für ellipsenförmige Niveaulinien (schlechte Konditionierung) in zwei Dimensionen. Abbildung aus dem Seminarbuch von 2014 [2].

24.2 Gradient Descent

Um den CG-Algorithmus zu verstehen, ist es hilfreich zuerst die Gradient Descent Methode zu analysieren. Gradient Descent ist eine bekannte Methode um iterativ ein Minimierungsproblem zu lösen. Dabei wird immer eine gewisse Schrittweite weit entlang des Gradienten des Minimierungsproblems abgestiegen. Eine Darstellung von einem zweidimensionalen Minimierungsproblem mit Gradient Descent findet sich in Abbildung 24.1.

24.2.1 Minimierungsproblem

Eine Lösung für x kann durch Minimieren von

$$\Phi(x) = \frac{1}{2}x^T Ax - x^T b \quad (24.2)$$

gefunden werden. Der folgende Beweis zeigt, wieso dies ein sinnvoller Ansatz ist.

Beweis. Wir definieren eine zweite Variable $z = x + \lambda y$, was uns erlaubt die folgende Differenz auszurechnen

$$\Phi(z) - \Phi(x) = \frac{1}{2}(x + \lambda y)^T A(x + \lambda y) - (x + \lambda y)^T b - \frac{1}{2}x^T Ax + x^T b \quad (24.3)$$

$$= \frac{1}{2}(x^T Ax + x^T A\lambda y + \lambda y^T Ax + \lambda y^T A\lambda y) - x^T b - \lambda y^T b - \frac{1}{2}x^T Ax + x^T b. \quad (24.4)$$

Da A symmetrisch ist, können die Terme $x^T A\lambda y$ und $\lambda y^T Ax$ zusammengefasst werden (analog zur binomischen Formel)

$$\Phi(z) - \Phi(x) = \frac{1}{2}\cancel{x^T Ax} + x^T A\lambda y + \frac{1}{2}\lambda y^T A\lambda y - \cancel{x^T b} - \lambda y^T b - \frac{1}{2}\cancel{x^T Ax} + \cancel{x^T b} \quad (24.5)$$

$$= \lambda x^T Ay + \frac{1}{2}\lambda^2 y^T Ay - \lambda y^T b \quad (24.6)$$

$$= \frac{\lambda^2}{2}y^T Ay + \lambda y^T (Ax - b). \quad (24.7)$$

Nun können wir den Beweis führen, indem wir $\Phi(z) \geq \Phi(x)$ setzen (da $\Phi(x)$ ja minimiert wird)

$$\Phi(z) \geq \Phi(x) \quad (24.8)$$

$$\Phi(z) = \frac{\lambda^2}{2} y^T A y + \lambda y^T (Ax - b) + \Phi(x) \quad (24.9)$$

$$0 \leq \frac{\lambda^2}{2} y^T A y + \lambda y^T (Ax - b) \quad \forall \quad y \in \mathbb{R}^N. \quad (24.10)$$

Der erste Term ist dabei quadratisch in λ , A ist positiv definit und somit ist immer $\frac{\lambda^2}{2} y^T A y \geq 0$. Beim zweiten Term ist diese Bedingung nur erfüllt, wenn $Ax - b = 0$ für alle y . Damit ist bewiesen, dass dann eine Lösung für die Gleichung $Ax = b$ durch Minimierung von $\Phi(x)$ gefunden wird. \square

24.2.2 Berechnung der Schrittweite

Falls eine Suchrichtung gegeben ist, kann die optimale Schrittweite bestimmt werden, um $\Phi(x)$ minimal werden zu lassen. Als Suchrichtung bezeichnen wir die Richtung, in welcher das aktuelle x verbessert werden soll. Wir suchen also ausgehend vom aktuellen x in dieser Richtung nach einem minimalen $\Phi(x)$. Gegeben:

- Aktueller Index k
- Suchrichtung d_k
- Startpunkt x_k

Wir suchen nun die optimale Schrittweite α , um möglichst nahe an die Lösung zu kommen in der gegebenen Suchrichtung. Dazu stellen wir wieder ein Minimierungsproblem auf

$$\Phi(x_k + \alpha d_k) = \frac{1}{2} x_k^T A x_k + \alpha x_k^T A d_k + \frac{1}{2} \alpha^2 d_k^T A d_k - x_k^T b - \alpha d_k^T b \quad (24.11)$$

$$= \frac{1}{2} \alpha^2 d_k^T A d_k + \alpha (x_k^T A d_k - d_k^T b) + \frac{1}{2} x_k^T A x_k - x_k^T b. \quad (24.12)$$

Durch die Symmetrie von A gilt $x_k^T A d_k = d_k^T A x_k$ und man kann das Minimierungsproblem weiter vereinfachen zu

$$\Phi(x_k + \alpha d_k) = \frac{1}{2} \alpha^2 d_k^T A d_k + \alpha d_k^T (A x_k - b) + \frac{1}{2} x_k^T (A x_k - b). \quad (24.13)$$

Hier haben wir eine quadratische Gleichung in α , welche einer nach oben geöffneten Parabel entspricht (da $d_k^T A d_k \geq 0$). Somit ist es möglich, ein eindeutiges Minimum zu finden, indem wir die Gleichung nach α ableiten und null setzen:

$$\frac{\partial \Phi(x_k + \alpha d_k)}{\partial \alpha} = \alpha d_k^T A d_k + d_k^T (A x_k - b) = 0. \quad (24.14)$$

Dies ergibt für α

$$\alpha = \frac{d_k^T (b - A x_k)}{d_k^T A d_k}. \quad (24.15)$$

Wenn wir nun den Fehler der momentanen Approximation als Residuum $r_k = b - A x_k$ bezeichnen, erhalten wir

$$\alpha = \frac{\langle d_k, r_k \rangle}{\langle d_k, d_k \rangle_A}, \quad (24.16)$$

wobei $\langle d_k, d_k \rangle_A = d_k^T A d_k$ das verallgemeinerte Skalarprodukt zu A darstellt. Somit haben wir nun die optimale Schrittänge gefunden.

Daraus lässt sich nun der nächste Punkt x_{k+1} berechnen als

$$x_{k+1} = x_k + \frac{\langle d_k, r_k \rangle}{\langle d_k, d_k \rangle_A} d_k. \quad (24.17)$$

24.2.3 Berechnung des Gradienten

Die neue Suchrichtung d_{k+1} entspricht dem negativen Gradienten von $\Phi(x_{k+1})$. Dieser Gradient lässt sich berechnen als

$$d_{k+1} = -\nabla \Phi(x_{k+1}) = -\nabla \left(\frac{1}{2} x_{k+1}^T A x_{k+1} - x_{k+1}^T b \right) = -(Ax_{k+1} - b) = b - Ax_{k+1}, \quad (24.18)$$

Wobei zur Berechnung der quadratischen Ableitung Lemma 7.18 zur Anwendung kommt.

Beobachtung 1. Die neue Abstiegsrichtung ist identisch zum neuen Residuum r_{k+1} .

24.2.4 Probleme beim Gradient Descent

Mit den hier hergeleiteten Resultaten kann eine approximative Lösung gefunden werden. Die Konvergenzgeschwindigkeit ist allerdings stark abhängig von A . Das Beispiel in Abbildung 24.1 zeigt (in zwei Dimensionen), wie Gradient Descent bei ovalen Niveaulinien zu oszillieren beginnt. Die Konvergenz ist also in diesem Fall sehr langsam, die exakte Lösung wird nie erreicht. Falls die Niveaulinien allerdings eher rund sind, erhöht sich die Konvergenzgeschwindigkeit.

Diese Probleme werden mit der Erweiterung zum CG-Algorithmus behoben.

24.2.5 Beispiel zur Motivation der A -Orthogonalität

Das folgende Beispiel soll ausgehend von Gradient Descent die Motivation liefern, weshalb im CG-Algorithmus die A -Orthogonalität wichtig ist. Unter A -Orthogonalität (\perp_A) zweier Vektoren versteht man, dass deren verallgemeinertes Skalarprodukt zu A verschwindet:

$$a \perp_A b \implies a^T A b = \langle a, b \rangle_A = 0. \quad (24.19)$$

Richtungen, welche A -orthogonal stehen, werden auch konjugiert bezüglich A genannt. Dies erklärt den Namen des CG-Verfahrens. Für das Beispiel definieren wir A , b , x_1 und berechnen x_2 :

$$A = \begin{pmatrix} 4 & 0 \\ 0 & 9 \end{pmatrix}$$

$$b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$r_1 = b - Ax_1 = -\begin{pmatrix} 4 \\ 9 \end{pmatrix}$$

$$d_1 = r_1$$

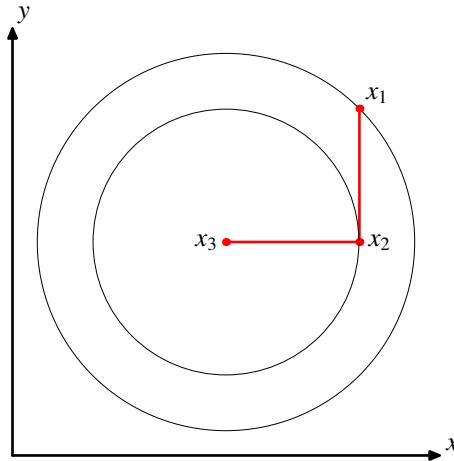


Abbildung 24.2: Zweidimensionaler Abstieg entlang der Koordinatenachsen, zwei Schritte genügen um die exakte Lösung zu finden. Abbildung aus dem Seminarbuch von 2014 [2].

$$x_2 = x_1 + \frac{\langle r_1, r_1 \rangle}{\langle r_1, r_1 \rangle_A} r_1 = \begin{pmatrix} 0.51 \\ -0.1 \end{pmatrix}. \quad (24.20)$$

(24.21)

Dieses Gleichungssystem hat den Nullpunkt als Lösung $x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Um im nächsten Schritt eine Lösung finden zu können, müsste die nächste Abstiegsrichtung dem Vektor von x_2 zum Ursprung entsprechen. Wenn wir das A -Skalarprodukt von $-x_2$ und d_1 berechnen, verschwindet dieses:

$$\langle -x_2, d_1 \rangle_A = -x_2^T A d_1 = \begin{pmatrix} 0.51 & -0.1 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 9 \end{pmatrix} \begin{pmatrix} -4 \\ -9 \end{pmatrix} = 0$$

was der A -Orthogonalität entspricht. Damit ist gezeigt, dass die optimale zweite Abstiegsrichtung A -orthogonal zur ersten Abstiegsrichtung wäre. Somit ist die A -Orthogonalität das entscheidende Konzept um dem CG-Algorithmus zu ermöglichen in N -Schritten eine exakte Lösung zu finden. Abbildung 24.4 zeigt das Resultat einer solchen optimalen Wahl der zweiten Abstiegsrichtung.

24.3 Herleitung des Algorithmus

24.3.1 Intuition

Die Idee des CG-Algorithmus ist, die Schwächen von Gradient Descent auszubessern. Dies geschieht, indem pro Dimension des Minimierungsproblems nur ein Schritt benötigt wird. Intuitiv kann man sich dies als Abstieg entlang der Koordinatenachsen vorstellen, wie in Abbildung 24.2 gezeigt. Ein solches Verfahren findet immer in N Schritten die exakte Lösung.

Auf den ersten Blick ist dies sehr vielversprechend. Allerdings ist die Konvergenz dieses Verfahrens unter Umständen sehr schlecht, da die Koordinatenachsen in hochdimensionalen Problemen weit weg vom Gradienten sind. Dabei haben viele Dimensionen keinen richtigen Einfluss und ein Abstieg in deren Richtung verbessert die Approximation nur minimal. In Abbildung 24.3 sieht man

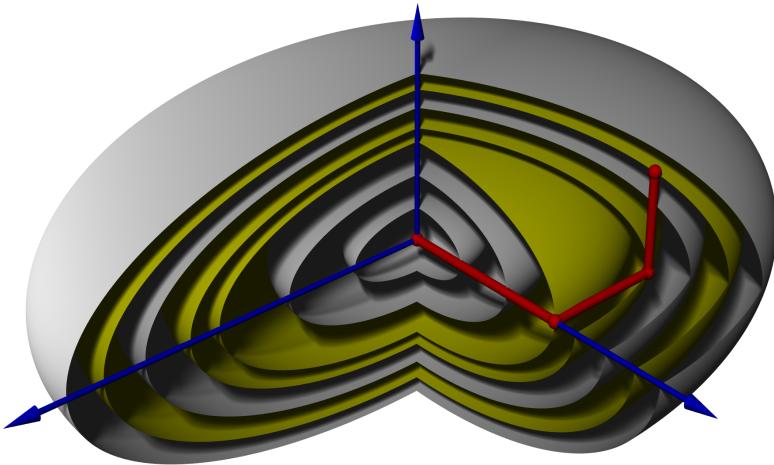


Abbildung 24.3: 3D Abstieg entlang der Koordinatenachsen, 3 Schritte genügen um die exakte Lösung zu finden. Abbildung aus dem Seminarbuch von 2014 [2].

dieses Verhalten noch besser an einem 3D Beispiel. Es wäre also wünschenswert einen Algorithmus zu finden, welcher:

- in N Schritten die exakte Lösung findet
- trotzdem schnell konvergiert (eine gute Approximation findet sich bereits nach weniger Schritten)

Dies wird mit dem CG-Algorithmus erreicht.

24.3.2 Optimale Suchrichtung

Wie in 24.2.5 anhand eines Beispiels gezeigt wurde, ist es sinnvoll dazu jeweils A -orthogonalisierte Richtungen verwendet werden. Kombiniert mit der optimalen Abstieglänge aus 24.2.2 führt dies dazu, dass genau N Schritte zur exakten Lösung führen. Wie beim Abstieg entlang der Koordinatenachsen werden also nur neue Richtungen d_{k+1} verwendet, welche A -orthogonal auf allen bisherigen Richtungen stehen:

$$d_{k+1} \perp_A d_k \perp_A d_{k-1} \cdots \perp_A d_0. \quad (24.22)$$

Jetzt stellt sich also noch die Frage, wie diese A -orthogonalen Richtungen zu wählen sind um eine schnelle Konvergenz zu erreichen. Es bietet sich wiederum der negative Gradient $r_{k+1} = b - Ax_{k+1}$ (Residuum, vgl. Beobachtung 1) an, welcher danach mithilfe eines Orthogonalisierungsverfahrens A -orthogonalisiert werden kann. Der negative Gradient entspricht per Definition der lokal optimalen Richtung um schnell abzusteigen und ist somit die beste Wahl.

Wir verwenden das Gram-Schmidt-Orthogonalisierungsverfahren um den Gradienten auf den bisherigen Richtungen zu A -orthogonalisieren. Dabei wollen wir orthogonal in Bezug auf A sein, weshalb das verallgemeinerte Skalarprodukt $\langle \cdot, \cdot \rangle_A$ verwendet wird

$$d_{k+1} = r_{k+1} - \sum_{i=0}^k \frac{\langle d_i, r_{k+1} \rangle_A}{\langle d_i, d_i \rangle_A} d_i. \quad (24.23)$$

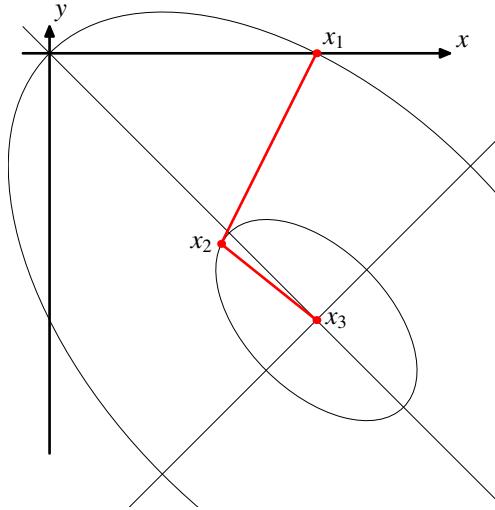


Abbildung 24.4: Zweidimensionaler Abstieg mit dem CG-Algorithmus, zwei Schritte genügen um die exakte Lösung zu finden. Abbildung aus dem Seminarbuch von 2014 [2].

Diese vielen A -Skalarprodukte sind rechenintensiv und würden die Performance des CG-Algorithmus beeinträchtigen. Im nächsten Abschnitt wird gezeigt, dass es genügt das Orthogonalisierungsverfahren auf der letzten Abstiegsrichtung durchzuführen. Die neue Richtung ist dann automatisch auch auf allen vorherigen Richtungen orthogonal. In der Summe von (24.23) wird daher nur der letzte Term benötigt. Dies führt zur folgenden, vereinfachten Berechnung

$$d_{k+1} = r_{k+1} - \frac{\langle d_k, r_{k+1} \rangle_A}{\langle d_k, d_k \rangle_A} d_k. \quad (24.24)$$

Abbildung 24.4 zeigt, wie in einem zweidimensionalen Problem in zwei Schritten die exakte Lösung gefunden wird und schon nach Schritt 1 eine möglichst gute Approximation erreicht wird. Die A -Orthogonalisierungen sind besser im dreidimensionalen Beispiel von Abbildung 24.5 zu sehen.

24.3.3 Wieso genügt Orthogonalisierung auf der letzten Richtung?

Für die Effizienz des Algorithmus ist es entscheidend, dass eine einzige Orthogonalisierung genügt. Wieso dies funktioniert, soll hier in abgekürzter Form bewiesen werden.

Beweis. Für den Beweis genügt es den Fall $b = 0$ zu betrachten, da b nur einen Offset bei der Residuumsberechnung darstellt. Damit eine A -Orthogonalisierung des Residuums auf d_k genügt, muss das Residuum bereits A -orthogonal auf den vorherigen $\langle d_1, \dots, d_{k-1} \rangle$ Richtungen stehen

$$r_{k+1} \perp_A \langle d_1, \dots, d_{k-1} \rangle. \quad (24.25)$$

Aus dem Aufbau des Algorithmus lässt sich die Vermutung ableiten, dass die Residuen und die Richtungen den selben Raum aufspannen. Wir nehmen also an, dass

$$\langle r_1, \dots, r_{k-1} \rangle = \langle d_1, \dots, d_{k-1} \rangle. \quad (24.26)$$

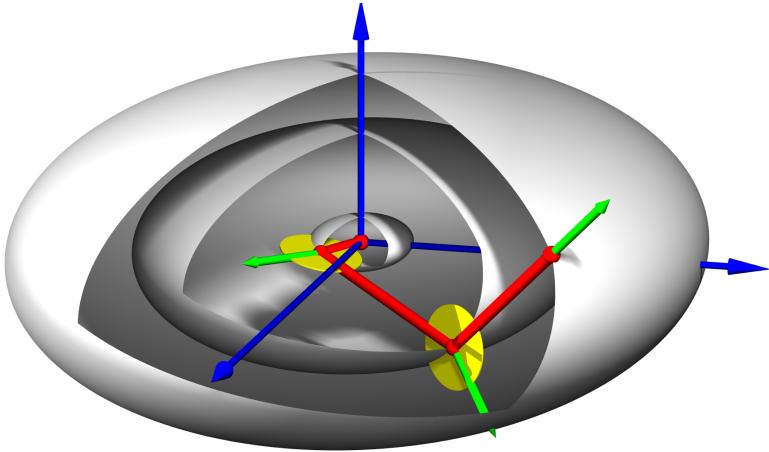


Abbildung 24.5: Dreidimensionaler Abstieg mit dem CG-Algorithmus, drei Schritte genügen um die exakte Lösung zu finden. Abbildung aus dem Seminarbuch von 2014 [2].

Die Korrektheit dieser Annahme wird nachträglich bewiesen. Mit dieser Eigenschaft lässt sich die Gleichung (24.25) umstellen zu

$$\begin{aligned} r_{k+1} &\perp_A \langle r_1, \dots, r_{k-1} \rangle \\ 0 &= \langle r_{k+1}, r_i \rangle_A \quad \forall i < k-1 \\ &= \langle r_{k+1}, Ar_i \rangle \quad \forall i < k-1. \end{aligned} \quad (24.27)$$

Wir nehmen ausserdem an, dass sich die Residuen als Linearkombinationen von r_1 und A ausdrücken lassen

$$\langle r_1, \dots, r_k \rangle = \langle r_1, Ar_1, \dots, A^{k-1}r_1 \rangle. \quad (24.28)$$

Auch diese Annahme wird später noch bewiesen. Dann können wir die Menge der Vektoren Ar_i aus (24.27) schreiben als

$$\begin{aligned} A\langle r_1, r_2, \dots, r_{k-1} \rangle &= A\langle r_1, Ar_1, \dots, A^{k-2}r_1 \rangle \\ &= \langle Ar_1, A^2r_1, \dots, A^{k-1}r_1 \rangle \\ &\subset \langle r_1, Ar_1, \dots, A^{k-1}r_1 \rangle = \langle r_1, r_2, \dots, r_k \rangle. \end{aligned} \quad (24.29)$$

Da der Algorithmus eine optimale Schrittweite verwendet, steht der neue Vektor x_{k+1} A -orthogonal auf allen bisherigen Abstiegsrichtungen d_1, \dots, d_k . Dieses Verhalten ist gut sichtbar in Abbildung 24.4. Da $r_{k+1} = -Ax_k$ (weil $b = 0$), führt die A -Orthogonalität von x_{k+1} dazu, dass r_{k+1} orthogonal auf d_1, \dots, d_k steht

$$\langle x_{k+1}, d_i \rangle_A = \langle Ax_{k+1}, d_i \rangle = \langle r_{k+1}, d_i \rangle = 0 \quad \forall i \leq k. \quad (24.30)$$

Durch Anwenden von Gleichung (24.26) auf (24.29) ist die Menge der Vektoren Ar_i

$$\langle r_1, r_2, \dots, r_k \rangle = \langle d_1, d_2, \dots, d_k \rangle. \quad (24.31)$$

Wenn wir dies mit Gleichung (24.30) vergleichen, sehen wir dass dadurch die Bedingung aus Gleichung (24.27) erfüllt ist. Somit ist die Anfangsvermutung $r_{k+1} \perp_A \langle d_1, \dots, d_{k-1} \rangle$ bewiesen. \square

In diesem Beweis haben zwei Teile gefehlt, welche nun noch nachträglich bewiesen werden. Das erste wäre die Annahme, dass die Residuen und die Richtungen den selben Raum aufspannen $\langle r_1, \dots, r_{k-1} \rangle = \langle d_1, \dots, d_{k-1} \rangle$.

Beweis. Wir beginnen damit den Algorithmus für die Berechnungen von d_k aufzuschreiben (vgl. Gleichung (24.23)) und erhalten

$$\begin{aligned} d_1 &= r_1 \\ d_2 &= r_2 - a_{21}d_1 \\ d_3 &= r_3 - a_{31}d_1 - a_{32}d_2 \\ &\vdots \\ d_k &= r_k - \sum_{i=1}^{k-1} a_{ki}d_i. \end{aligned}$$

Durch Umstellen nach r_k , sieht man dass r_k aus $\langle d_1, d_2, \dots, d_k \rangle$ linear kombiniert werden kann. Dasselbe macht man für d_k , wobei d_i mit $i < k$ aufgelöst wird bis $d_1 = r_1$. Daraus folgt dass auch d_k aus $\langle r_1, r_2, \dots, r_k \rangle$ linear kombiniert werden kann. Somit ist $\langle r_1, \dots, r_{k-1} \rangle = \langle d_1, \dots, d_{k-1} \rangle$ bewiesen. \square

Das zweite wäre die Annahme, dass sich die Residuen als Linearkombinationen von r_1 und A ausdrücken lassen $\langle r_1, \dots, r_k \rangle = \langle r_1, Ar_1, \dots, A^{k-1}r_1 \rangle$.

Beweis. Wir beginnen wiederum damit, den Algorithmus aufzuschreiben (diesmal für r_k) und erhalten

$$\begin{aligned} r_1 &= -Ax_1 \\ r_2 &= -Ax_2 = -A(x_1 - r_1) = Ar_1 - r_1 \\ r_3 &= -Ax_3 = -A(x_2 - r_2) = Ar_2 - r_2 \\ &\vdots \\ r_k &= Ar_{k-1} - r_{k-1} \in \langle r_1, Ar_1, \dots, A^{k-1}r_1 \rangle. \end{aligned}$$

womit diese zweite Annahme bereits bewiesen wäre. \square

Eine noch ausführlichere Variante dieses Beweises findet sich im Dokument [1].

24.4 Algorithmus

In diesem Abschnitt wird der ganze Algorithmus noch einmal formell aufgeschrieben. Diese Formulierung des Algorithmus bildet die Grundlage für eine erfolgreiche Implementation. Die Resultate einer solchen (einfachen) Implementation folgen im nächsten Abschnitt.

1. Wähle initiales x_1 zufällig.
2. Berechne die erste Abstiegsrichtung als $d_1 = r_1 = b - Ax_1$.
3. Berechne die optimale Schrittänge $\alpha = \frac{\langle d_k, r_k \rangle}{\langle d_k, d_k \rangle_A} = \frac{d_k^T r_k}{d_k^T A d_k}$.
4. Führe den Schritt aus $x_{k+1} = x_k + \alpha d_k$.

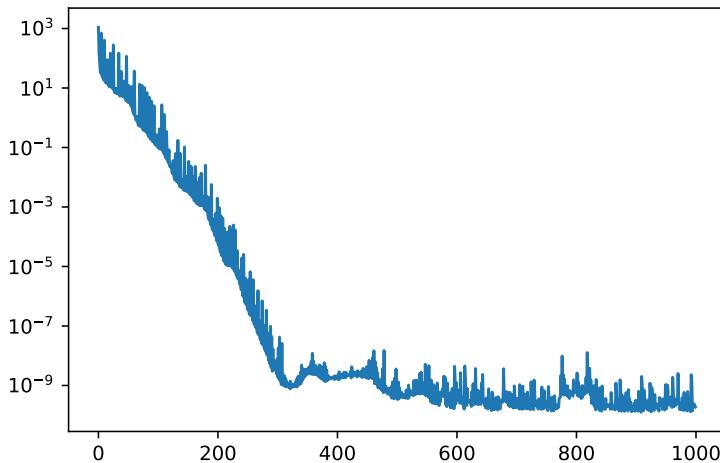


Abbildung 24.6: Betrachtung des absoluten Fehlers in Abhängigkeit der Anzahl Schritte bei schlechte Konditionierung ($\kappa_A = 750000$).

5. Berechne das neue Residuum $r_{k+1} = b - Ax_{k+1}$.
6. Falls $r_{k+1} = 0$: Beende den Algorithmus.
7. Berechne neue Abstiegsrichtung $d_{k+1} = d_{k+1} = r_{k+1} - \frac{\langle d_k, r_{k+1} \rangle_A}{\langle d_k, d_k \rangle_A} d_k = r_{k+1} - \frac{d_k^T A r_{k+1}}{d_k^T A d_k} d_k$.
8. Wiederhole ab Punkt 3.

24.5 Ergebnisse

In diesem Abschnitt werden kurz die Ergebnisse der Implementation dargestellt. Die Implementation wurde mit Python erstellt und ermöglicht eine Konvergenzuntersuchung (Code ist zu finden in [3]). Es zeigte sich, dass die Konvergenzgeschwindigkeit stark von der Konditionierung von A abhängt. Die Konditionierung wird mit der Konditionszahl κ_A gemessen, welche das Verhältnis vom grössten zum kleinsten Eigenwert von A bezeichnet. Je grösser κ_A ist, desto schwerer ist ein Problem zu lösen. Optimal wäre ein $\kappa_A = 1$, wobei dann die Niveaulinien die Form einer Hyperkugel (Kreis in zwei Dimensionen) annehmen.

Alle nachfolgenden Beispiele lösen ein $N = 1000$ -dimensionales Problem, was in einer 1000×1000 Matrix A resultiert. In Abbildung 24.6 sieht man ein Beispiel einer schlechten Konditionierung. Dabei wird bei etwa $k = 200$ Schritten eine Genauigkeit von 10^{-3} erreicht. Abbildung 24.7 dagegen zeigt das Verhalten bei sehr guter Konditionierung. Hier wird bereits nach wenigen Schritten die Genauigkeit von 10^{-3} erreicht. Also haben wir einen starken Einfluss der Konditionierung auf die Konvergenzgeschwindigkeit. Falls schnelle Konvergenz verlangt wird, kann sich somit eine Präkonditionierung lohnen, welche κ_A verbessern würde.

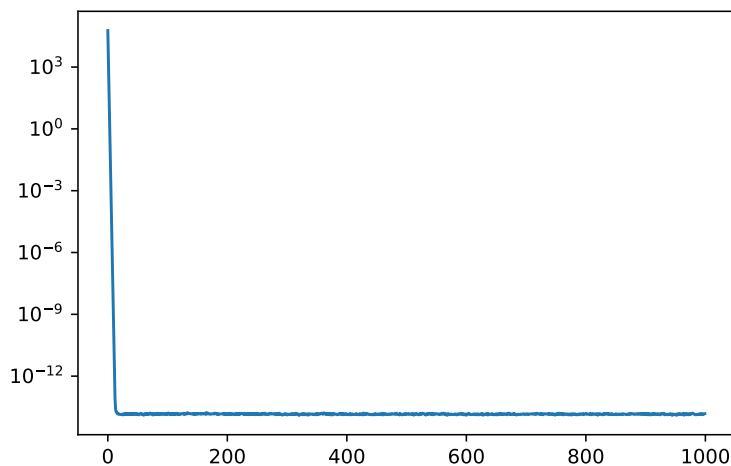


Abbildung 24.7: Betrachtung des absoluten Fehlers in Abhangigkeit der Anzahl Schritte bei guter Konditionierung ($\kappa_A = 1.1$).

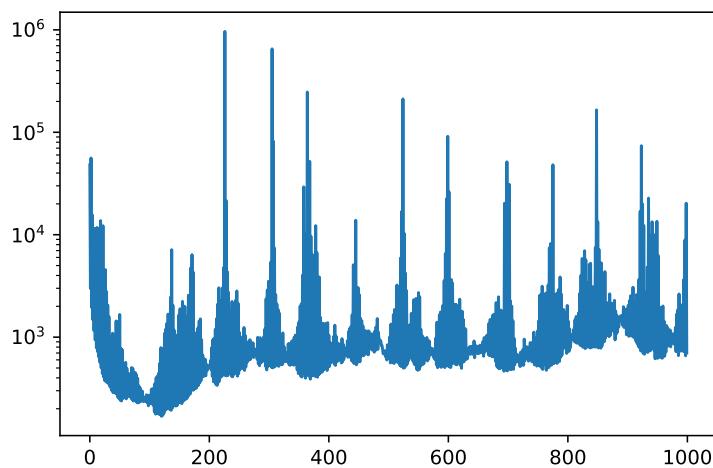


Abbildung 24.8: Betrachtung des absoluten Fehlers in Abhangigkeit der Anzahl Schritte falls A nicht positiv definit ist. Es ergibt sich keine Konvergenz.

Zum Abschluss zeigt Abbildung 24.8 was passiert wenn A nicht positiv definit ist. Das in 24.2.1 aufgestellte Minimierungsproblem stimmt dadurch nicht und der CG-Algorithmus scheitert. Dies äussert sich durch starke Oszillation bei hohem absolutem Fehler.

24.5.1 Fazit

Die hier vorgestellte Methode der konjugierten Gradienten (CG) liefert schnelle, genaue Resultate für grosse lineare Gleichungssysteme. Besonders interessant ist, dass theoretisch sogar die exakte Lösung gefunden wird (nach N Schritten). In der Implementation stellt sich dies aufgrund der begrenzten numerischen Genauigkeit natürlich als unmöglich heraus. Trotzdem ist diese Eigenschaft hilfreich, da es eine fixe Obergrenze für die Anzahl Schritte darstellt. Falls die Kondition gut ist (was häufig der Fall ist), ist die Konvergenz sehr schnell und es werden deutlich weniger als N Schritte benötigt.

Literatur

- [1] Andreas Müller. “CG-Algorithmus und Orthogonalität”. In: (2020). URL: <https://github.com/AndreasFMueller/SeminarNumerik/tree/master/scratch/cg>.
- [2] Andreas Müller. *High Performance Computing*. Hochschule für Technik Rapperswil, 2014.
- [3] Raphael Unterer. *Einfache Python Implementation*. 2020. URL: <https://github.com/AndreasFMueller/SeminarNumerik/tree/master/buch/papers/cg/python>.

25

Numerische Ableitung

Martin Stypinski

25.1 Einleitung

Dieses Kapitel befasst sich mit der numerischen Ableitung, im Speziellen mit der Finite Differenzen Methode (FDM). Bei dieser Methode handelt es sich um ein Verfahren zur Bestimmung der Ableitung. Die Methode wurde von Brook Taylor im siebzehnten Jahrhundert eingeführt, war aber konzeptionell schon Isaac Newton und Leonhard Euler viel früher bekannt. Das Verfahren ist mathematisch sehr einfach zu erklären und ist ebenfalls einfach umzusetzen. Die Einfachheit der Methode lässt Raum, um ein etwas schwierigeres Beispiel zu verwenden, da die Umsetzung in Code unkompliziert ist.

Als Einführung der Methode wird aus diesem Grunde der Gradient im neuronalen Netzwerk berechnet, welcher eine zentrale Rolle im Lernprozess spielt. Neuronale Netzwerke sind durch die kostengünstig verfügbare Computerleistung in den letzten Jahrzehnten sehr populär geworden. Moderne Forschung setzt den Schwerpunkt oft auf den angewandten Bereich des Gebiets. Im Fokus ist eine Vielzahl von Problemen, meistens im Bereich der Bild-, Signal- oder Mustererkennung, aber auch Sprachsynthese und -übersetzung können von neuronalen Netzwerken profitieren. Die fundamentalen und mathematischen Grundlagen dieses Gebiets werden heute teils weniger stark hinterfragt, da ein sehr hoher Abstraktionsgrad die Mathematik versteckt.

Die weite Verbreitung und die günstige Computerleistung resultieren in immer grösseren Netzwerken und komplizierteren Strukturen, welche durch ihre wachsende Tiefe auch anfälliger für numerische Probleme sind. Deshalb liegt der Fokus dieser Arbeit auf einfacher Mathematik, die aufzeigt, wo Probleme während des Trainings von sehr tiefen neuronalen Netzwerken entstehen können.

25.2 Lernen in neuronalen Netzwerken

Die Optimierung von neuronalen Netzen wird auch als *Lernen* bezeichnet und funktioniert mittels Gradientenabstiegverfahren. Nach jeder vollständigen Iteration über den gesamten Datensatz

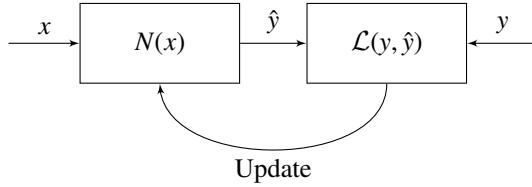


Abbildung 25.1: Trainieren eines neuronalen Netzwerks

(Epoche) wird der Gradient neu berechnet, um den Fehler zu minimieren. Die Gradientenberechnung funktioniert in den meisten neuronalen Frameworks mittels Backpropagation [1]. Die Stärke des Backpropagation-Algorithmus ist, dass die Gradientenberechnung auf moderner Hardware sehr rasch vollzogen werden kann. Gleichzeitig ist aber ein Nachteil, dass die errechnete Ableitung anfällig für Rauschen der Daten sein kann. Aus diesem Grund ist es spannend, einen anderen Ansatz zu wählen, welcher in den kommenden Abschnitten beschrieben wird.

25.2.1 Was ist Lernen?

Neuronale Netzwerke lernen durch den Vergleich einer errechneten Ausgabe mit der erwarteten Ausgabe. Sie gehören in der Regel zur Gruppe der *supervised learning*-Algorithmen. Diese Gruppe von Algorithmen benötigen einen Datensatz, welcher aus einem Input x und einem erwarteten Output y besteht. Das neuronale Netzwerk als Blackbox versucht aus dem Input x einen brauchbaren Output \hat{y} zu erzeugen, welcher dann mittels einer Fehlerfunktion \mathcal{L} mit dem bestehenden Output y verglichen wird.

Der Lernprozess kann schematisch wie in Abbildung 25.1 dargestellt werden. Nach jeder Trainingsiteration werden die inneren Parameter des neuronalen Netzwerks $N(x)$ optimiert, um das bestmögliche Resultat für die gegebenen Datenpaare (x, y) zu errechnen. Die Bewertung des aktuellen Lernstands ist mittels der Fehlerfunktion

$$\mathcal{L}(y_i, \hat{y}_i) \quad (25.1)$$

gegeben. Dabei ist $\hat{y} = N(x)$ der berechnete Output des neuronalen Netzwerks. Dies lässt sich in die Fehlerfunktion

$$\mathcal{L}(y, N(x)) \quad (25.2)$$

einsetzen. Die Fehlerfunktion \mathcal{L} liefert somit ein Mass für den Fehler, also eine quantitative Bewertung des Fehlers und wird in der Literatur auch oft *Loss*-Funktion genannt. Als einfaches Beispiel für eine solche Fehlerfunktion \mathcal{L} kann die mittlere quadratische Abweichung (Mean-Squared-Error)

$$\text{MSE: } \mathcal{L}(y, \hat{y}) = \frac{1}{2} \sum_i (\hat{y}_i - y_i)^2 \quad (25.3)$$

verwendet werden. Diese hat die Eigenschaft, dass sie vorzeichenunabhängig ist und bei grösserer Abweichung zwischen y und \hat{y} der Fehler auch entsprechend quadratisch grösser wird. Der Lenprozess versucht also ein Minimum

$$\min(\mathcal{L}(y, N(x))) \quad (25.4)$$

zu finden, in dem es die inneren Parameter des neuronalen Netzwerks so wählt, dass der Fehler möglichst klein wird.

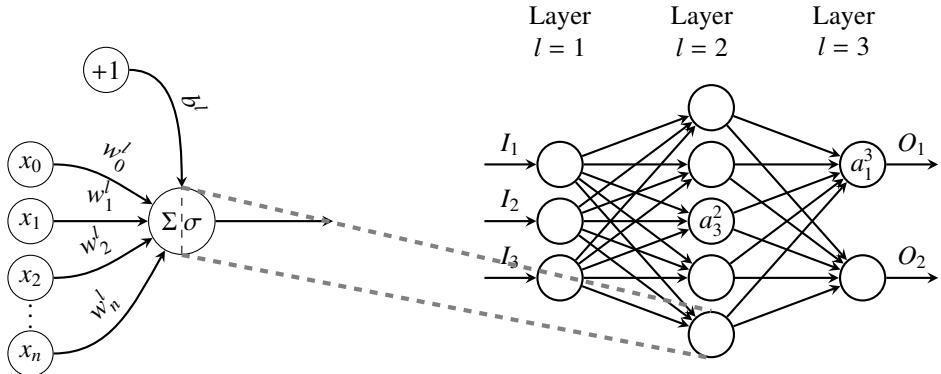


Abbildung 25.2: Übersicht eines neuronalen Netzwerks

25.2.2 Der Parameterraum

Das neuronale Netzwerk wird durch eine sehr hohe Anzahl an Parametern charakterisiert, welche durch den Lernprozesses optimiert werden müssen. Der innere Aufbau eines neuronalen Netzwerks erklärt die Herkunft dieser Parameter. Abbildung 25.2 zeigt die Komponenten eines einfachen neuronalen Netzwerks. W sind die Gewichte, b die Offsets, oft auch *bias* genannt, und σ ist die Aktivierungsfunktion. Die Aktivierungsfunktion in den mittleren Schichten wird benötigt, um die Linearität der verschiedenen Schichten zu brechen. In der letzten Schicht wird die Aktivierungsfunktion σ aber hauptsächlich verwendet, um den Output auf das vorliegende Problem abzubilden. Eine Ausgabe zwischen $[-\infty, \infty]$ ist erwünscht, sofern ein Regressionsproblem vorliegt. In diesem Fall ist ausnahmsweise eine lineare Funktion für σ in der letzten Schicht erlaubt. Bei einem stochastischen Problem werden die Outputwerte als Wahrscheinlichkeiten interpretiert und sollten daher Werte im Intervall $[0, 1]$ annehmen, folglich wird eine Funktion mit entsprechendem Wertebereich für σ gewählt. Aus diesem Grund kann σ oft eine sigmoid oder tanh Funktion und in Ausnahmefällen auch eine lineare Funktion sein. Die Funktion

$$a_j^l = \sigma \left(\sum_k w_{jk}^l \cdot a_k^{l-1} + b_j^l \right) \quad (25.5)$$

beschreibt für jedes Neuron a_j^l aus Abbildung 25.2 den Zusammenhang zwischen den Gewichten und den vorhergehenden Neuronen a_k^{l-1} . Der Input Layer

$$I_k = a_k^0 \quad (25.6)$$

nimmt die Daten x_i des Datensatzes x entgegen. Da die Neuronen mit den vorangegangenen Schichten gekoppelt sind, kann man die Gleichung (25.5) für die verschiedenen Schichten ineinander einsetzen und abkürzend

$$\hat{y}(x) = \sigma^l(W^l \sigma^{l-1}(W^{l-1} \cdots \sigma^1(W^1 x + b^1) + b^{l-1} \cdots)) \quad (25.7)$$

schreiben. Dementsprechend resultiert daraus das Minimalproblem

$$\min \{\mathcal{L}(y, \sigma^l(W^l \sigma^{l-1}(W^{l-1} \cdots \sigma^1(W^1 x + b^1) + b^{l-1} \cdots))), \quad (25.8)$$

wobei die Parameter w und b während dem Training bestimmt werden, während x und y durch den Trainings-Datensatz gegeben sind. Um die Nomenklatur zusammenzufassen, gelten folgende Bezeichnungen in einem neuronalen Netzwerk:

- x : Input ins neuronale Netzwerk
- y : Erwarteter Output aus dem neuronalen Netzwerk gemäss dem Trainingsdatensatz
- \hat{y} : Effektiver gerechneter Output aus den neuronalen Netzwerk
- \mathcal{L} : Die Verlustfunktion (*loss*) welche den Fehler zwischen dem gerechneten und dem tatsächlichen Output berechnet.
- w_{jk}^l : Das Gewicht des k -ten Neurons in der Schicht $(l - 1)$ zum j -ten Neuron.
- b_j^l : Das Offset-Gewicht (*bias*) für das j -te Neuron in der Schicht l .
- σ : Die Aktivierungsfunktion zur Brechung der Linearität.

Das Trainieren oder Lernen eines neuronalen Netzwerks ist im Grunde nichts anderes als das Bestimmen des Minimums des Fehlers mit Hilfe der Parameter w und b für die gegebene Problemstellung.

25.2.3 Das Optimierungsproblem

Wie bereits erwähnt, ist durch die extrem hohe Anzahl an Parametern ein Durchprobieren aller Parameterkombinationen nicht möglich. Es könnte kein Resultat in nützlicher Frist gefunden werden können. Aus diesem Grund werden andere Verfahren verwendet, um das neuronale Netzwerk zu optimieren. Alle diese Verfahren basieren auf dem Gradientenabstieg und werden in der Literatur oft auch *Optimizer* genannt. Näheres zum Gradientenabstieg kann in Kapitel 24 gefunden werden.

Ein einfacher Algorithmus ist der stochastische Gradientenabstieg. Der Abstiegsschritt

$$p_{\text{new}} := p - \eta \nabla \mathcal{L}_i(p) \quad (25.9)$$

findet Parameterwerte p_{new} , für die der Fehler kleiner ist. Die Parameter p stehen für sämtliche inneren Gewichte w und Offsets b . Die Schrittweite η , oft auch *learning rate*, muss experimentell ermittelt werden. Die Kunst liegt darin, η so zu wählen, dass lokale Minima und Rauschen die Konvergenz nicht behindern. Der Algorithmus bricht erst ab, wenn ein Minimum erreicht ist oder die vorgegebene maximale Anzahl an Iterationen ausgeschöpft ist. Die Fehlerfunktion

$$\mathcal{L}(w_1, \dots, w_n, b_1, \dots, b_n) = \mathcal{L}(p) = \sum_{i=1}^n \mathcal{L}_i(p) = \sum_{i=1}^n \mathcal{L}(\hat{y}_i - y_i)^2 \quad (25.10)$$

berechnet den Fehler, welcher durch w und b beeinflusst wird. Die Konsequenzen der partiellen Ableitung nach den jeweiligen Parametern werden am Beispiel eines Netzwerks mit nur einem Neuron wie in Abbildung 25.3 berechnet. An diesem Beispiel

$$\hat{y} = \sigma(w_1 \cdot x + b_1) \quad (25.11)$$

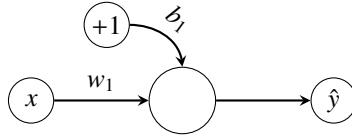


Abbildung 25.3: Übersicht über ein neuronales Netzwerk

lassen sich alle Effekte gut zeigen. Dieses Netzwerk kann in die Fehlerfunktion \mathcal{L} gemäss der Gleichung (25.2) eingesetzt werden. Die anschliessende Differentiation nach den inneren Parametern w und b führt zum Gradienten der beiden Parameter. Die neuen Parameter sind

$$\begin{aligned} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix}_{\text{new}} &:= \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} - \eta \sum_{i=1}^n \left[\frac{\partial}{\partial w_1} (\sigma(w_1 x_i + b_1) - y_i)^2 \right] \\ &= \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} - \eta \sum_{i=1}^n \left[2(\sigma(w_1 x_i + b_1) - y_i) \cdot \sigma'(w_1 x_i + b_1) \cdot (w_1 x_i + b_1) x_i \right] \end{aligned} \quad (25.12)$$

werden in jeder Iteration aktualisiert.

Das Beispiel eines neuronalen Netzwerks mit einem Neuron zeigt auf, dass mit steigender Netzkomplexität die Terme noch viel grösser werden. Dies ist auf die Kettenregel $u'(v(x)) = u'(v(x)) \cdot (v'(x))$ zurückzuführen. Mit diesem Verfahren ist der Gradientenabstieg zwar möglich, jedoch sehr rechenintensiv. Aus diesem Grunde verwenden die meisten Frameworks den Backpropagation-Algorithmus, welcher etwas effizienter ist.

25.2.4 Der Backpropagation-Algorithmus

Die analytische Berechnung aller partiellen Ableitung nach allen Parametern ist nicht wirklich praktikabel. Es sind einfach zu viele Dimensionen, welche mittels der Kettenregel abgeleitet werden müssen. Weiter können die Zusammenhänge zwischen den Schichten nicht wirklich genutzt werden, da nach jedem Parameter einzeln abgeleitet wird.

Aus diesem Grund ist der Backpropagation-Algorithmus sehr verbreitet. Er löst genau dieses Problem, in dem er den Fehler pro Schicht berechnet und dann den Fehler der davor liegende Schicht davon abhängig macht. Dies spart einen Teil des Rechenaufwandes der Gradientenberechnung und macht den Algorithmus so einiges effizienter.

Das Ziel des Backpropagation-Algorithmus ist nun die partiellen Ableitungen $\partial \mathcal{L} / \partial w_{jk}^l$ und $\partial \mathcal{L} / \partial b_j^l$ für den Fehler eines neuronalen Netzwerks der Form

$$\mathcal{L}(y_i, \sigma^l(W^l \sigma^{l-1}(W^{l-1} \cdots \sigma^1(W^1 x_i + b^1) + b^{l-1} \cdots))) \quad (25.13)$$

zu finden, wobei W^n und b^n jeweils die Gewichte im n -ten Layer des neuronalen Netzwerks darstellen. Dies gilt unter der Annahme, dass die Fehlerfunktion \mathcal{L} einen Skalar als Resultat liefert und die einzelnen Schichten l jeweils nur mit ihrer benachbarten Schicht $l+1$ verbunden sind.

Für den Zusammenhang der Layer führen wir die Notation

$$\begin{aligned} a_j^l &= \sigma(z_j^l) \\ z_j^l &= \sum_k w_{jk}^l \cdot a_k^{l-1} + b_j^l \end{aligned} \quad (25.14)$$

ein. a ist bereits aus der Gleichung (25.5) bekannt, z ist im Grunde der gewichtete Input in ein Neuron ohne die Aktivierungsfunktion σ . Diese Notation vereinfacht die weitere Herleitung.

Die Änderung des Fehlers in einem einzelnen Neuron

$$\delta_j^l = \frac{\partial \mathcal{L}}{\partial a_j^l} \sigma'(z_j^l) \quad (25.15)$$

lässt sich Ausdrücken als Ableitung der Fehlerfunktion \mathcal{L} nach dem jeweiligen Neuron. In der letzten Schicht mit Index L (*last*) gilt somit in Matrixschreibweise

$$\delta^L = \nabla_a \mathcal{L} \odot \sigma'(z^L), \quad (25.16)$$

wobei \odot der komponentenweise Multiplikation der Matrix (Hadamard Produkt) entspricht. Um den Fehler zu berechnen, legen wir uns für die Funktion \mathcal{L} auf die quadratische Fehlerfunktion

$$\mathcal{L} = \frac{1}{2} \sum_j (\hat{y} - y)^2. \quad (25.17)$$

fest, auch bekannt als L_2 Mean-Square-Error. Somit resultiert nach der partiellen Ableitung von \mathcal{L} und einsetzen von a^L die Gleichung

$$\delta^L = (\hat{y} - a^L) \odot \sigma'(z^L). \quad (25.18)$$

δ^L ist somit ein Ausdruck der Änderung des Fehlers in der letzten Schicht.

Als nächstes muss nun rückwärts gerechnet — back propagiert — werden. Die Änderung des Fehlers

$$\delta^l = ((W^{l+1} \delta^{l+1})^T \odot \sigma'(z^l)) \quad (25.19)$$

beschreibt nun den Zusammenhang zwischen den Schichten $l+1$ und l , was an den Ausdrücken δ^{l+1} und δ^l abzulesen ist. Die Transponierte der Matrix W^{l+1} kann man sich intuitiv vorstellen, als das 'Rückwärtsrechnen' des gewichteten Fehlers, da $W^{l+1} \cdot (W^{l+1})^T = I$, wobei I für die Einheitsmatrix steht. Dies ermöglicht die Änderung des Fehlers pro Neuron in der davor liegenden Schicht zu bestimmen. Abschließend müssen noch die Gradienten

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial b} &= \delta \\ \frac{\partial \mathcal{L}}{\partial w} &= a_k^{l-1} \delta_j^l = a_{\text{in}} \delta_{\text{out}} \end{aligned} \quad (25.20)$$

berechnet werden. Das Resultat wird verwendet, um mittels Gradientenabstieg die neuen inneren Parameter w_{new} und b_{new} zu bestimmen.

Die Herleitung zeigt sehr schön, dass eine hohe Netztiefe und die rekursive Eigenschaft des Algorithmus viele Multiplikationen als Konsequenz hat. Dies hat zur Folge, dass durch numerische Effekte die Gradientenberechnung rauschbehaftet sein kann, weiter sind die vorderen Schichten viel stärker Betroffen, da mehr Multiplikationen nötig sind um den Gradienten zu bestimmen. Auf dieses Problem wird in den folgenden Kapiteln mit einem Gedankenexperiment eingegangen.

25.3 Ein alternativer Lösungsansatz

Wie bereits erklärt, basiert die Lösung für die Gradientenberechnung darauf ein geeignetes numerisches Verfahren anstelle des Backpropagation-Algorithmus zu verwenden. Der Gradientenabstieg kann somit weiterhin vom Framework übernommen werden, obschon die Gradientenberechnung ersetzt wurde. Dies ermöglicht zu einem späteren Zeitpunkt einen Vergleich beider Verfahren.

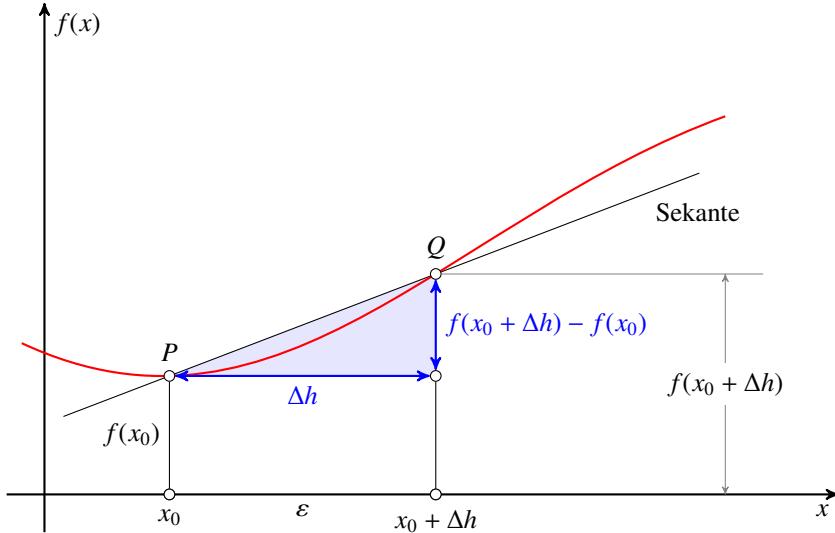


Abbildung 25.4: Differential Quotient in grafischer Darstellung

25.3.1 Finite Differenzen Methode

Die Finite Differenzen Methode ist motiviert durch die klassische Ableitung. Gemäss Abbildung 25.4 wird die Ableitung als Sekante zweier Punkte der Funktion definiert, mit anschliessendem Grenzübergang $\Delta x \rightarrow 0$. Dies entspricht dann der Tangente, welche die Ableitung am Punkt $f(x_0)$ darstellt. Dieses Konzept ist als Differentialquotient

$$f'(x_0) = \lim_{\Delta h \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \quad (25.21)$$

bekannt. In der Numerik existiert die Möglichkeit von $\Delta x \rightarrow 0$ nicht, da die Datenpunkte diskretisiert sind. Stattdessen sei h der kleinstmögliche Abstand zweier benachbarter Datenpunkte. Entsprechend kann man die numerische Ableitung (Differenzenquotient) als Quotienten

$$\begin{aligned} f'(x_0) &= \frac{f(x_0 + h) - f(x_0)}{h} \\ f'(x_0) &= \frac{f(x_0) - f(x_0 - h)}{h} \\ f'(x_0) &= \frac{f(x_0 + h) - f(x_0 - h)}{2h} \end{aligned} \quad (25.22)$$

definieren. Die Genauigkeit ist limitiert durch den Abstand der zwei Stützstellen.

25.3.2 Taylor-Methode

Mithilfe der Taylor-Reihe

$$f(x) = f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \dots \quad (25.23)$$

lässt sich jede beliebige analytische Funktion $f(x)$ approximieren. Dies wird für die Herleitung einer etwas stabileren numerischen Methode ausgenutzt. Um eine höhere Genauigkeit zu erreichen, kann die Anzahl der Stützstellen erhöht werden. Im Grunde wird ein Polynom höherer Ordnung gesucht, welches Stützstellen in einem uniformen Abstand hat. Die Ableitung dieses Polynoms sollte dann eine höhere Genauigkeit aufweisen, da durch die höhere Ordnung die Ursprungsfunktion besser um den Punkt der Ableitung approximiert werden kann. Als Beispiel soll die erste Ableitung mit fünf Stützstellen in gleichem Abstand h berechnet werden:

$$\frac{df}{dx} \approx Af(x - 2h) + Bf(x - h) + Cf(x) + Df(x + h) + Ef(x + 2h). \quad (25.24)$$

Die Koeffizienten A, \dots, E sind so zu bestimmen, dass der Ausdruck rechts in möglichst hoher Ordnung in h mit der Ableitung übereinstimmt. Die Funktionswerte an den Stützstellen, $x - 2h, x - h, \dots, x + 2h$ können nun mittels Taylor-Reihe approximiert werden:

$$Af(x - 2h) \approx Af(x) + Af'(x)(-2h) + A\frac{1}{2}f''(x)(-2h)^2 + A\frac{1}{6}f'''(x)(-2h)^3 + A\frac{1}{24}$$

$$+ f''''(x)(-2h)^4 + A\frac{1}{120}f''''''(x)(-2h)^5$$

$$Bf(x - h) \approx Bf(x) + Bf'(x)(-h) + B\frac{1}{2}f''(x)(-h)^2 + B\frac{1}{6}f'''(x)(-h)^3 + B\frac{1}{24}$$

$$+ f''''(x)(-h)^4 + B\frac{1}{120}f''''''(x)(-h)^5$$

$$Cf(x) = Cf(x)$$

$$Df(x + h) \approx Df(x) + Df'(x)(+h) + D\frac{1}{2}f''(x)(+h)^2 + D\frac{1}{6}f'''(x)(+h)^3 + D\frac{1}{24}$$

$$+ f''''(x)(+h)^4 + D\frac{1}{120}f''''''(x)(+h)^5$$

$$Ef(x + 2h) \approx Ef(x) + Ef'(x)(+2h) + E\frac{1}{2}f''(x)(+2h)^2 + E\frac{1}{6}f'''(x)(+2h)^3 + E\frac{1}{24}$$

$$+ f''''(x)(+2h)^4 + E\frac{1}{120}f''''''(x)(+2h)^5$$

Nach Einsetzen dieser Ausdrücke in die Gleichung 25.24 darf nur der Koeffizient von $f'(x)$ bestehen bleiben und muss folglich 1 ergeben. Der Rest soll verschwinden. Zur Unterstützung der Lesbarkeit können die Terme nach Ableitungsgrad sortiert werden:

$$\begin{aligned} f(x) \cdot (& A + B + C + D + E) = 0 \\ \frac{1}{2}f'(x)(h) \cdot (& -2A - B + D + 2E) = \frac{2}{h} \\ \frac{1}{6}f''(x)(h^2) \cdot (& 14A + B + D + 4E) = 0 \\ \frac{1}{24}f'''(x)(h^3) \cdot (& -8A - B + D + 8E) = 0 \\ \frac{1}{120}f''''(x)(h^4) \cdot (& 16A + B + D + 16E) = 0 \end{aligned} \quad (25.25)$$

Dieses lineare Gleichungssystem für die gesuchten Koeffizienten lässt sich auch in Matrixform

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = \frac{1}{h} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (25.26)$$

schreiben. Die Koeffizientenmatrix lässt sich nach

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = \frac{1}{h} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{h} \begin{bmatrix} \frac{1}{12} \\ -\frac{2}{3} \\ 0 \\ \frac{2}{3} \\ \frac{1}{12} \end{bmatrix} \quad (25.27)$$

invertieren. Dies ergibt die erste Ableitung

$$f'(x) \approx \frac{1f(x-2h) - 8f(x-h) + 8f(x+h) - 1f(x+2h)}{12h} \quad (25.28)$$

angenähert durch fünf Stützstellen. Analog kann man natürlich Approximationen für höhere Ableitungen oder für non-uniforme Stützstellen bekommen.

Fehlerabschätzung

Zur Berechnung der ersten Ableitung mit fünf Stützstellen werden die ersten fünf Terme der Taylor-Reihe verwendet. Die Ungenauigkeit dieser Methode ist somit abhängig vom Restterm

$$\frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} = f'(x) - \frac{1}{30} f^{(5)}(x)h^4 + R_n(x). \quad (25.29)$$

Der Restterm R_n ist abhängig von der Taylor-Reihe der zu ableitenden Funktion. Funktionen lassen sich somit mit mehr Stützstellen genauer approximieren, da der Rest R_n mit steigender Taylor-Ordnung kleiner wird. Wenn aber die Terme höherer Ordnung schneller anwachsen als die Fakultät, also $f^{(n)} > n!$, so ist die Approximation nicht genau.

Aus der Fehlerabschätzung ergeben sich grundsätzlich zwei Typen an Problemen, die interessant sind zu untersuchen. Die Terme des ersten Types werden mit zunehmender Ordnung kleiner, während bei dem zweiten Typ die Terme immer grösser werden. Gleichzeitig muss aber ein guter Kompromiss gefunden werden, welcher den Abstand h nicht zu klein werden lässt. Ein sehr kleines h führt zu mehr Rauschanfälligkeit, da bei benachbarten Punkten ein Rauschen höher ins Gewicht fällt.

Als Beispiel für eine Funktion, bei welcher die Terme kleiner werden, können die trigonometrischen Funktionen dienen. Im Speziellen hat der Sinus

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} f^{(2n+1)}(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + R_n \quad (25.30)$$

eine alternierende Taylor-Reihe. Der Fehler

$$R_n < \left| \frac{x^n}{n!} \right| \quad (25.31)$$

kann somit, bei Approximation durch 5 Stützstellen, auf einem Intervall von $-1 < x < 1$ nicht grösser werden als $\approx 2.5 \cdot 10^{-8}$. Auf der anderen Seite erweist sich die die geometrische Reihe

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n \approx 1 + x + x^2 + x^3 + x^4 + R_n \quad (25.32)$$

als sehr schwer zu approximieren. Der Fehler R_n wächst ab $x > 1$ unbeschränkt.

Für die Experimente werden aus diesem Grund verschiedene Abstände h und eine unterschiedliche Anzahl Stützstellen verwendet, um ein Optimum für den jeweiligen Fall zu finden. Es ist jedoch zu erwarten, dass mit steigender Anzahl an Stützstellen das neuronale Netzwerk schneller konvergiert, da die einzelnen Gradienten weniger Rauschen aufweisen.

25.4 Experiment

Die Resultate dieser Arbeit sollen hauptsächlich zeigen, ob die Finite Differenzen Methode als Option zur Optimierung neuronaler Netzwerke geeignet ist. Es ist klar, dass implementationsbedingt die Trainingsgeschwindigkeit dem Backpropagation-Algorithmus nicht das Wasser reichen kann. Viel wichtiger ist eine qualitative Aussage machen zu können, ob die Methode eine interessante Variante darstellt und ob das Gedankenexperiment funktioniert.

Um dies an einem einfachen Beispiel zu verdeutlichen, wird versucht, eine logische XOR-Verknüpfung zu trainieren. Ein XOR-Gate ist ein elektronisches Bauteil, welches eine *Entweder-Oder*-Logik abbildet. Im gewählten Fall hat das Bauteil zwei digitale Eingänge. Ist entweder der Erste oder der Zweite aktiv, so ist der Ausgang ebenfalls aktiv. Ist keiner der beiden Ausgänge oder beide Ausgänge aktiv, so ist der Ausgang inaktiv. Dies kann man der Logiktabelle in Abbildung 25.5 entnehmen. Das gewählte Netzwerk, um diese Logik abzubilden, besteht aus drei Schichten

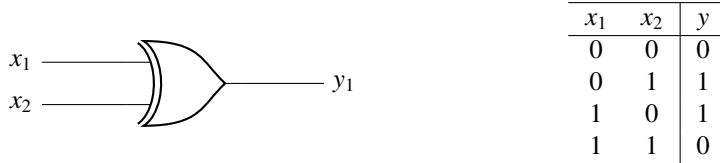


Abbildung 25.5: Das XOR-Gate und die dazugehörige Logiktabelle

und ist in Abbildung 25.6 ersichtlich. Die Mittelschicht (hidden layer) wird benötigt, da sich mit nur einer Schicht keine Lösung finden lässt, welche die zwei Inputs auf den Output abbilden kann. In Abbildung 25.7 wird das Ganze im zweidimensionalen Raum visualisiert. Um die Kriterien der Logiktabelle umsetzen zu können, wird eine Sattelfläche im dreidimensionalen Raum benötigt. Eine Lösung ist in Abbildung 25.8 ersichtlich, welche diese Anforderungen erfüllt. Das Experiment wurde basierend auf dem PyTorch Framework implementiert. Das Netzwerk wurde jeweils mit der Finiten Differenzen Methode und dem Backpropagation-Algorithmus trainiert. Bei der FDM wurden verschiedene Anzahlen an Stützstellen (Support) gewählt und unterschiedliche Abstände h verwendet. Die Resultate sind in den Abbildungen 25.9, 25.10, 25.9 zusammengestellt. Die vertikale-Achse

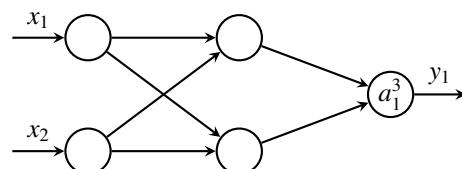


Abbildung 25.6: Das neuronale Netzwerk zum Erlernen des XOR Logikoperators

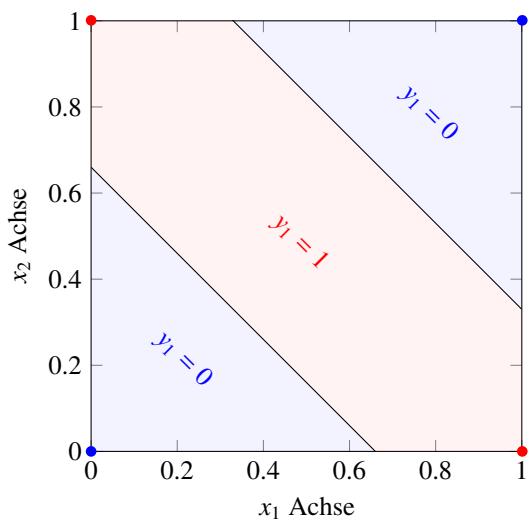


Abbildung 25.7: Visualisierung der Punkte in der x_1 - x_2 -Ebene und deren Separierbarkeit.

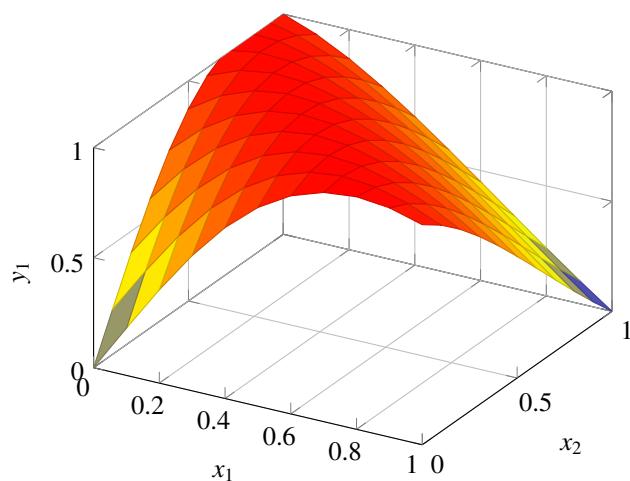
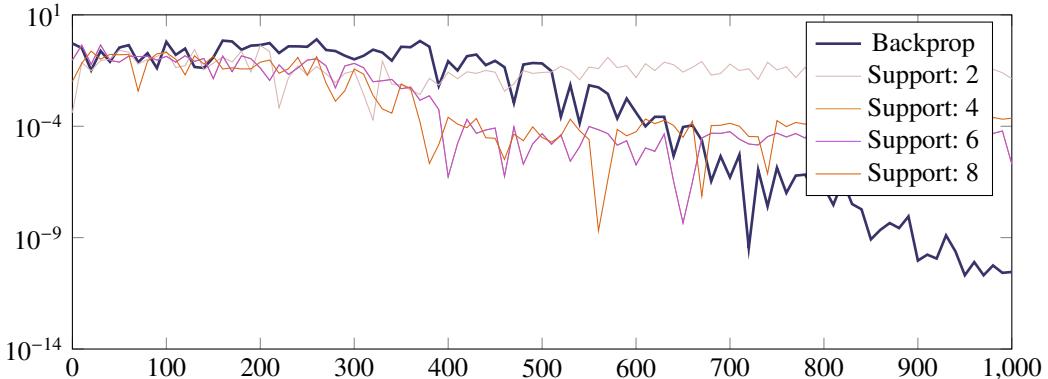
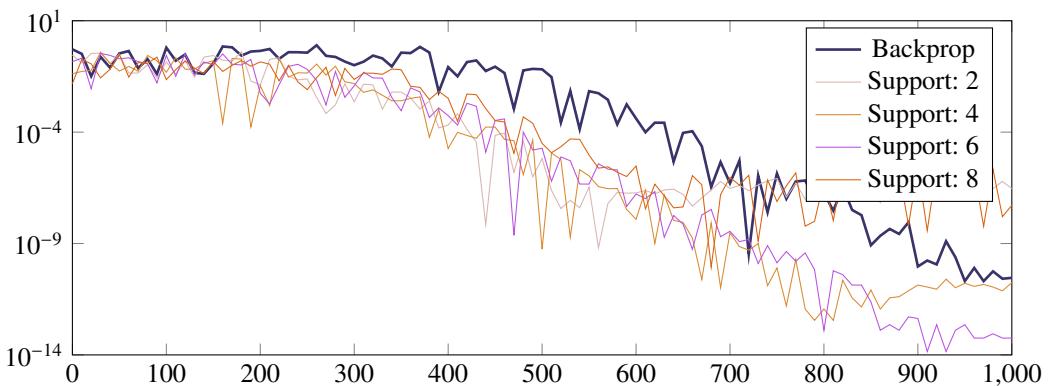


Abbildung 25.8: Visualisierung der Sattelfläche für den Output y_1 .

Abbildung 25.9: Resultate des Gradientenabstiegs mit Stützstellenabstand $h = 1$.Abbildung 25.10: Resultate des Gradientenabstiegs mit Stützstellenabstand $h = 0.1$.

in den Abbildungen stellt den quadratischen Fehler zwischen \hat{y} und y dar, während die horizontale Achse die Entwicklung über die Anzahl Iterationen aufzeigt.

Es ist erstaunlich, dass bereits bei diesem kleinen Netzwerk numerische Effekte im Backpropagation-Algorithmus in einem etwas höheren Fehler resultieren. Der Fehler von $\approx 10^{-14}$ im besten Fall (FDM) ist um Faktor 10^3 kleiner als die des Backpropagation-Algorithmus.

25.5 Folgerungen

Die Resultate zeigen anhand eines einfachen Netzwerks, dass die hier vorgestellte Implementation für die Berechnung des Gradienten funktioniert. Im direkten Vergleich zwischen dem FDM- und dem Backpropagation-Algorithmus, hat ersterer früher zu einem kleineren Fehler konvergiert, unter der Voraussetzung, dass die Parameter gut gewählt wurden. Nach der gleichen Anzahl an Iterationen lag der Fehler der FDM-Implementation um einen Faktor 10^3 niedriger. Dabei spielt die Anzahl an Stützstellen eine wichtigere Rolle, als die Wahl des Abstandes h . Ohne an dieser Stelle zu stark im Detail auf die Fehlerrechnung einzugehen, kann die folgende Intuition dieses Phänomen erklären.

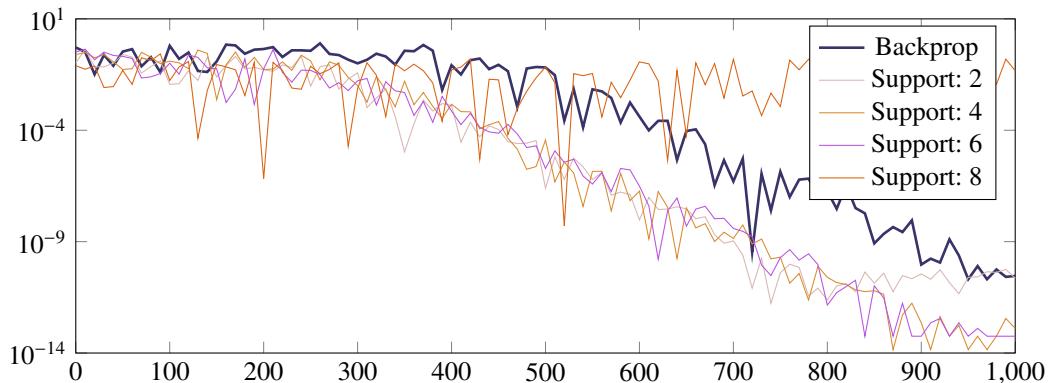


Abbildung 25.11: Resultate des Gradientenabstiegs mit Stützstellenabstand $h = 0.01$.

Die am häufigsten verwendeten Aktivierungsfunktionen in neuronalen Netzwerken haben einen Wertebereich $[-1, 1]$ oder $[0, 1]$. Die Kettenregel hat zur Folge, dass für die Berechnung des Fehlers in den vorderen Schichten diese kleinen Zahlen oft miteinander multipliziert werden müssen. Der Fehler in der Gradientenberechnung nimmt zu, da das numerische Rauschen durch die Gleitkommadarstellung von Schicht zu Schicht grösser wird. Dieses Phänomen wird auch als *vanishing gradient problem* bezeichnet und ist ein bekanntes Problem, welches bei Netzwerken mit hoher Tiefe besonders stark ausgeprägt ist [2]. Durch den Verzicht der Rückwärtsberechnung mittels Kettenregel entfällt dieser Fehler bei der FDM vollständig.

Weiter verbessert eine höhere Anzahl an Stützstellen die Genauigkeit des Gradienten, da gewisse Toleranzen in der Gleitkommadarstellung weniger stark ins Gewicht fällt. Dies kann wie folgt erklärt werden: Das Zeichnen einer Parabel durch drei Punkte mit gewisser Unschärfe ist deutlich ungenauer, als das Zeichnen einer Parabel durch sieben Punkte mit ähnlicher Unschärfe. Als Unschärfe ist die Genauigkeit der Position der Punkte gemeint, durch welche die gezeichnete Parabel verlaufen soll.

Das Experiment hat gezeigt, dass die FDM durchaus Potential zum Trainieren von tiefen neuronalen Netzwerken hat. An diesem Punkt wäre es ausserordentlich spannend gewesen tiefere Netzwerke zu trainieren. An einem etwas älteren aber gut dokumentierten Netzwerk (LeNet-5) sollte dies probiert werden [3]. Das Netzwerk wurde in den achziger Jahren verwendet um Handschriften auf dem bekannten MNIST-Datensatz zu erkennen. Es bietet eine gute Grundlage um qualitative Aussagen zu machen.

Die erfreuliche Genauigkeit, die mit der FDM erreicht werden konnte, hat aber auch ihren Preis. Das Training mittels Backpropagation Algorithmus war in wenigen Minuten beendet, während das Training mittels FDM nach mehreren Tagen noch nicht genügend Iterationen durchlaufen hat, um eine abschliessende qualitative Aussage zu machen. Es war nie Ziel dieses Projektes den Algorithmus soweit zu optimieren, dass ein Vergleich einer solchen Struktur möglich gewesen wäre. Da die Laufzeit sich um einen sehr grossen Faktor unterschieden hat, wurde auf eine Optimierung verzichtet.

Literatur

- [1] Yann Le Cun. “A Theoretical Framework for Back-Propagation”. In: (1988), S. 21–28.

- [2] Sepp Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". In: (Apr. 1991).
- [3] Y. Lecun u. a. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324.

26

Interpolation und numerische Ableitung

Andreas Müller

In Kapitel 3 wurde gezeigt, wie man durch Interpolation gute Approximationen einer Funktion in einem Intervall finden kann. Für äquidistante Stützstellen ist das Interpolationspolynom vor allem in der Mitte des Intervalls sehr genau. Man kann daher versuchen, diese Approximation durch das Interpolationspolynom auch für eine Approximation der Ableitung der Funktion zu verwenden, indem man das Interpolationspolynom ableitet. Am Ende dieses Prozesses sollte sich eine Formel ergeben, welche die Ableitung näherungsweise aus Funktionswerten in der Nähe des interessierenden Punktes bestimmt, ganz ähnlich wie das aus der Taylor-Reihe abgeleitete, in Kapitel 25 beschriebene Verfahren. Dieses Kapitel führt die Berechnung der Gewichte der Funktionswerte durch und untersucht mit Hilfe der Fourier-Transformation, was man von dem Verfahren erwarten kann.

26.1 Das Problem der numerischen Ableitung

Wir gehen von einer in einem Intervall $I = [a, b]$ definierten differenzierbaren Funktion $f: [a, b] \rightarrow \mathbb{R}$ aus. Zu bestimmen ist die Ableitung von f in einem Punkt $x \in [a, b]$. Der Differenzenquotient

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

ist die naheliegende Approximation, doch für kleines h werden $f(x+h)$ und $f(x)$ fast gleich gross, die Differenz ist daher stark verschmiert. Grosses h wiederum beschränkt die Genauigkeit, denn die Taylor-Entwicklung liefert für den Differenzenquotienten

$$f'(x) \approx \frac{f(x) + hf'(x) + h^2 f''(x)/2 + O(h^3) - f(x)}{h} = f'(x) + h \frac{f''(x)}{2} + O(h^2),$$

für nicht allzu kleines h wird der Einfluss des $f''(x)$ -Terms merklich. Es muss also angestrebt werden, genaue Resultate auch bei relativ grossem h bekommen zu können.

Eine erste Verbesserung kann mit dem symmetrischen Differenzenquotienten

$$f'(x) \approx \frac{f(x + \frac{h}{2}) - f(x - \frac{h}{2})}{h}$$

erreicht werden. Setzt man wieder die Taylor-Reihe ein, erhält man

$$\begin{aligned} f'(x) &\approx \frac{1}{h} \left(f(x) + \frac{h}{2} f'(x) + \frac{h^2}{8} f''(x) + \frac{h^3}{48} f'''(x) + O(h^4) \right. \\ &\quad \left. - \left(f(x) - \frac{h}{2} f'(x) + \frac{h^2}{8} f''(x) - \frac{h^3}{48} f'''(x) + O(h^4) \right) \right) \\ &= f'(x) + \frac{h^2}{24} f'''(x) + O(h^3). \end{aligned}$$

Danke der Symmetrie ist der $f''(x)$ -Term verschwunden, der Fehler ist nur noch von Ordnung h^2 .

Den symmetrischen Differenzenquotienten kann man auch als Wert eines Interpolationspolynoms ansehen. Die beiden Stützstellen $x_{0,1} = x \pm \frac{h}{2}$ definieren die lineare Interpolationsfunktion

$$\begin{aligned} p(x) &= f(x_0)l_0(x) + f(x_1)l_1(x) = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0} = -f(x_0)\frac{x - x_1}{h} + f(x_1)\frac{x - x_0}{h} \\ &= \frac{f(x_1) - f(x_0)}{h}x + \text{const.} \end{aligned}$$

Die symmetrische Differenz ist daher auch die Ableitung

$$p'(x) = \frac{f(x_1) - f(x_0)}{h}$$

des Interpolationspolynoms. Es liegt daher nahe, die Ableitung $f'(x)$ als Ableitung eines Interpolationspolynoms höherer Ordnung von $f(x)$ zu berechnen.

26.2 Die Ableitung des Interpolationspolynoms

In Kapitel 3 wurde für das Interpolationspolynom der Ausdruck

$$p(x) = \sum_{j=0}^n f(x_j)l_j(x) \quad \text{mit} \quad l_j(x) = \frac{(x - x_0)(x - x_1) \cdots (\widehat{x - x_j}) \cdots (x - x_n)}{(x_j - x_0)(x_j - x_1) \cdots (\widehat{x_j - x_j}) \cdots (x_j - x_n)}$$

gefunden. Seine Ableitung ist

$$p'(x) = \sum_{j=0}^n f(x_j)l'_j(x).$$

Die Ableitungen der speziellen Interpolationspolynome $l_j(x)$ kann man ebenfalls direkt berechnen:

$$l'_j(x) = \frac{1}{(x_j - x_0)(x_j - x_1) \cdots (\widehat{x_j - x_j}) \cdots (x_j - x_n)} \sum_{k \neq j} (x - x_0) \cdots (\widehat{x - x_k}) \cdots (\widehat{x - x_j}) \cdots (x - x_n). \quad (26.1)$$

Durch Einsetzen der Stützstellen lassen sich die $l_j(x)$ aus Formel (26.1) direkt berechnen, so dass man die Näherungsformel

$$f'(x) = \sum_{j=0}^n f(x_j) l'_j(x) \quad (26.2)$$

erhält.

Für äquidistante Stützstellen mit Abstand h sind alle Differenzen im Nenner von $l_j(x)$ Vielfache von h . Bei n Stützstellen geben die Faktoren vor dem Faktor $(x_j - x_0)$, der weggelassen muss, einen Beitrag $jh \cdot (j-1)h \cdots h$ im Nenner, die Faktoren danach liefern den Beitrag $h \cdot 2h \cdots (n-j)h$. Der Nenner ist daher

$$(-1)^j j! \cdot (n-j)! h^n,$$

die Ableitung kann jetzt als

$$l'_j(x) = \frac{(-1)^j}{j! \cdot (n-j)! h^n} \sum_{k \neq j} (x - x_0) \cdots (\widehat{x - x_k}) \cdots (\widehat{x - x_j}) \cdots (x - x_n). \quad (26.3)$$

geschrieben werden.

Die Interpolationspolynome $l_j(x)$ können leicht mit einem Computeralgebra system wie Maxima berechnet werden.

26.3 Ableitungsverfahren

Wir wenden die Formeln (26.2) und (26.1) zunächst auf den Fall zweier Stützstellen $x_0 = \xi - \frac{h}{2}$ und $x_1 = \xi + \frac{h}{2}$ an. Es folgt

$$\begin{aligned} l_0(x) &= -\frac{1}{h}(x - x_1) & l'_0(x) &= -\frac{1}{h} \cdot 1 & l'_0(\xi) &= -\frac{1}{h} \cdot 1 \\ l_1(x) &= \frac{1}{h}(x - x_0) & l'_1(x) &= \frac{1}{h} \cdot 1 & l'_1(\xi) &= \frac{1}{h} \cdot 1 \\ f'(x) &\approx \frac{1}{h}(-f(x_0) + f(x_1)) = \frac{f(x + \frac{h}{2}) - f(x - \frac{h}{2})}{2}, \end{aligned}$$

der bereits bekannte symmetrische Differenzenquotient.

Für $n = 2$ und die Stützstellen $x_0 = x - h$, $x_1 = x$ und $x_2 = x + h$ erhält man dagegen

$$\begin{aligned} l_0(x) &= \frac{1}{h \cdot 2h}(x - x_1)(x - x_2) & l'_0(x) &= \frac{1}{2h^2}(2x - x_2 - x_1) & l'_0(\xi) &= \frac{1}{2h^2}(2\xi - (\xi + h) - \xi) = -\frac{1}{2h} \\ l_1(x) &= \frac{1}{h^2}(x - x_0)(x - x_2) & l'_1(x) &= \frac{1}{h^2}(2x - x_0 - x_2) & l'_1(\xi) &= \frac{1}{h^2}(2\xi - (\xi - h) - (\xi + h)) = 0 \\ l_2(x) &= \frac{1}{2h \cdot h}(x - x_0)(x - x_1) & l'_2(x) &= \frac{1}{2h^2}(2x - x_0 - x_1) & l'_2(\xi) &= \frac{1}{2h^2}(2\xi - (\xi - h) - \xi) = \frac{1}{2h} \\ f'(\xi) &\approx \frac{1}{2h}(f(x_2) - f(x_0)) = \frac{f(\xi + h) - f(\xi - h)}{2h}, \end{aligned}$$

also wieder eine symmetrische Differenz. Die zusätzliche Stützstelle in der Mitte bringt keinen Genauigkeitsgewinn.

Stützstellen	-4	-3	-2	-1	0	1	2	3	4
3				$-\frac{1}{2h}$	0	$\frac{1}{2h}$			
5			$\frac{1}{12h}$	$-\frac{2}{3h}$	0	$\frac{2}{3h}$	$-\frac{1}{12h}$		
7		$\frac{1}{60h}$	$\frac{3}{20h}$	$\frac{3}{4h}$	0	$\frac{3}{4h}$	$\frac{3}{20h}$	$\frac{1}{60h}$	
9	$\frac{1}{280h}$	$-\frac{4}{105h}$	$\frac{1}{5h}$	$-\frac{4}{5h}$	0	$\frac{4}{5h}$	$-\frac{1}{5h}$	$\frac{4}{105h}$	$-\frac{1}{280h}$
Taylor			$\frac{1}{12h}$	$-\frac{8}{12h}$	0	$\frac{8}{12h}$	$-\frac{1}{12h}$		

Tabelle 26.1: Koeffizienten für die Berechnung der Ableitung aus den Stützstellen $x_j = x + jh$ mit $-n \leq j \leq n$. Die Koeffizienten, die aus der Taylorreihe gewonnen wurden sind in der letzten Zeile dargestellt, sie stimmen mit den Koeffizienten überein, die sich für fünf Stützstellen aus der Interpolation ergeben.

Stützstellen	-7	-5	-3	-1	1	3	5	7
2				$-\frac{1}{h}$	$\frac{1}{h}$			
4			$\frac{1}{24h}$	$-\frac{9}{8h}$	$\frac{9}{8h}$	$-\frac{1}{24h}$		
6		$-\frac{3}{640h}$	$\frac{25}{384h}$	$-\frac{75}{64h}$	$\frac{75}{64h}$	$-\frac{25}{384h}$	$\frac{3}{640h}$	
8	$\frac{5}{7168h}$	$-\frac{49}{5120h}$	$\frac{245}{3072h}$	$-\frac{1225}{1024h}$	$\frac{1225}{1024h}$	$-\frac{245}{3072h}$	$\frac{49}{5120h}$	$-\frac{5}{7168h}$

Tabelle 26.2: Koeffizienten für die Berechnung der Ableitung aus den Stützstellen $x + j\frac{h}{2}$.

Für grössere Anzahlen von Stützstellen wird die Berechnung etwas mühsam und kann mit Computeralgebra vereinfacht werden. In den Tabellen 26.1 und 26.2 sind die Koeffizienten zusammengestellt, die sich für eine ungerade bzw. gerade Anzahl von Stützstellen zusammengestellt.

In der letzten Zeile von Tabelle 26.1 sind auch die Koeffizienten des in Kapitel 25 aus dem Taylor-Polynom abgeleiteten Verfahrens aufgelistet. Nicht ganz überraschend stimmen Sie mit dem Verfahren überein, welches hier aus dem Interpolationspolynom abgeleitet wurde.

26.4 Fourier-Spektrum der Ableitung

Die Fourier-Transformation der Ableitung einer Funktion ist

$$\mathcal{F} \frac{df}{dx} = i\omega \mathcal{F} f.$$

Dies gibt uns eine Möglichkeit, die Qualität der Ableitungsmethoden von Abschnitt 26.3 zu messen. Diese Berechnungsmethode ist eine Faltung mit der Folge der Ableitungskoeffizienten. Im Fre-

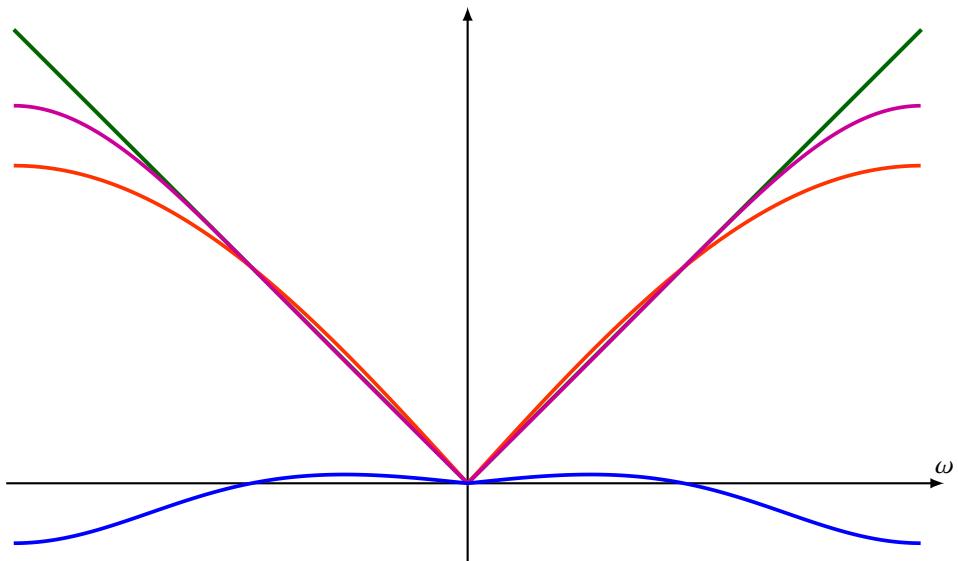


Abbildung 26.1: Fourier-Transformierte der Koeffizientenfolge für die Ableitung basierend auf dem Interpolationspolynom.

quenzbereich wird daraus eine Multiplikation mit der Fourier-Transformierten. Der absolute Betrag der diskrete Fourier-Transformierten der Ableitungskoeffizienten müsste also eine Betragsfunktion ergeben.

Die Resultate der Rechnung sind in Abbildung 26.1 dargestellt. Die Betragsfunktion ist grün dargestellt, der gewöhnliche Differenzenquotient in hellrot. Die pinke Kurve zeigt das Ableitungsverfahren, welches acht Stützstellen verwendet. Ganz offensichtlich wird dadurch das ideale Spektrum der Ableitung besser approximiert. Man kann allerdings auch erkennen, dass das Verfahren höherer Ordnung die hohen Frequenzen verstärkt, es ist also anfälliger auf Rauschen.

Index

2. Newtonsches Gesetz, 102
3D, 14
45°-Gerade, 30
0, 14
- Abel, Niels Henrik, 47, 440
Abhängigkeit von Parametern, 146
Ableitung, 40, 54, 59, 63, 66, 78, 80, 81, 139, 188, 317
dritte, 33
nach der Anfangsbedingung, 143
Ableitung eines Polynoms, 61
Ableitung, numerische, 463, 477
Ableitungsmatrix, 57
Abschluss, 209
Abschätzung, 29, 132
absoluter Fehler, 12
Abstiegsrichtung, 454
Adams, John Crouch, 157
Adams-Bashforth-Verfahren, 157
Addition, 10, 24
schriftliche, 10
Additionstheorem, 86
Adressierung, 10
relative, 10
Ähnlichkeitstransformation, 440
Akkumulation, 25
Aktivierungsfunktion, 465
Algorithmus, 5, 11
All pole model, 393
analytische Lösung, 6
Anfangsbedingung, 6, 160
Anfangsgeschwindigkeit, 142, 165, 166
Anfangsschätzung, 63
Anfangswerte, 141
Anfangswertproblem, 141, 168
angeregte Van der Pol-Gleichung, 304
Ankunftsrichtung, 108
- A-orthogonal, 454
A-Orthogonalität, 454
Approximant, 381
Approximation, 5, 38, 40, 53, 65, 85, 89, 120, 122, 133, 159, 223, 247
linear, 29, 147
Approximation durch Polynome, 248
Approximationsfehler, 5
äquidistante Stützstellen, 89
 $\arctan(x)$, 264, 375
 \tan^{-1} , 375
Arcussinus, 166
Arithmetik-Einheit, 8
Artillerie, 169
assoziativ, 7
Assoziativgesetz, 7
Atom, 264, 356
Audiosignal, 390
Ausbreitungsgeschwindigkeit, 234
Ausbuchtung, 446
Ausgangsimpedanz, 386
Ausklammern, 59
Auslöschung, 20, 22, 40, 45, 46, 63, 198, 279
Auswertepunkte, 121
Auswertungen, 39
AVR, 14
- Backpropagation, 464
Bahn, 142, 165
Bahnkrümmung, 106
Ball, 142, 165
Ballistik, 47
Ballwurf, 142
Band, 185
Bandmatrix, 185
baryzentrische Formel, 96
Basis, 8, 12
 bc , 20

- Bernoulli-Polynome, 128
Bernstein-Polynom, 111
Beschleunigung, 134, 140, 323, 344
Bestapproximation, 374
beste Approximation, 374
Bewegung, 142
bias, 465
Bibliothek, 8, 9, 171
Bifurkation, 268
Bifurkationsdiagramm, 268
Bildverarbeitung, 463
binary16, 14
Binomialkoeffizient, 195
binär, 12
Binärdarstellung, 21
Binärzahl, 9
Black-Box, 48
Blockmatrix, 194, 255
Bouhafs-Keller, Benjamin, 264, 363
Bra-Ket-Notation, 357
Bromwich-Kontur, 331
Bruch, 47, 70
Bruchzahl, 8
Bucher, Daniel, 263, 343
Burgers, 211, 238, 264
Burgers, Gleichung von, 403
butterfly effect, 308
Bézier-Kurve, 6, 105, 106, 108, 115
 höhere Ordnung, 110
 kubisch, 108
- C, 22, 161
C++, 14, 20, 40, 50
Cardano, Gerolamo, 47
Cattaneo, Manuel, 263, 301
Cauchy-Folge, 49
CG-Algorithmus, 451
Chaos, 268, 306
chaotisch, 263, 265, 301
charakteristisches Polynom, 194, 355, 440, 441
Chebyshev-Polynome, 292
Compiler, 9, 14
computational mode, 234, 237, 409
 numerische Anregung, 237
Computergraphik, 105
 $\cosh(x)$, 167
 $\cos(x)$, 44
- Courant-Friedrichs-Lowy-Kriterium, 407
Crank-Nicholson-Verfahren, 226
 c_W , 344
- ∂A , 209
dal Ferro, Scipione, 47
de Casteljau, Paul, 107
de l'Hospital, 44
Definitionsgebiet, 208, 209
Deflation, 60, 62
Deformation, 63
deformiert, 63
denormalisiert, 13
denormalisierte Zahl, 14, 15
Denormalisierung, 13–15
Determinante, 79, 197, 440
Dezimalzahl, 9
Diagonale, 185
Diagonalform, 193, 201, 440
diagonalisierbar, 193
Diagonalisierung, 201
Diagonalmatrix, 194
Dichte, 6, 240, 404
Differentialgleichung, 139, 315, 379
 explizite Form, 139
 gewöhnlich, 139, 207
 gewöhnliche, 6, 119, 263
 implizite Form, 139
 partiell, 185, 207, 208
 partielle, 1, 6
 Van der Pol-, 301
 Vektor-, 140
Differentialgleichung, partielle, 417
Differenzengleichung, 159
Differenzenquotient, 38, 238, 477
differenzierbare Funktion, 26
Dirichlet-Randbedingung, 213, 215
Diskretisation, 6, 201
Diskretisierung, 405
Divergenz, 240
Divergenzsatz von Gauss, 241
Division, 11, 60
Doppelfakultät, 279
Doppelpendel, 308
doppelte Nullstelle, 61
`double`, 9, 14
Drehmatrix, 201

- Drehung, 201
Dreieck, 54
Dreiecksmatrix, 198, 200
dritte Ableitung, 33
Dämpfungskonstante, 303
- e*, 24
ebene Kurve, 79
Echtzeitdatenverarbeitung, 391
Effekte
 numerische, 20
eig., 439
Eigenbasis, 193, 358
Eigenfunktion, 356
Eigenvektor, 439
Eigenwert, 439
Eigenwertproblem, 6, 179, 247, 355
einbetten, 326
Einbettung, 326
Eindeutigkeit, 141
Eingangsimpedanz, 386
Einheitsdreieck, 421
Einheitskreis, 115, 124, 393
Einheitsmatrix, 190, 255
Einheitsvektor, 198
Einschrittverfahren, 6, 151, 156
Einschwingen, 269
elektrisch, 207
elektromagnetisches Feld, 6
elektronische Schaltung, 329
Elimination, 133
elliptisch, 212
elliptische partielle Differentialgleichung, 212
Elsener, Patrick, 263, 275
Endgeschwindigkeit, 165
endliche Impulsantwort, 385
Energie, 222
 kinetisch, 166
entartet, 358, 360
Entartung, 359
entgegengesetzte Zahl, 10
Epidemie, 315
Epoche, 464
 ε , 44
 ε , 13, 50
 $\text{erfc}(x)$, 22
 $\text{erf}(x)$, 21, 62
- Euler, Leonhard, 99, 463
Euler-Lagrange-Differentialgleichung, 99
Euler-Lagrange-Gleichung, 102
Euler-Maclaurin-Summenformel, 125, 129, 130
Euler-Maclaurinsche Summenformel, 130
Euler-Schritt, 153
Euler-Verfahren, 148, 150, 155, 224, 228, 234, 236, 301
 verbessert, 153, 154
Existenz, 141
Experiment, 20
explizite Form einer Differentialgleichung, 139
Exponent, 12, 21, 50
Exponentiafaktor, 12
Exponentialfunktion, 14, 16, 22, 24, 47, 379
Exponentialreihe, 22–24
exponentiell, 227, 238
exponentiell schneller, 36
exponentielles Wachstum, 265
Extrapolation, 390
- Faktorisierung, 61
Fassregel, Kepler, 135
Fehlberg, 326
Fehler, 28, 69, 80, 85, 113, 121, 133, 150, 295
 relativer, 12, 15
 Trapezformel, 130
Fehler des Approximationspolynoms, 75
Fehler eliminieren, 35
Fehlerabschätzung, 75, 80, 82, 471
Fehlerformel, 296, 298
Fehlerfunktion, 21, 62, 464
Fehlerfunktion, komplementäre, 22
Fehlergesetz, 35
Fehlerpeak, 390
Fehlerschätzung, 326
Feigenbaum, Mitchell, 274
Feigenbaum-Konstante, 273
Feld, 207
 elektromagnetisches, 6
Feldeffekttransistor, 302
Feldlinien, 417
Festkommadarstellung, 21
Festkommazahl, 10
Filter, 238
Filterentwicklung, 390
finite Differenzen, 6, 185

- Finite Differenzen Methode, 463
finite Elemente, 6, 264, 417
finite impulse response, 385
finite Volumina, 6, 238, 242
Fior, Antonio Maria, 47
FIR, 385
Fixpunkt, 26, 39, 54, 269
Fixpunktiteration, 187
`float`, 9, 12, 14, 50
Floatingpoint, 8
Fluid, 239
Fluidynamik, 1
Flussintegral, 241, 246
Fourier-Transformation, 477, 480
Fraktal, 268
Francis, John, 264, 439
Francis-Algorithmus, 179, 203, 439
Francis-Iteration, 445
Francis-Iteration der Ordnung n , 447
Frequenzbereich, 332, 384
Frequenzgang, 385
Fritsche, Reto, 263, 323
 $f'''(x)$, 33
Funktion, 5, 139
 differenzierbar, 26
 gebrochen rational, 377
 hyperbolisch, 168
 linear, 70
 matrixwertig, 55
 trigonometrisch, 113
 trigonometrische, 14
Funktional, 243, 244, 246
Funktionsauswertungen, 39

Galliani, Niccolò, 263, 301
Ganzzahltyp, 8
Gauss
 Divergenzsatz, 241
 Gauss, Carl Friedrich, 179
 Gauss-Algorithmus, 6, 72, 179, 180, 183, 184, 187, 255
 Gauss-Chebyshev-Formel, 295
 Gauss-Laguerre-Formel, 296
 Gauss-Legendre-Formel, 295
 Gauss-Quadratur, 135, 263, 283
 Gauss-Quadraturformel, 292
 Gauss-Seidel-Algorithmus, 187
 Gauss-Seidel-Iteration, 186
 Gauss-Seidel-Verfahren, 190
 Gauss-Tableau, 180
 Gebiet, 209
 gebrochen rationale Funktion, 377
 Gedankenexperiment, 468
 Gelfand
 Satz von, 195
 Gelfand-Radius, 189, 193
 Genauigkeit, 154, 155, 323
 Genauigkeitsverlust, 24, 25
 Gerade, 70
 Geschwindigkeit, 99, 105, 140, 239, 323
 Geschwindigkeitsvektor, 105, 116
 Gewicht, 51, 97, 289, 465
 Gewichtskraft, 166
 Gewichtung, 289
 gewöhnliche Differentialgleichung, 6, 139, 207
 Github-Repository, 1
 Gitter, 215
 Gitterkonstante, 215, 223
 Gitterpunkte, 238
 Givens-Rotation, 201, 431, 434, 442
 Gleichung, 5
 kubische, 47
 von Burgers, 211
 Gleichung von Burgers, 264, 403
 Gleichungssystem
 für das Interpolationspolynom, 72
 iteratives Verfahren, 6
 linear, 6, 186
 Gleichungssystem, lineares, 1
 Gleitkommadarstellung, 21
 Gleitkommazahl, 12
 $\text{GL}_k(\mathbb{R})$, 55
 GNU GMP, 8, 16
 GNU MPFR, 16, 18
 GNU Scientific Library, 280
 GNU scientific library, 161
 GNU-Compiler, 14
 GPU, 14
 Grab, Tobias, 264, 439
 Gradient, 100, 253, 454, 463
 Gradient Descent, 452
 Gradientabstieg, 466
 stochastischer, 466
 Gram-Schmidt-Orthonormalisierung, 456

- Gram-Schmidt-Prozess, 198, 201
Gram-Schmidt-Verfahren, 432
Graph, 29, 30, 70
Graphikkarte, 14
Graphikprozessor, 105
graphische Analyse einer Iterationsfolge, 30
Gravitationskraft, 344
Gravitation, 349
Gravitationsfeld, 330
Gravitationszentrum, 323
Green
 Satz von, 241
Grenzwert, 193
Grenzzyklus, 303
Grenzübergang, 58
GSL, 161
gut konditioniert, 44
- $H_0(x)$, 94
 $h_0(x)$, 94
 $H_0^1(x)$, 94
 $h_0^1(x)$, 94
 $H_1(x)$, 94
 $h_1(x)$, 94
 $H_1^1(x)$, 94
 $h_1^1(x)$, 94
Hadamard-Produkt, 468
Haftriebung, 106
Halbierung, 133
Halbwinkelformel, 45, 46
Hamilton-Operator, 356
Hardware, 26
Hardwarebeschleunigung, 15
harmonische Reihe, 25
Hermite-Interpolation, 89, 106, 108
Hermite-Interpolationspolynom, 89, 105, 114
 zweite Ableitung, 95
Hermite-Interpolationsproblem, 89
 spezielles, 92
Hermite-Polynome, 292
Hessenberg-Form, obere, 441
hidden layer, 472
Hochpräzisionsbibliothek, 16
höhere Ableitungen, 142
homogen, 303
homogene Randbedingung, 227
Homotopie, 63
- Homotopie-Verfahren, 65
Horizontalgeschwindigkeit, 165
Horner-Schema, 59, 61
Householder-Transformation, 442
hyperbolisch, 212
hyperbolische Funktionen, 168
hyperbolische partielle Differentialgleichung, 212
Hyperkugel, 460
Hypotenuse, 54
Höchstnenner, 374
- $\bar{I}(D)$, 121
 $\underline{I}(D)$, 121
IEEE 754, 10, 13, 16, 18, 24
IEEE 754, 14
IIR, 385
Impedanz, 386
Implementation, 16
implizite Form einer Differentialgleichung, 139
implizites Verfahren, 411
Impulsantwort, 385
Impulsantwort, endliche, 385
impulse reponse
 fininite, 385
 infininite, 385
Indizierung, 10
Induktion, 86
Induktion, vollständige, 81
Induktionsschritt, 82
Induktionsvoraussetzung, 81
Induktivität, 385
Infizierte, 315
Ingenieurwissenschaften, 377
inifinite impulse response, 385
Inkrementfunktion, 152, 156, 317
Inneres, 209
instabil, 5, 90, 234, 263, 393
instabile Lösung, 234
Instabilität, 1, 8, 40, 43, 275
 numerisch, 41
Integral, 6, 25, 41, 119, 133
Integralform, 240
Integralefunktion, 324
Integralprinzip, 100, 102, 243
Integrand, 47
Integration, 119
 partiell, 101, 125, 244

- Integration, numerische, 283
Integrationskonstante, 128
Integrationsreihenfolge, 245
Interpolation, 390, 477
 linear, 69
Interpolationsfunktion, 70
Interpolationspolynom, 72, 82, 113, 477
 spezielles, 72
Intervallhalbierung, 48, 53, 75
Intervallschachtelung, 48
Inverse, 197
 einer Blockmatrix, 255
Inversionsformel, Riemann-, 331
irrationale Zahl, 368
Iteration, 5, 26, 40, 46, 187
Iterationsfolge, 26, 29, 30, 40, 57, 63
 graphische Konvergenzanalyse, 30
 Konvergenzkriterium, 31
 lineare Konvergenz, 37
 quadratische Konvergenz, 37
Iterationsformel, 53, 187, 190
Iterationsverfahren, 189
iterativ, 6
iterative Methode, 355
- Jacobi-Matrix, 143, 171
 numerische Lösung, 145
Jacobi-Verfahren, 179, 189
Jordan-Block, 194
jordansche Normalform, 440
- Kahan-Matrix, 197
Kahan-Summation, 25
Kapazität, 385
Kaskade von Periodenverdoppelungen, 273
Kennlinie, 302
Kepler-Gleichung, 64, 356
Keplersche Fassregel, 135
Kettenbruch, 264, 363
 einfach, 364
 regulär, 364
Kettenlinie, 166
Kettenregel, 144, 475
kinetische Energie, 99, 166
Kirchhoffsches Gesetz, 302
Kistler, Thomas, 263, 343
Klassifikation
 partielle Differentialgleichung, 213
- kleinste darstellbare Zahl, 13
kleinsten Quadrate Methode der, 431
Knotenvariable, 218
Koeffizient, 72
Koeffizientenmatrix, 186
Koeffizientenvektor, 248
Koeffizientenvergleich, 380
Komet, 330
komplementäre Fehlerfunktion, 22
komplex, 193
Kompression, 390
Kondensator, 302
Kondition, 43, 451
 schlecht, 197
konditioniert
 gut, 44
 schlecht, 44
Konditionszahl, 196, 197, 460
konjugierte Gradienten, 190, 451
Kontinuitätsgleichung, 238, 239
Kontrollpunkt, 108, 109, 115
Kontur von Talbot, 333
Kontur, Bromwich-, 331
Konvention
 für denormalisierte Zahlen, 13
konvergent
 linear, 36
 quadratisch, 36
Konvergenz, 26, 40, 53, 125, 132, 133, 188, 319, 460
 linear, 35, 37
 quadratisch, 36, 37, 54
Konvergenzbedingung, 186, 188
Konvergenzbeschleunigung, 37, 125, 134
Konvergenzgeschwindigkeit, 33, 35, 50, 65, 192
 Intervalhalbierung, 50
Konvergenzkriterium, 31, 32, 193, 225
Korn, 120
Kraft, 99, 166
Kreisbogen, 331, 376
kritischer Punkt, 306
Kryptographie, 8
Krümmung, 48, 323
kubische Bézier-Kurve, 108
kubische Gleichung, 47
Kurve, 105
Kurve, ebene, 79

- l*-Rekursion, 277
- Ladung, 207
- Ladungsverteilung, 212
- Lagrange-Funktion, 99
- Lagrange-Interpolationspolynom, 82, 89, 114
- Lagrange-Multiplikator, 253
- Lagrange-Restglied, 80
- Lagrange-Restterm, 80
- Laguerre-Polynome, 292
- Langzeitvorhersage, 263
- Laplace-Inverse, 263, 332
- Laplace-Inversion, 331
- Laplace-Operator, 208, 219
- Laplace-Transformation, 263, 331, 332, 385
- Laufzeit, 8, 9, 179
- Laufzeitkomplexität, 179
- Le Verrier, Urbain, 157
- Leap-Frog Methode, 407
- learning rate, 466
- learning, supervised, 464
- Least-Squares, 264, 391, 431
- Least-Squares-Problem, 252
- Legendre-Gleichung, 276
- Legendre-Polynom, 263
- Legendre-Polynome, 275, 292
- Legendre-Polynome, zugeordnete, 275
- Leonhard Euler, 99
- Lernen, 463, 466
- Lernprozess, 464
- Lewis Fry Richardson, 234
- limit cycle, 303
- <limits>, 14
- linear konvergent, 36
- linear time-invariant system, 384
- Linear zeitinvariantes System, 384
- lineare
 - Algebra, 7, 179, 195
 - Approximation, 29, 147
 - Funktion, 70
 - Interpolation, 69
 - Konvergenz, 35, 37
- lineares
 - Gleichungssystem, 6, 179
 - lineares Gleichungssystem, 1, 142, 186
 - Linearkombination, 73
 - Linux, 20
 - Lipschitz-Bedingung
- untere, 52
- Lipschitz-Eigenschaft, 119
- Lipschitz-stetig, 141
- Lipschitz-Verfahren, 51
- Lissajous-Figur, 85
- $l_j(x)$, 73
- [L/M], 381
- logarithmische Skala, 23
- Logarithmus, 23, 28, 43, 47, 376
- Logiktabelle, 472
- logistische Funktion
 - mit $\lambda = 3, 28$
- logistische Gleichung, 32, 263, 265
- long double, 9, 14, 63
- Lorenz, Edward, 308
- Loss-Funktion, 464
- lsode, 161
- LTI, 384
- Luftdruck, 343
- Luftwiderstand, 343, 356
- $l(x)$, 73
- LZI, 384
- Lösung
 - analytische, 6
 - einer Gleichung, 5
- Lösung in geschlossener Form, 139
- Lösungsformel, 47
- Lösungskurve, 165
- m*-Rekursion, 277
- magnetisch, 207
- Mandelbrotmenge, 268, 271
- Mantisse, 12, 14, 21, 24, 50
- Mars, 330
- Marti, Fabio Daniel, 263, 315
- Maschengleichung, 302
- Masse, 99, 165, 240
- Massendichte, 166
- Materie, 239
- Materiefluss, 241
- MATLAB, 360, 439
- Matrix
 - orthogonal, 201
 - symmetrisch, 179
 - tridiagonal, 183
- Matrixform, 223
- Matrixinvertierung, 185

- Matrixprodukt, 144
Matrixzerlegung, 264
Maupertuis-Prinzip, 99
Maxima, 480
Maxwellsche Gleichungen, 207
mehrfacher Eigenwert, 360
Mehrschrittverfahren, 6, 139
Menge
 offen, 209
Methode der kleinsten Quadrate, 431
Microcontroller, 8, 14
Minimalpolynom, 440
Minimalprinzip, 248
Minimalproblem, 190, 242
 nichtlinear, 253
Minimierungsproblem, 452
Mittel
 gewichtet, 51, 97
Mittellinie, 122
Mittelpunktformel, 6, 120, 136
Mittelpunktregel, 121
Mittelpunktsregel, 283
Mittelschicht, 472
Mittelwertsatz, 78, 81, 215
 der Differentialrechnung, 78
Mittelwertsatz in \mathbb{R}^n , 80
MNIST-Datensatz, 475
Molekül, 264
Monte-Carlo-Methode, 135
Müller, Andreas, 397, 477
Multicore CPU, 12, 15
Multiple Precision Floatingpoint Reliable Library, 18
Multiplikation, 11
Multiprecision-Library, 8
Mustererkennung, 463
- N, 7
 ∇ , 207
Nachkommateil, 11
Navier-Stokes-Gleichung, 242
Navier-Stokes-Gleichungen, 403
Nebendiagonale, 180
Nennerpolynom, 381
Neptun, 157
Netzwerk, 385
Neumann-Randbedingung, 214
- Neuron, 465
neuronales Netzwerk, 463, 464
Newton
 drittes Gesetz von, 166
Newton, Isaac, 463
Newton-Cotes-Formel, 283
Newton-Verfahren, 53, 54, 58, 63, 65, 147, 168, 169, 190
 für Vektorgleichungen, 56
 in \mathbb{R}^n , 55
newtonisches Fluid, 404
Newtonsches Gesetz, 165
Newtonsches Gesetz, zweites, 102
nicht entartet, 358
nichtlinear, 301, 403
Nichtlineares Minimalproblem, 253
Nichtlinearität, 403
Niveaulinien, 460
Norm, 188
Normalableitung, 214
Normalform, jordansche, 440
Normalverteilung, 21, 62, 89, 119, 324
Normalverteilungsfunktion, 63
Null, 14
Nullpolynom, 72
Nullstelle, 48, 53, 60, 66, 75, 85
 doppelte, 61
Numerik, 1
numerische Ableitung, 477
numerische Effekte, 20
numerische Instabilität, 1, 8, 40, 41, 43
numerische Integration, 283
NumPy, 360
numpy, 299
Näherungsgesetz, 374
Näherungsverfahren, 148
- obere Hessenberg-Form, 441
obere Riemann-Summe, 120
Octave, 7, 20, 124, 161
offene Menge, 209
Offset, 465
OpenFOAM, 1
Operationsverstärker, 302, 385
Optimierungsproblem, 98
Optimizer, 466
Ordnung, 210

- einer Differentialgleichung, 139
einer partiellen Differentialgleichung, 210
Reduktion der, 140
orthogonale Matrix, 201
orthogonale Polynome, 291
orthonormiert, 198
Ortsvektor, 140
Oszillation, 84, 86, 89
Oszillator, Van der Pol, 301
- Padé-Approximation, 377
Padé-Approximation, 264
Parabel, 32, 453
parabolisch, 212
parabolische partielle Differentialgleichung, 212
Parallelisierbarkeit, 8
Parameter, 6
Parameterdarstellung, 105
Parametrisierung, 116
partielle Differentialgleichung, 1, 6, 185, 207, 208, 403, 417
 Ordnung, 210
partielle Integration, 101, 125, 244
Periodenverdoppelung, 263, 268
perturbation, 355
Phasenverlauf, 385
Physik, 99, 165, 377
 π , 9, 20, 24
Picard-Lindelöf, Satz von, 141
PID-Glied, 385
Pivotdivision, 185, 255
Pivotelement, 180
Planet, 356
Poisson-Problem, 183, 220
Polarkoordinaten, 57, 251
Polarwinkel, 250
Polygonzug, 70
Polynom, 5, 58, 69, 126, 264
 charakteristisch, 194
 Nullstellen, 58
 trigonometrisch, 85
Polynomansatz, 276
Polynomdivision, 42, 60
Population, 265
Positionsdarstellung, 8
positiv definit, 190
Potential, 99
- potentielle Energie, 99
Potenzansatz, 159
Potenzfunktion, 50
Potenzmethode, 355
Potenzreihe, 32
Potenzreihenansatz, 379
Prinzip der kleinsten Wirkung, 99
Produktregel, 61
Programm, 163
Projektion, 358
Prozessor, 10
Präzision, 16
Punktmasse, 323
Python, 299, 304, 335, 345, 436, 460
PyTorch Framework, 472
- \mathbb{Q} , 7
 q_{factor} , 325
QR-Algorithmus, 264
 implizit geshifteter, 439
QR-Zerlegung, 6, 179, 197, 199, 431
Quadrant, 124, 244
quadratisch konvergent, 36
quadratische
 Funktion, 216
 Gleichung, 28
 Konvergenz, 5, 36, 37, 54
 Verfahren, 152
quadratische Ableitung, 454
quadratischer Fehler, 391
quadratisches Minimalproblem, 250, 252
Quadratsumme, 244
Quadratur, 283, 376
Quadratwurzel, 40, 55, 237
Quantenmechanik, 264, 356
quasilinear, 211
Quotient
 Polynom, 62
- \mathbb{R} , 7
Raketenmotor, 1
Rand, 209
Randbedingung, 100, 213, 221
 Dirichlet, 213
 homogen, 227
 Neumann, 214
Randwertproblem, 6, 139, 142, 163
Rang, 440

- rationale Zahl, 11
 Rauschen, 471, 472
 Rayleigh-Quotienten-Shift, 447
 Rechenaufwand, 154
 Rechenleistung, 323
 Rechenregeln, 7
 Rechenzeit, 185
 Rechsteiner, Joël, 264, 417
 Reduktion der Ordnung, 140, 303, 345
 Regel von Sarrus, 442
 reibungsfrei, 403
 Reihe
 der Exponentialfunktion, 22
 harmonische, 25
 Reihenfolge, 24
 abnehmend, 24
 aufsteigend, 24
 Rekursion, 237
 Rekursionsbeziehung, 276
 Rekursionsformel, 42, 43, 45, 86, 127, 148, 159, 292
 relativer Fehler, 15, 150
 Relaxation, 191
 Renaissance, 47
 Renda, Cédric, 264, 377
 Residuum, 190, 453
 Resonanzfrequenz, 303
 Restterm, 471
 Resultatvergleich, 299
 ρ_{Luft} , 344
 Richardson, Lewis Fry, 234, 404, 408
 Richardson-Verfahren, 190
 Richtungsableitung, 99, 252
 Richtungsfeld, 139
 Riemann-Integral, 119–121
 Riemann-Inversionsformel, 331
 Riemann-Summe, 120
 Risch-Algorithmus, 47
 RLC, 302
 $R_{[L/M]}$, 381
 Robert-Asselin-Time-Filter, 409
 Roboter, 106
 Rolle, Satz von, 75
 Romberg-Algorithmus, 132, 133
 Romberg-Beschleunigung, 336
 Romberg-Integration, 40
 Romberg-Verfahren, 135
 Rundung, 14
 Rundungsfehler, 5, 7, 40, 44, 237, 263
 Rundungsregel, 14
 Runga-Kutta-Verfahren, 301
 Runge, 84
 Runge-Kutta-Fehlberg-Verfahren, 162, 163
 Runge-Kutta-Prince-Dormand-Verfahren, 162
 Runge-Kutta-Verfahren, 6, 139, 154–156, 162, 305, 316, 344
 vereinfacht, 154
 Runges Phänomen, 84, 96, 113
 Rücktransformation, 332
 Rückwärtsdifferenz, 215, 224
 Rückwärtseinsetzen, 182, 185, 255
 Rückwärtssiteration, 42, 43
 Rückwärtsverfahren, 228
 Samen, 449
 Sarrus, Regel von, 442
 Satellit, 263
 Satellitenbahnen, 343, 349
 Saturn, 157
 Satz von Euler-Lagrange, 370
 Satz von Gelfand, 195
 Satz von Green, 241
 Satz von Picard-Lindelöf, 141
 Satz von Rolle, 75, 79
 Satz von Stokes, 241
 Satz von Stone-Weierstrasse, 69
 Schablone, 218
 Schaltung, elektronisch, 329
 Schiebeoperation, 12
 Schiessverfahren, 169
 schlecht konditioniert, 44
 Schmid, Michael, 264, 403
 Schmid, Mike, 263, 283
 Schneeberger, Michael Arthur, 263, 265
 Schnittpunkt, 29, 30, 51
 schriftliche Addition, 10
 Schrittänge, 129, 323
 Schrittängensteuerung, 313, 323
 Schrittweite, 124, 133, 147, 150, 156, 163, 215, 263
 Schrödinger-Gleichung, 356
 Schwerkraft, 165
 Schwingkreis, 302
 Schwingungsdifferentialgleichung, 145

- Schätzwert, 65
SciPy, 345
Seil, 166
Sekante, 63, 75
Sekantenverfahren, 51, 54, 58, 63
selbstadjungiert, 356
Selbstähnlichkeit, 268
Separation, 167
Serienenschwingkreis, 302
Sherman-Morrison-Woodbury-Formel, 183
Shift, 446
 Rayleigh-Quotienten-, 447
 Wilkinson-, 447
Shift-Strategie, 447
Signal, 390
Signalmodellierung, 390
Signalverarbeitung, 384, 463
signifikant, 12, 21
similarity transformation, 440
Simpsonsche Formel, 136
Simpsonsche Regel, 135
Simulation, 417
Simulator, 329
Singularität, 331
 $\sinh(x)$, 167
Sinusfunktion, 113, 132
 $\sin(x)$, 113
Skalarprodukt, 198, 252, 358
Skalarprodukt, verallgemeinertes, 454
Software, 6
Softwarebibliothek, 160
SOR, 190
Spaltenvektor, 146, 184
Spannung, 302
Speicherbedarf, 8
Speicherplatz, 15
Spektrallinie, 356
Spektralradius, 6, 188, 192, 227, 228
Spektrum, 264
spezielles Hermite-Interpolationsproblem, 92
spezielles Interpolationspolynom, 72
spherical harmonics, 276
Spice, 329
Spiegelung, 179, 198
Spiegelungsmatrix, 199
Spinnwebdiagramm, 269
Spline, 105
 Spline-Funktion, 98, 108
 Spline-Interpolation, 6, 98
 Spline-Interpolationsfunktion, 105, 180
 Sprachsynthese, 463
 Sprachübersetzung, 463
 Sprungantwort, 385
 Spur, 440
 $\sqrt{10}$, 20
 $\sqrt{2}$, 22
 Stab, 221
 stabil, 53, 275, 393
 Stabilität, 269, 275, 385
 Stammfunktion, 119, 128, 247
 Standardbasisvektor, 184
 Standardnormalverteilung, 21, 119, 324
 Standardnormalverteilungsdichte, 62, 89
 Startgeschwindigkeit, 343
 Startrichtung, 108
 Startwert, 26, 40, 43, 45, 54, 63
 Steigung, 48, 70
 Stokes
 Integralsatz, 241
 Stone-Weierstrass
 Satz von, 69
 Strahlen, 210
 Strahlensatz, 51
 Strom, 302
 Strömungsdynamik, 1
 Strömungsgeschwindigkeit, 239
 Strömungswiderstand, 344
 Stufen, 122
 Stypinski, Martin, 463
 Stypinsky, Martin, 264
 Störfunktion, 303
 Störungstheorie, 263, 343, 355
 stückweise linear, 72
 Stützstelle, 69
 Stützstellen, 72, 113
 äquidistant, 89
 Subtraktion, 10, 21
 successive overrelaxation, 190
 Suchrichtung, 453
 sukzessiv, 191
 Summationsreihenfolge, 24
 Summenformel, 125, 129, 283
 Summenformel, Euler-Maclaurin, 130
 Summierung, 24

- supervised learning, 464
Supremum-Norm, 83
Symbolmatrix, 212
Symmetrie, 128, 165
symmetrisch, 190, 203, 228, 356
symmetrische Differenz, 216, 478
symmetrische Matrix, 179, 355
system, linear time-invariant, 384
System, linear-zeitinvariante, 384
- Tableau, 183
Talbot, 333
Talbot-Kontur, 333
Tangens, 46
Tangente, 63, 75, 79
Tangentialvektor, 105
 $\tan(x)$, 46
Tartaglia, Niccolò, 47
Taylor, Brook, 463
Taylor-Approximation, 316
Taylor-Polynom, 80, 82, 143, 480
Taylor-Reihe, 16, 22, 23, 29, 36, 44, 57, 80, 89,
 150, 154, 156, 237, 263, 315, 377, 469,
 477
Teilchen, 99
Teilprodukte, 59
Teilreste, 60
Temperatur, 6
Temperaturgradient, 221
Temperaturverteilung, 221
Testschritt, 324
Thomas-Algorithmus, 182–184, 413
Thread, 15
Tiefpassfilter, 409
Tischhauser, Manuel, 264, 431
Tobler, Nicolas, 264, 355
Totzeit, 384
Trainieren, 466
Training, 463
Trapez, 122
Trapezformel, 129
 Fehler, 130
Trapezregel, 6, 40, 119, 122, 124, 126, 133, 136,
 283
Trapezsumme, 124, 129
Treiberprogramm, 16
Treppenlinie, 30
- tridiagonal, 180, 413
tridiagonale Matrix, 183
Tridiagonalmatrix, 441
trigonometrische Funktion, 14, 113
trigonometrisches Polynom, 85
Tschebyscheff-Polynom, 85, 86
Tschebyscheff-Stützstellen, 86, 89
Tunneldiode, 302
- Überrelaxation, 191
 $U_\delta(x_0)$, 208
Umgebung, 208
Umkehrfunktion, 62
Umlaufbahn, 356
unendlich, 15
unendlichdimensional, 100
universelle Eigenschaft, 273
untere Lipschitz-Bedingung, 52
untere Riemann-Summe, 120
Unterer, Raphael, 264, 451
Unterlauf, 15
Unterraum, 252
Unterteilung, 120
Uranus, 157
- Vakuumröhre, 301
Van der Pol, Balthasar, 301
Van der Pol-Differentialgleichung, 301
Van der Pol-Gleichung, 263
 angeregte, 304
Van der Pol-Oszillator, 301
vanishing gradient problem, 475
Variablentransformation, 21, 130
Variation, 243
Variationsrechnung, 99
Vektor-differentialgleichung, 140, 160
Vektorgleichung, 55
Vektorraum
 unendlichdimensional, 100
verallgemeinertes Skalarprodukt, 454
verbessertes Euler-Verfahren, 153, 154
vereinfachtes Runge-Kutta-Verfahren, 154
Verfahren k -ter Ordnung, 151
Verfeinerung, 121, 124
Verschachtelung, 271
Verschiebeoperation, 11
Verschmierung, 16, 22, 24
Verteilungsfunktion, 21, 63, 119, 324

- Vertikalgeschwindigkeit, 165
Verzweigung, 268
Verzögerung, 384
Viskosität, 404
vollständige Induktion, 81, 86, 181
Vorhersage, 404
Vorwärtsdifferenz, 215, 224, 236
Vorwärtsreduktion, 180, 181, 185
Vorzeichen, 10, 12, 21

Wachstumsrate, 316
Wahrscheinlichkeit, 21
Wahrscheinlichkeitsdichtefunktion, 324
Wasserstoff, 264
Weiss, Severin, 263, 331
Wellengleichung, 212, 264
Widerspruch, 101
Widerstand, 302, 385
Widerstandsfläche, 344
Wilkinson, James Hardy, 447
Wilkinson-Shift, 447
Wirkung, 99
Wirkungsintegral, 99
 w_j , 97
Wolfram Alpha, 263, 275
Wortlänge, 10
Wurzel, 14, 40, 55, 159
Wärmefluss, 221, 222
Wärmekapazität, 221
Wärmeleitfähigkeit, 221, 234
Wärmeleitung, 221
Wärmereservoir, 221

XOR, 472

 \mathbb{Z} , 7
Z-Transformation, 385
Zahl
 rationale, 11
Zahlensystem, 5, 7
Zehnerlogarithmus, 23
Zeilenoperation, 180, 183, 185
Zeilenvektor, 184, 253
Zeitaufwand, 163
Zeitbereich, 384
Zeitrichtung, 212
Zeitschritt, 224
Zentripetalkraft, 106

Zerlegungsverfahren, 189
Ziel, 142, 165
Zufallszahlgenerator, 449
Zweierkomplement, 10
Zweierpotenz, 11
Zweikörpermodell, 356
zweite Ableitung, 95
 des Hermite-Interpolationspolynoms, 95
Zwischenwertsatz, 48, 75
 für Nullstellen, 48
Zähler, 10, 163
Zählerpolynom, 383

Überlauf, 12, 15
Übertragungsfunktion, 386

