
Numerisches Programmieren

TUTOR SKRIPT SEIT SOMMERSEMESTER 2017

Autor
HENDRIK MÖLLER

Version vom:
1. MÄRZ 2023

Inhaltsverzeichnis

1	Vorwort	5
2	Legende	6
3	Liste der Notationen	7
4	Zahlendarstellung	9
4.1	Dezimal zu Binär	10
4.2	Binär zu Hex	11
4.3	Dezimalbrüche zu Binär	12
4.4	Maschinenzahlen	13
4.4.1	2-Komplement-Darstellung	13
4.4.2	Gleitkommazahlen	15
4.4.3	IEEE-Standard	17
4.4.4	Rundung	19
4.4.5	Maschinengenauigkeit	21
4.4.6	Absorption	22
4.4.7	Auslöschung	23
5	Kondition	24
6	Stabilität	26
7	Lineare Gleichungssysteme	30
7.1	Gauß-Elimination	30
7.2	Pivotisierung	32
7.3	LR-Zerlegung	33
7.4	Orthogonale Matrizen	38
7.5	QR-Zerlegung	40
7.5.1	Givens-Rotation	40
7.5.2	Weiterführung	43
7.6	Lineares Ausgleichsproblem	45
7.6.1	Lösung mit QR-Zerlegung	49
8	Matrixnorm	52
8.1	Euklidische Norm	52
8.2	Spektralnrm	52
8.3	Rechen Eigenschaften	53
8.4	Kondition einer Matrix	54
8.5	Kondition eines LGS	55

9	Interpolation	56
9.1	Polynominterpolation	56
9.1.1	Basisfunktionen	57
9.1.2	Lagrange-Basis	59
9.1.3	Interpolation nach Newton	61
9.1.4	Aitken-Neville	64
9.1.5	Fehlerabschätzung	67
9.2	Runge-Effekt	68
9.3	Lineare Interpolation	70
9.4	Hermite Interpolation	72
9.5	Interpolation mit Splines	78
10	Komplexe Zahlen	83
11	Frequenzanalyse	85
11.1	Diskrete Fourier-Transformation	87
11.2	Schnelle (Inverse) Fourier-Transformation	90
12	Quadratur	93
12.1	Klassische Quadratur	97
12.2	Restglied	99
12.3	Quadratur nach Romberg	100
12.4	Gauß Quadratur	104
13	Fixpunkte	108
13.1	Newton Verfahren	108
13.2	Eigenschaften	110
13.3	Banach'scher Fixpunktsatz	114
14	Fixpunktiteration (LGS)	116
14.1	Vektoriteration	116
14.1.1	Jacobi Verfahren	118
14.1.2	Gauß-Seidel Verfahren	119
14.2	Verfahren des steilsten Abstieges	121
15	Eigenwertproblem	124
15.1	Kondition	126
15.2	Rayleigh Quotient	127
15.3	Vektor-/Poweriteration	128
15.3.1	Direct Power Iteration	128
15.3.2	Shifted Power Iteration	129
15.3.3	Konvergenzrate	131
15.3.4	Inverse Power Iteration	132

16 Differentialgleichungen	133
16.1 Separation der Variablen	133
16.2 Verkürzte Version	134
16.3 Kleiner Trick	136
16.4 Kondition eines AWP's	138
16.5 (Explizite) Einschrittverfahren	139
16.5.1 Richtungsfelder	139
16.5.2 Explizites Euler-Verfahren	141
16.5.3 Heun-Verfahren	141
16.5.4 Runge-Kutta Verfahren	141
16.6 Diskretisierungsordnung	142
16.7 Implizites Euler-Verfahren	143
16.8 Wichtige Begriffe	145
16.9 Quadratur und AWP-Lösung	147
16.10 Mehrschrittverfahren	148
16.10.1 Mittelpunktsregel	148
17 Tipps für die Klausur	150
18 Typische Fehler	152
19 Abschlussworte	154

1 Vorwort

Dieses Dokument wurde von einem Tutor erstellt. Des Weiteren ist dies kein offizielles Dokument unterstützt oder in Relation mit irgendwelchen Lehrstühlen oder deren Mitarbeitern.

Insofern besteht keine Garantie auf **Vollständigkeit oder Korrektheit** der Inhalte.

Ich habe dieses Skript im Sommersemester 2017 angefangen zu schreiben und seitdem erweitert. Hier könnt ihr Erklärungen, Beispiele sowie Methodiken zu dem gesamten Lernstoff finden. Es soll der **Lernunterstützung und Vorbereitung** auf die Klausur dienen.

Wie sollte man es nutzen? Ich persönlich empfehle folgendes:

1. Skript durchlesen, Notizen rausschreiben und anhand davon Stoff verstehen.
2. Übungsaufgaben (wie z.B. in meinem Trainer) durchgehen und erlangtes Wissen überprüfen/vertiefen.
3. Fragen stellen, falls man etwas nicht verstanden hat.
4. Die Probeklausuren als finale Vorbereitung durchrechnen.
5. Richtige Klausur schreiben.
6. Profit. 😊

Die aktuellen Dokumente findest du immer auf meiner Webseite:

hendrik.fam-moe.de.

Nun wünsche ich frohes Lernen und viel Erfolg! 😊

Liebe Grüße
Hendrik

2 Legende

Beispiel

Beispiele aus Erklärungen und Methoden findet man in diesen Abschnitten.

Beispiel

Diese Box enthält ein Beispiel einer Beispielbox.

Erklärung

Diese Boxen enthalten Texte, die euch ein Thema näher bringen sollen. Verständnis ist das Ziel, weswegen diese Boxen mitunter die wichtigsten sind.

Wichtig

Diese Anmerkungen sollte man nicht überlesen da sie meist wichtige **ABER** und/oder Ausnahmen von Regeln enthalten.

Vorgehen

Hier stehen die Schritte einer Methode kurz und knapp beschrieben, um ein gewisses Problem zu lösen. Diese entsprechen in den meisten Fällen dem Schema X, um einen gewissen Aufgabentyp zu lösen.

Wichtig

Die Methoden sollten gut gekannt werden, aber ihr müsst die nicht unbedingt auswendig wissen. Verständnis ist wichtiger als Schema X!

Vertiefung

Vertiefungen, Beweise, Verweise und nennenswerte Fakten stehen in solchen Boxen drin. Der Inhalt könnte nicht prüfungsrelevant sein, aber immer interessant und wissenswert!

3 Liste der Notationen

Notation	Beschreibung
\mathbb{R}	Die reellen Zahlen.
\mathbb{N}	Die natürlichen Zahlen, Null enthaltend.
\mathbb{C}	Menge der komplexen Zahlen.
u_i	i . Eintrag des Vektors u .
\mathcal{O}	Big O Notation für Laufzeitklassen.
$ x $	Absoluter Wert von x .
$\ x\ _2$	Normiertes x ($\ell - 2$ Norm).
$f'(x)$	Die Funktion $f(x)$, einmal abgeleitet.
$f^{(d)}(x)$	Die d . Ableitung von $f(x)$.
A^T	Transponierte der Matrix A .
A^{-1}	Das Inverse der Matrix A .
$\text{diag}(A)$	Vektor mit den Diagonaleinträgen der Matrix A .
$\det(A)$	Die Determinante der Matrix A .
$\text{rd}(x)$	x , gerundet.
$y \stackrel{!}{=} x$	y soll gleich x sein.
$y := x$	y ist definiert als x .
$y \gg x$	y viel größer als x .
$y \ll x$	y deutlich kleiner als x .
$\exp(x)$	Andere Schreibweise für e^x .
$\text{Re}(z)$	Realteil einer komplexen Zahl.
$\text{Im}(z)$	Imaginärteil einer komplexen Zahl.
\bar{z}	Das komplex konjugierte einer komplexen Zahl z .
\tilde{a}	Eine durch Numerik verfälschte Variable a .

Notation	Beschreibung
n	Anzahl Stückstellen oder Anzahl Bits (je nach Kapitel).
f_{abs}	Absoluter Fehler.
f_{rel}	Relativer Fehler.
$\text{cond}()$	Die relative Konditionszahl.
ϵ_{Ma}	Maschinengenauigkeit (der maximal bei einer Rundung auftretende Fehler).
op	Beliebige, exakte Operation.
op_{M}	Eine beliebige, fehlerbehaftete Operation.
$\mathbb{1}$	Die Einheitsmatrix.
G_{φ}	Rotationsmatrix bezüglich des Winkels φ .
c_i	Koeffizient mit Index i .
$p(x)$	Interpolation (polynoms-) funktion.
L_j	Lagrange Basis.
H_i	Kubische Basispolynome.
H	Insgesamte Größe des zu interpolierenden / integrierenden Bereiches.
h	Länge eines (oder aller) Bereiche zwischen zwei Stützstellen.
Q_{T}	Die Trapezregel oder Keplersche Fassregel.
Q_{TS}	Die Trapezsumme.
Q_{S}	Simpsonregel.
Q_{SS}	Simpsonsumme.
DFT	Diskrete Fourier Transformation.
IDFT	Inverse diskrete Fourier Transformation.
FFT	Fast Fourier Transformation.
BFO	Butterfly Operator.
x^*	Fixpunkt.
$\Phi(x)$	Iterationsvorschrift einer iterativen Methode.
δt	Schrittweite eines Iterationsverfahrens.
$\lambda(A)$	Eigenwerte der Matrix A .
$\lambda_{\text{max}}(A)$	Der größte Eigenwert der Matrix A .
μ	Shift einer Power Iteration.

4 Zahlendarstellung

Erklärung

Zahlendarstellungen sind eine wichtige Grundlage der Numerik. Zunächst betrachten wir die Zahlenbasen, welche für uns relevant sind und wie wir sie übersetzen.

Die wichtigsten Zahlenbasen sind für uns:

- Basis 2, binäre Zahlen genannt,
- Basis 10, die Dezimalzahlen und
- Basis 16 oder auch Hex-Zahlen.

Falls eine Darstellung nicht hinreichend gekennzeichnet ist, schreibe ich eine 2 in den Index bei binären Zahlen, bei Hex-Zahlen eine 0x vor die Zahl oder eine 16 in den Index. Falls die Darstellung durch einen Hex-Buchstaben trivialerweise einer Hex-Zahl entspricht, lasse ich eine Kennzeichnung weg.

In der nächsten Tabelle lasse ich z.B. die Kennzeichnungen weg, da die erste Spalte die Basis der Zahlen angibt.

Basis	Beispiele			
Binär	01100	10001	11101	00011
Dezimal	12	17	29	03
Hex	0C	11	1D	03

Tabelle 1: Beispiele für alle Basen.

Ganzzahl	1	2	4	8	16	32	64	128	256	512	1024
Potenzschreibweise	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}

Tabelle 2: Die wichtigsten Zweierpotenzen.

Hex-Buchstabe	A	B	C	D	E	F
Dezimalwert	10	11	12	13	14	15

Tabelle 3: Wdh. Hex Buchstaben.

4.1 Dezimal zu Binär

Du hast beispielweise die Zahl 12 und möchtest diese binär darstellen.

Vorgehen

- Überlege dir, welche größte Zweierpotenz noch in deine Zahl reinpasst (Bei 12 wäre das 8, also 2^3)
- Subtrahiere deine Zahl durch diese Potenz und schreibe eine »1«.
- Geh jetzt jeweils durch alle kleineren Zweierpotenzen (bis 2^0) und subtrahiere deine Zahl durch diese, wenn möglich (Solang die Subtraktion nicht kleiner null wird)
- Immer wenn man die Zahl nicht subtrahieren kann (< 0), schreibe eine »0«, ansonsten eine »1«.

Beispiel

(von oben)

$$12 - 2^3 = 4 \rightarrow 1$$

$$4 - 2^2 = 0 \rightarrow 1$$

$$0 - 2^1 < 0 \rightarrow 0$$

$$0 - 2^0 < 0 \rightarrow 0$$

Somit ergibt sich (von oben nach unten) »1100₂«.

Beispiel

Umwandlung von 37:

$$37 - 2^5 = 5 \rightarrow 1$$

$$5 - 2^4 < 0 \rightarrow 0$$

$$5 - 2^3 < 0 \rightarrow 0$$

$$5 - 2^2 = 1 \rightarrow 1$$

$$1 - 2^1 < 0 \rightarrow 0$$

$$1 - 2^0 = 0 \rightarrow 1$$

Das Ergebnis ist »100101₂«.

4.2 Binär zu Hex

Vorgehen

Wenn wir eine binäre Zahl in eine hexadezimale Zahl umwandeln wollen, ergeben vier binäre Ziffern jeweils eine Hex-Ziffer der Hex-Zahl. Also geht man durch die binäre Zahl von rechts nach links, trennt sie in Viererpakete auf und wandelt dann jedes Paket einzeln um.

Beispiel

101010₂ zu Hex umwandeln (Das wäre 42 als Dezimalzahl):

Auftrennen in Viererpakete: **0010|1010**

Die letzten (rechtsten) vier Ziffern ergeben im Dezimalsystem 10. Dies entspricht in Hex einem »A« (siehe [Tabelle 3](#)). Das andere Päckchen entspricht einer 2 und in Hex damit einer 2.

Die Hex Zahl ist somit »2A«.

Beispiel

Umwandlung von 110110011₂:

Viererpakete: **0001|1011|0011**

Einzelne Umwandlungen:

$$0001_2 \rightarrow 1$$

$$1011_2 \rightarrow 11 (= B)$$

$$0011_2 \rightarrow 3$$

Die Hex Zahl ist somit »1B3«

4.3 Dezimalbrüche zu Binär

Vorgehen

Führe schriftliches Dividieren der Brüche im binären System aus (das geht analog zum dezimalen). Das Ergebnis der Division ist auch das Ergebnis der Konvertierung.

Beispiel

$$\frac{3}{4} = 011_2 : 100_2$$

$$\begin{array}{r} 0 \ 1 \ 1 : 100_2 = 0.11_2 \\ \underline{1 \ 1 \ 0} \\ - 1 \ 0 \ 0 \\ \underline{ 1 \ 0 \ 0} \\ - 1 \ 0 \ 0 \\ \underline{ 0} \\ 0 \end{array}$$

Das Ergebnis ist also »0.11₂«.

Beispiel

$$\frac{2}{3} = 010_2 : 011_2$$

$$\begin{array}{r} 0 \ 1 \ 0 : 011_2 = 0.101_2 \\ \underline{1 \ 0 \ 0} \\ - 0 \ 1 \ 1 \\ \underline{ 0 \ 0 \ 1} \ 0 \text{ (Periodisch)} \end{array}$$

Hier haben wir den Fall, dass sich die Rechnung ständig wiederholt, als nächstes würden wieder eine Null und eine Eins kommen usw.

Das Ergebnis ist also:

$$0.10\overline{1}_2$$

4.4 Maschinenzahlen

Erklärung

Der Computer speichert Zahlen in binärer Form und hat dafür nur begrenzt viel Speicher. Diese Zahlen nennt man **Maschinenzahlen** (kurz M.Zahl).

Es stehen allgemein gesprochen n **Bits** für eine Zahl zur Verfügung.

4.4.1 2-Komplement-Darstellung

Erklärung

Die 2-Komplement-Darstellung ist eine Möglichkeit, mit Bits (also binär) positive sowie negative Ganzzahlen darzustellen (Ganzzahlen sind solche, die ohne ein Komma auskommen).

In dieser Darstellung sind alle Bits normal binär kodiert mit einer Ausnahme. Das vorderste (erste/linkeste) Bit repräsentiert -2^{n-1} .

Das bedeutet: Wenn nur das erste Bit eine 1 ist, stellt die Zahl die kleinst-mögliche Zahl dar. Bei 5 Bits wäre das z.B. $-2^{5-1} = -2^4 = -16$

Daraus folgt, dass man mit n Bits die Zahlen von (-2^{n-1}) bis $(2^{n-1} - 1)$ darstellen kann.

Beispiel

Mit 5 Bits kann man die Zahlen von -16 $[10000_2]$ bis 15 $[01111_2]$ darstellen.

Vorgehen

Wenn man nun umrechnen will, geht man am besten von links nach rechts und addiert alle umgewandelten Ziffern aufeinander (eigentlich analog zu einer binären Zahl).

Die Bitstelle im Folgenden wurde von links nach rechts nummeriert. Das n -te Bit ist also das Bit ganz rechts.

Stelle des Bits	1	2	3	...	n
Wert des Bits	-2^{n-1}	2^{n-2}	2^{n-3}	...	2^0

Tabelle 4: Darstellung

Anmerkung:

Im Grunde sind das normale binäre Zahlen mit Ausnahme des ersten Bits!

Beispiel

Im Folgenden seien 4 Bits verfügbar:

(Zur besseren Übersicht wurde das erste Bit sichtbar getrennt)

M.Zahl	→	Dezimalzahl
0 001	→	1
0 101	→	5
1 000	→	-8
1 001	→	-7
1 011	→	-5

Vorgehen

Eine Zahl zu invertieren (z.B. 5 in -5)

Erst einmal die Zahl in die 2-Komplement Darstellung bringen.

- Bitweise Negation (also nullen zu einsen und andersherum)
- Addiere 1 auf die Zahl

Von Minus nach Plus dann alles rückwärts (statt addieren ist es dann subtrahieren)

Beispiel

5 in -5 umwandeln:

5	→	0101	(Binär schreiben)
0101	→	1010	(Bitweise Negation)
1010	→	1011	(Addition)

-7 in 7 umwandeln:

-7	→	1001	(Binär schreiben)
1001	→	1000	(Subtraktion)
1000	→	0111	(Bitweise Negation)

4.4.2 Gleitkommazahlen**Erklärung**

Nun möchten wir aber auch reelle Zahlen, oder Approximationen solcher, darstellen können. Das nennen wir die Gleitkommazahlen (im englischen »floating point numbers«).

Für diese benötigen wir das Verständnis von binären Kommazahlen, also hier eine kurze Wiederholung:

Binär	1.0_2	0.1_2	0.01_2	0.001_2	0.0001_2	0.00001_2	...
Dezimalwert	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$...

Tabelle 5: Wdh. Binäre Kommazahlen

Beispiel

Binäre Zahl	→	Dezimalzahl
1.11_2	→	1.75
111.001_2	→	7.125
$0.01_2 + 0.01_2 = 0.1_2$	→	$\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$

Tabelle 6: Binäre Kommazahlen und deren dezimales Äquivalent.

Erklärung

Gleitkommazahlen bestehen (in den meisten Darstellungen) aus **Vorzeichen, Exponent und Mantisse**.

$$\text{Wert der Gleitkommazahl} = (-1)^{\text{Vorzeichen}} \cdot 2^{\text{Exponent}} \cdot \text{Mantisse}_2 \quad (1)$$

Abgekürzt schreibt man das häufig auch:

$$G = (-1)^V \cdot 2^E \cdot M$$

Man kann sich die Zahl so vorstellen, dass der Exponent lediglich eine Kommaverschiebung ist. Zum Beispiel gilt

$$(1.0)_2 \cdot 2^0 = (0.01)_2 \cdot 2^2 = (1000)_2 \cdot 2^{-3}$$

In dezimal wäre das:

$$1 \cdot 1 = \frac{1}{4} \cdot 4 = 8 \cdot \frac{1}{8}$$

Wichtig

Ohne eine Normierung ist die Darstellung nicht eindeutig, was bedeutet, dass es zur einer (dezimal) Zahl mehrere Möglichkeiten gibt, diese korrekt darzustellen (siehe oben). Also wenn zwei Zahlen (ohne Normierung) verglichen werden, muss nicht zwangsläufig diejenige Gleitkommazahl größer sein, die den größeren Exponenten hat!

Erklärung

Normierung

Bei einer normierten Mantisse wird der Exponent genau so gewählt, dass die Mantisse eine von Null verschiedene Stelle vor dem Komma hat (also dass die Mantisse die Form $1, \dots$ hat). So löst sich auch der letzte **Wichtig**-Block. Das machen wir aus dem Grund, da nun jede Zahl eine eindeutige Darstellung hat. Also ist die Mantisse genau dann normiert, wenn diese eine führende Eins hat und sofort ein Komma folgt.

Damit bezeichnet die normierte Mantisse unsere Nachkommastellen (plus die Eins vor dem Komma).

Beispiel

Eine dezimale 16 würde man von $(0.1)_2 \cdot 2^5$ zu $(1.0)_2 \cdot 2^4$ normieren.

4.4.3 IEEE-Standard

Erklärung

Der sogenannte IEEE Standard ist eine Darstellungsmöglichkeit, die in vielen Bereichen verwendet wird. Ihr stehen 32 Bits zur Verfügung, die folgende Verteilungen besitzen:

- 1 Bit für das Vorzeichen:
Null für positiv, Eins für negativ
- 8 Bits für den Exponenten:
Der Exponent soll aber auch negative Werte annehmen können, eine Erklärung wie das gemacht wird folgt noch.
- 23 Bits für die Mantisse (Nachkommastellen):
Von einer Normalisierung mit führender Eins wird ausgegangen.
Diese Eins muss **nicht** abgespeichert werden, wodurch man sich ein Bit spart.
- Genügen 23 Bits nicht für die Darstellung, muss gerundet werden.

Der Exponent entspricht einer normalen binären Zahl, aber da man negative Werte darstellen möchte, zieht man eine Konstante ab. So shiftet man den darstellbaren Bereich für den Exponenten von 0 bis $(2^n - 1)$ auf $(-2^{n-1} + 1)$ bis (2^{n-1}) . Die Konstante entspricht also $(2^{n-1} - 1)$. Im Beispiel der 8 Bits des Exponenten würde also die binäre Zahl um 127 verringert werden, damit man den eigentlichen Exponenten hat.

Beispiel

Abgespeicherten Bits		Wert
01111101	→	-2
00000001	→	-126
11111110	→	127
10000000	→	1

Tabelle 7: Beispiele der IEEE Exponenten und ihre tatsächlichen Werte (wegen dem konstanten Abzug von 127).

Wichtig

00000000_2 und 11111111_2 des Exponenten sind allerdings speziell reserviert (0, Infinity, NaN, ...).

Vorgehen

Dezimalzahl in IEEE umwandeln:

- Dezimalzahl in binäre Zahl umwandeln,
- Exponent so bestimmen, dass die restliche Zahl (Mantisse) normiert ist,
- Vorzeichen direkt umwandeln (Eins für negative, Null für positive Zahl),
- Exponent +127 nehmen und dann in binär umwandeln,
- Mantisse einfach hinschreiben (die Eins vor dem Komma weglassen!) und wenn nötig runden.

Beispiel

Dezimal	Binär	Gleitkommadarstellung	IEEE (V E M)
32.00	100000_2	$2^5 \cdot 1.0_2$	0 10000100 000000000000000000000000
13.75	1101.11_2	$2^3 \cdot 1.10111_2$	0 10000010 101110000000000000000000
-0.125	-0.001_2	$-1 \cdot 2^{-3} \cdot 1.0_2$	1 01111100 000000000000000000000000

Tabelle 8: Beispiele von Dezimalzahlen und ihre dazugehörige normierte Gleitkommadarstellung und wie die 32 Bits in IEEE aussehen würden.

4.4.4 Rundung

Erklärung

$\text{rd}(x)$ ist die Zahl, die herauskommt, wenn man x rundet.

Aber **wie** rundet man denn überhaupt bei Gleitkommazahlen?

Im Folgenden sei eine Zahl bis zu dem Zeichen »|« korrekt darstellbar (sprich alle Zahlen dahinter müssen gerundet werden).

Vorgehen

Es gibt vier Fälle für das Runden. Sei y jeweils beliebig:

$y \mid 0x$	mit x »beliebig«	→ Abrunden
$y \mid 1x$	mit x »nicht nur Nullen«	→ Aufrunden
$y1 \mid 1x$	mit x »nur Nullen«	→ Aufrunden
$y0 \mid 1x$	mit x »nur Nullen«	→ Abrunden

Wichtig

Die letzten beiden Regeln des vorangegangenen Blocks werden häufig auch die »uneindeutigen Fälle« genannt, je nach Notation und Darstellung können die Rundungsregeln dafür anders definiert sein (oben sind sie so definiert, dass sie immer zur geraden Mantisse gerundet wird). Schaut also hierbei, was in der Klausur angegeben ist!

Beispiel

Originalzahl	Gerundet	Aktion
$1.0 \mid 1010_2$	1.1_2	Aufgerundet
$1.0 \mid 0111_2$	1.0_2	Abgerundet
$1.1 \mid 1000_2$	10_2	Aufgerundet
$1.0 \mid 1000_2$	1.0_2	Abgerundet

Erklärung

Durch Rundungen entstehen jedes Mal Fehler. Wir unterscheiden zwischen folgenden Fehlertermen.

Absoluter Fehler:

$$f_{\text{abs}} := |x - \text{rd}(x)| \quad (2)$$

Relativer Fehler:

$$f_{\text{rel}} := \left| \frac{f_{\text{abs}}}{x} \right| \quad (3)$$

Beispiel

Sei eine Zahl x bis zu dem Zeichen »|« exakt darstellbar. Mit $x = 10.0|11_2$ wird aufgerundet (siehe oben). Also ist $\text{rd}(x) = 10.1_2$.

Der absolute Fehler dieser Rundung beträgt:

$$\begin{aligned} f_{\text{abs}}(x) &= |x - \text{rd}(x)| \\ &= |10.011_2 - 10.1_2| \\ &= 0.001_2 \\ &= \frac{1}{8} \end{aligned}$$

Den relative Fehler können wir dann wie folgt bestimmen:

$$\begin{aligned} f_{\text{rel}}(x) &= \left| \frac{f_{\text{abs}}}{x} \right| \\ &= \left| \frac{1/8}{10.011_2} \right| \\ &= \left| \frac{1/8}{2.375} \right| \\ &= \left| \frac{1}{8} \cdot \frac{8}{19} \right| \\ &= \frac{1}{19} \end{aligned}$$

4.4.5 Maschinengenauigkeit

Erklärung

Die Maschinengenauigkeit beschreibt die größte positive Zahl x , sodass $\text{rd}(x + 1) = 1$ gilt.

Wird berechnet durch:

$$\epsilon = \frac{1}{2} \cdot \beta^{1-t} \quad (4)$$

- ϵ = Maschinengenauigkeit
- β = Zahlenbasis (bei uns immer 2)
- t = Mantissenlänge
(bei IEEE ist dies 24, weil 23 Bits plus die nicht abgespeicherte Eins)

Damit beschreibt die Maschinengenauigkeit das Maß der Rundungsfehler, sie gibt also den maximalen relativen Rundungsfehler (bei der Darstellung von Zahlen) an. Mit anderen Worten: Egal wie präzise und genau ein Algorithmus ist, in aller Regel kann er nicht genauer als die Maschinengenauigkeit sein (deswegen der Name ☺).

Beispiel

Mit IEEE (siehe [Kapitel 4.4.3](#)) wäre die Maschinengenauigkeit also:

$$\begin{aligned} \epsilon_{\text{IEEE}} &= \frac{1}{2} \cdot \beta^{1-t} \\ &= \frac{1}{2} \cdot 2^{1-24} \\ &= 2^{-24} \end{aligned}$$

$1 + 2^{-24}$ im IEEE-Format ergibt gerundet noch genau Eins.

4.4.6 Absorption

Erklärung

Ein Problem der Gleitkommaarithmetik ist die sogenannte **Absorption**.

Diese findet statt, wenn bei der Addition zwei sehr unterschiedlich großer Zahlen Information verloren geht aka gerundet werden muss.

Beispiel

Man möchte 0.75 auf 7 aufaddieren, hat aber nur 3 Bits für die Mantisse zur Verfügung. Zu rechnen:

$$111_2 + 0.11_2 = 111.11_2$$

111.11_2 ist nicht mit 3 Bits darstellbar und führt damit zu einem Rundungsfehler!

In diesem Beispiel würde das Ergebnis aufgerundet werden und damit 1000_2 ergeben. Das wäre in dezimal eine 8. Mit exakter Arithmetik wäre es aber 7.75, wir haben einen absoluten Fehler von 0.25. Die 7 hat die 0.75 absorbiert, nennt man dann so.

Wichtig

Aus Absorption können wir folgendes herleiten: Bei Gleitkommazahlen geht die Eigenschaft des Assoziativgesetzes verloren!

4.4.7 Auslöschung

Erklärung

Unter **Auslöschung** versteht man den Verlust an Genauigkeit, wenn man zwei fast gleich große Gleitkommazahlen subtrahiert. Denn wenn man dies rechnet, rundet sich das Ergebnis zu einer Null, was verheerende Folgen haben kann. In aller Regel ist Auslöschung schlimmer als Absorption.

Beispiel

Sei

$$a = 2.3546; \quad b = 2.3510$$

Seien Zahlen im dezimalen nur auf 3 Ziffern genau darstellbar und der Rest jeweils normal nach der nächsten Ziffer gerundet.

Wenn wir jetzt auf die korrekten Ziffern kürzen:

$$2.35 - 2.35 = 0$$

Dabei wäre es analytisch:

$$2.3546 - 2.3510 = 0.0036 \implies \text{Keine einzige korrekte Ziffer!}$$

Vertiefung

Warum das so verheerend ist, kann man sich gut aus dem Beispiel erschließen. Denn gehen wir mal davon aus, dass wir nach der Operation aus dem Beispiel nur noch Multiplikationen ausführen müssen. Dann würde im verfälschten Fall immer Null das Ergebnis sein, exakt aber etwas völlig anderes!

Beispielsweise nach obiger Rechnung wird das Ergebnis (aus welchen Grund auch immer) mit 1000 multipliziert. Dann würde als verfälschte Lösung 0 rauskommen, exakt wäre aber eigentlich 3.6.

5 Kondition

Erklärung

Die Kondition beschreibt das Fehlerverhältnis zwischen Eingabe und Ausgabe. Man spricht von »guter« oder »schlechter« Kondition, es ist also ein qualitativer Begriff.

Die berechenbare Zahl, die Konditionszahl, entspricht dem ungünstigsten Faktor der Störung. Eine hohe Kondition bedeutet, dass ein Fehler in der Eingabe zu einem noch viel höherem Fehler in der Ausgabe führt. Die relative Konditionszahl $\text{cond}(f, x)$ wird folgendermaßen in Abhängigkeit von einer Funktion $f(x)$ berechnet:

$$\text{cond}(f, x) = \left| \frac{x \cdot f'(x)}{f(x)} \right| \quad (5)$$

Man unterscheidet generell folgende Fälle:

- $\text{cond}(f, x) \leq 1$

Da die Kondition den Fehlerfaktor beschreibt, bedeutet eine Eins, dass es maximal zu keiner Zunahme des relativen Fehlers kommt. Es ist »gut konditioniert«.

- $\text{cond}(f, x) \gg 1$

Der Fehlerfaktor ist deutlich größer eins und nimmt somit zu. Die Funktion ist »schlecht konditioniert«.

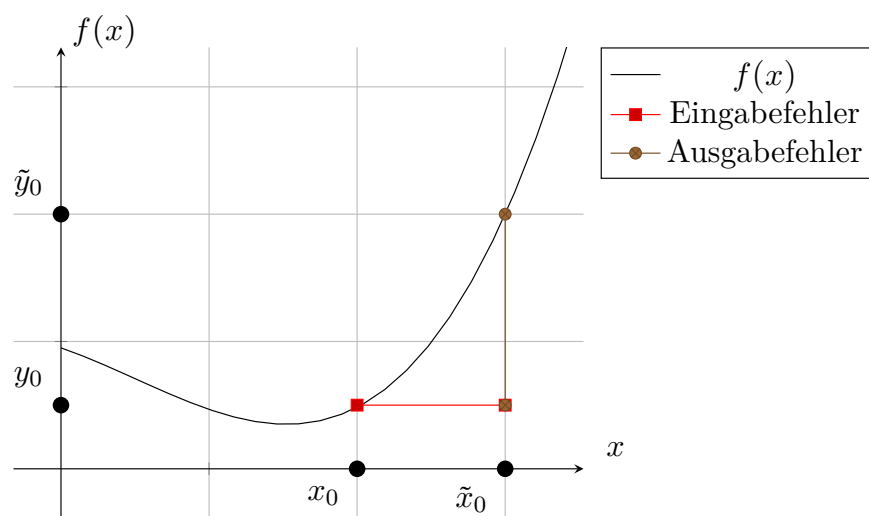


Abbildung 1: Plot einer Funktion $f(x)$ und ein Beispiel, wie die Kondition funktioniert. Die exakte Eingabe sei $x_0 = 2$ und wurde durch vorherige Operationen verfälscht (\tilde{x}_0). Die Differenz $|\tilde{x}_0 - x_0|$ nennt man den **Eingabefehler**. Dieser führt zu einer Veränderung der Ausgabe, also des y -Wertes. Der sogenannte **Ausgabefehler** beschreibt dann die Differenz zwischen der verfälschten Ausgabe \tilde{y}_0 und der exakten y_0 .

Beispiel

- $f(x) = x$

$$\text{cond}(f, x) = \left| \frac{x \cdot 1}{x} \right| = 1 \quad \implies \text{gut konditioniert}$$

Logisch, weil wir die Eingabe ja gerade wieder ausgeben. Wie soll sich da der Fehler der Eingabe überhaupt verändern?

- $g(x) = x^2$

$$\text{cond}(g, x) = \left| \frac{x \cdot 2x}{x^2} \right| = \frac{2x^2}{x^2} = 2 \quad \implies \text{immer noch gut konditioniert}$$

- $h(x) = e^x$

$$\text{cond}(h, x) = \left| \frac{x \cdot e^x}{e^x} \right| = |x|$$

Fallunterscheidung:

$$\begin{array}{ll} |x| \leq 1 & \implies \text{gut konditioniert} \\ |x| \gg 1 & \implies \text{schlecht konditioniert} \end{array}$$

- $i(x) = 1$

$$\text{cond}(i, x) = \left| \frac{x \cdot 0}{1} \right| = 0 \quad \implies \text{Fehler sind irrelevant}$$

Es ist »perfekt konditioniert«. Logisch, da die Ausgabe nicht abhängig von der Eingabe ist. Wie soll die Ausgabe also von einem Eingabefehler überhaupt beeinflusst werden?

6 Stabilität

Erklärung

Ein Verfahren heißt »stabil«, wenn Rundungen von Zwischenergebnissen keine große Abweichung der Endergebnisse zur Folge hat; ansonsten spricht man von einem »instabilen« Algorithmus.

Bei der Stabilitäts Untersuchung geht es darum, die Fehlerverstärkung innerhalb eines Berechnungsverfahrens zu analysieren. Instabil kann nur eine Verkettung von Operationen sein, einzelne sind immer stabil!

Weil viele damit Schwierigkeiten haben sind hier nochmal knapp die Erläuterung von Stabilität & Kondition:

- **Kondition**

- Ist eine Eigenschaft eines Problems.
- Sagt aus, wie groß sich ein Eingabefehler auf die Ausgabe auswirkt.
- Quasi $f(\text{rd}(x)) - f(x)$.

- **Stabilität**

- Ist eine Eigenschaft eines Algorithmus.
- Sagt aus, ob sich der relative Fehler durch Rundungen stark verschlechtert oder nicht.
- Quasi $\text{rd}(f(x)) - f(x)$.

Um die Stabilität zu untersuchen verwenden wir Epsilontik.

Erklärung

Epsilontik

Wir haben ja schon gelernt, dass das Ergebnis einer (auch sehr einfachen) Operation, die ich jetzt allgemein »op« nenne, nicht mehr als Maschinenzahl exakt darstellbar sein kann und dadurch gerundet werden muss (siehe [Kapitel 4.4.6 und 4.4.7](#)).

Außerdem haben wir gelernt, dass bei Rundungen immer ein relativer Fehler entsteht. Für diesen Fehler ϵ gilt immer $|\epsilon| \leq \epsilon_{\text{Ma}}$. Dieses ϵ_{Ma} ist hierbei die maximale Fehlergrenze und entspricht der Maschinengenauigkeit der Darstellung (siehe [Kapitel 4.4.5](#)).

$$f_{\text{rel}} = \left| \frac{\text{rd}(x) - x}{x} \right| \leq \epsilon_{\text{Ma}}. \quad (6)$$

Das heißt aber auch, dass es immer ein $\epsilon \in \mathbb{R}$ geben muss, damit: $\text{rd}(x) = (1 + \epsilon) \cdot x$; mit $-\epsilon_{\text{Ma}} \leq \epsilon \leq \epsilon_{\text{Ma}}$

Diese Eigenschaft verwenden wir bei der Epsilontik. Wir untersuchen den relativen Endergebnisfehler:

$$\left| \frac{\text{rd}(f)(x) - f(x)}{f(x)} \right| \quad (7)$$

Dabei gelten noch folgende Regeln:

Mit »op« als exakte Operation erzeugt jede Operation »op_M« einen Fehler $\epsilon_i \leq \epsilon_{\text{Ma}}$:

$$(a \text{ op}_M b) = \text{rd}_M(a \text{ op } b) = (a \text{ op } b) \cdot (1 + \epsilon_i) \quad \text{mit } |\epsilon_i| \leq \epsilon_{\text{Ma}} \quad (8)$$

Außerdem lassen wir Fehler höherer Ordnung weg, da wir davon ausgehen, dass unsere Fehler ϵ_i sehr klein sind und somit $\epsilon_i \cdot \epsilon_j = 0 \quad \forall i, j$. Die Werte werden einfach zu klein werden, um Relevanz zu haben. Eine sehr kleine Zahl multipliziert mit einer sehr kleinen Zahl wird nur noch kleiner.

Beispiel

- $f(x) = x$

$$\text{rd}(f)(x) = x$$

Wir haben keine Operationen, also findet auch keine Erzeugung eines relativen Fehlers statt. Das ist logischerweise stabil.

- $g(x) = x^2$

Wir haben eine Operation, nämlich die Potenz und dadurch wird ein Fehler ϵ erzeugt:

$$\text{rd}(g)(x) = x^2 \cdot (1 + \epsilon) = g(x) + g(x) \cdot \epsilon$$

Endergebnisfehler:

$$\left| \frac{\text{rd}(g)(x) - g(x)}{g(x)} \right| = \left| \frac{g(x) + g(x) \cdot \epsilon - g(x)}{g(x)} \right| = |\epsilon|$$

Als Endergebnisfehler kommt nur der Operationsfehler heraus, die Funktion ist offensichtlich stabil.

- $h(x) = e^x$

e^x soll hierbei auch nur einen relativen Fehler $\leq \epsilon_{\text{Ma}}$ erzeugen (kann in Klausuren anders definiert sein!).

$$\text{rd}(h)(x) = e^x \cdot (1 + \epsilon) = h(x) + h(x) \cdot \epsilon$$

Danach ist die Rechnung wie bei Beispiel 2. Es ist stabil.

- $i(x) = \frac{x+2}{a}$ (mit $a \in \mathbb{R}$)

$$\begin{aligned} \text{rd}(i)(x) &= \frac{(x+2) \cdot (1+\epsilon_1)}{a} \cdot (1+\epsilon_2) = \frac{(x+2)}{a} \cdot (1+\epsilon_1+\epsilon_2) \\ &= i(x) + i(x) \cdot (\epsilon_1 + \epsilon_2) \end{aligned}$$

Endergebnisfehler:

$$\left| \frac{\text{rd}(i)(x) - i(x)}{i(x)} \right| = \left| \frac{i(x) + i(x) \cdot (\epsilon_1 + \epsilon_2) - i(x)}{i(x)} \right| = |\epsilon_1 + \epsilon_2|$$

Diese Funktion ist offensichtlich auch stabil.

- $j(x) = x^{(x+1)}$

$$\begin{aligned} \text{rd}(j)(x) &= x^{(x+1) \cdot (1+\epsilon_1)} \cdot (1+\epsilon_2) \\ &= x^{(x+1)} \cdot x^{(x+1) \cdot \epsilon_1} \cdot (1+\epsilon_2) \\ &= \left(x^{(x+1)} + \epsilon_2 x^{(x+1)} \right) \cdot x^{(x+1) \cdot \epsilon_1} \\ &= (j(x) + \epsilon_2 j(x)) \cdot j(x)^{\epsilon_1} \\ &= j(x)^{(\epsilon_1+1)} + \epsilon_2 j(x)^{(\epsilon_1+1)} \end{aligned}$$

Endergebnisfehler:

$$\left| \frac{\text{rd}(j)(x) - j(x)}{j(x)} \right| = \left| \frac{j(x)^{(\epsilon_1+1)} + \epsilon_2 j(x)^{(\epsilon_1+1)} - j(x)}{j(x)} \right| = |j(x)^{\epsilon_1} + \epsilon_2 j(x)^{\epsilon_1} - 1|.$$

Nicht triviales Beispiel. Wir untersuchen (für dieses Beispiel) $x \rightarrow \infty$:

$$\begin{aligned} \lim_{x \rightarrow \infty} |j(x)^{\epsilon_1} + \epsilon_2 j(x)^{\epsilon_1} - 1| &= \\ \text{Da gilt: } \lim_{x \rightarrow \infty} |j(x)| &= \lim_{x \rightarrow \infty} |x^{(x+1)}| = \infty \\ \lim_{x \rightarrow \infty} |j(x)^{\epsilon_1} + \epsilon_2 j(x)^{\epsilon_1} - 1| &= \infty \end{aligned}$$

Scheinbar ist diese Funktion sehr deutlich instabil für $x \rightarrow \infty$.

Wichtig

Ein Algorithmus kann z.B. **instabil sein trotz guter Kondition** und auch andersherum, also stabil trotz schlechter Kondition!

In letzterem Falle ist die Eigenschaft der Stabilität in Anbetracht der schlechten Kondition wertlos! Was hilft uns eine kleine Abweichung der Endergebnisse (Stabilität), wenn wir schon »vorher« einen hohen Störungsfaktor zwischen Eingabe und Ausgabedaten haben (Kondition)?

Auch andersherum (gute Kondition, schlechte Stabilität) klingt sinnlos. Hier ist aber wichtig zu wissen, dass man Stabilität durch Umstellen des Algorithmus oder Operationen eventuell retten kann. Bei schlechter Kondition geht das nicht (außer vielleicht Vorkonditionierung).

7 Lineare Gleichungssysteme

Erklärung

Mit linearen Gleichungssystemen muss man sich des Öfteren auseinandersetzen, da sie alleine schon als Teilproblem häufig auftreten. Wann immer wir ein lineares Gleichungssystem (kurz LGS) haben, berechneten wir dessen Lösung bisher mithilfe des **Gauß-Elminationsverfahrens**.

Was war doch gleich ein LGS?

Ein LGS haben wir immer dann, wenn wir mehrere Gleichungen mit Unbekannten haben und wissen möchten, welche Werte die einzelnen Unbekannten haben müssen/-können, damit alle Gleichungen erfüllt sind

Allgemein beschreiben wir sie in der Form $Ax = b$, wobei A eine Matrix und x und b Vektoren sind (b wird auch manchmal »Ergebnisvektor« genannt).

Beispiel

$$1x + 2y + 2z = 3 \quad (1)$$

$$2x + 1y - 2z = 2 \quad (2)$$

$$3x + 0y + 2z = 6 \quad (3)$$

Zu lösen:

Welche Werte müssen x , y und z haben, damit alle Gleichungen (1) – (3) erfüllt sind?

7.1 Gauß-Elimination

Vorgehen

Wir führen alle Gleichungen in eine Matrix-Schreibweise um und führen eine **Vorwärtssubstitution** aus (das nennt man den Vorgang, die Matrix in eine rechts-obere Dreiecksmatrix zu bringen).

In dieser Vorwärtssubstitution verwenden wir die typischen drei elementaren Zeilenoperationen:

1. Reihe + Reihe
2. Reihe multipliziert mit Skalar
3. Vertauschen von 2 Reihen

Beispiel

(die Zahlen aus vorherigem Beispiel)

$$\left[\begin{array}{ccc|c} 1 & 2 & 2 & 3 \\ 2 & 1 & -2 & 2 \\ 3 & 0 & 2 & 6 \end{array} \right] = \left[\begin{array}{ccc|c} 1 & 2 & 2 & 3 \\ 0 & -3 & -6 & -4 \\ 0 & -6 & -4 & -3 \end{array} \right] = \left[\begin{array}{ccc|c} 1 & 2 & 2 & 3 \\ 0 & -3 & -6 & -4 \\ 0 & 0 & 8 & 5 \end{array} \right]$$

Ab diesem Zeitpunkt verwenden wir jetzt nur noch die **Rückwärtssubstitution** um auf alle Variablenwerte zu schließen, indem wir uns von unten nach oben vorarbeiten.

$$8 \cdot z = 5$$

$$z = \frac{5}{8}$$

$$-3y - 6z = -4$$

$$-3y - 6 \cdot \frac{5}{8} = -4$$

$$-3y = -4 + \frac{30}{8}$$

$$-3y = \frac{-1}{4}$$

$$y = \frac{1}{12}$$

$$1x + 2y + 2z = 3$$

$$1x + 2 \cdot \frac{1}{12} + 2 \cdot \frac{5}{8} = 3$$

$$x = 3 - \frac{1}{6} - \frac{10}{8}$$

$$x = \frac{19}{12}$$

Wir konnten die (in diesem Fall drei) Unbekannten x , y und z so bestimmen, dass sie alle Gleichungen erfüllen.

7.2 Pivotisierung

Erklärung

Jetzt möchten wir dieses Verfahren einem Computer beibringen. Als **Laufzeit** erhalten wir für Gauß (= Vorwärtssubst.) und Rückwärtssubst. zusammen $\mathcal{O}(n^3)$.

Wir wissen schon aus vergangenen Tutorübungen, dass Zahlen von einem PC häufig nicht korrekt dargestellt werden (können) und deswegen gerundet werden müssen (siehe [Kapitel 4.4.4](#)).

Was dagegen tun?

Als Pivotelement wird das Element **ganz links oben** in dem noch übrig gebliebenen Teil der Matrix genannt. Ein betragsmäßig kleines Pivot führt zu einem instabileren Algorithmus, da die Faktoren beim Verrechnen von zwei Zeilen dann größer werden.

Somit ist es immer besser, die Zeilen der Matrix so zu vertauschen, dass das betragsmäßig größte Element zum Pivot wird. Wenn wir dies bei jeder Iteration (Spalte) machen, erhalten wir die maximale Stabilität.

Wichtig

Kurze Überlegung:

Wenn das Pivot 0 ist, keine Pivotisierung angewendet wurde und mindestens ein anderer Wert in der ersten Spalte ungleich 0 ist, kann der Algorithmus gar nicht weiterarbeiten!

Wer mir nicht glaubt, probiere mal, die zweite Zeile mit der ersten so zu verrechnen, dass eine neue 0 entsteht ☺.

7.3 LR-Zerlegung

Erklärung

Der LR-Zerlegungs Algorithmus soll eine bessere Variante sein, ein System der Form $Ax = b$ auszurechnen. Dazu trennt es die Matrix in zwei andere Matrizen L und R auf, die weitere Berechnungen einfacher gestalten lassen sollen.

Was ist L , was ist R ?

Man schaue sich die Gauß Zeilenoperationen an einem allgemeinen Beispiel genauer an:

$$\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \Rightarrow \begin{bmatrix} a & d & g \\ 0 & e - \frac{d \cdot b}{a} & h - \frac{g \cdot b}{a} \\ c & f & i \end{bmatrix} \Rightarrow \dots$$

Nun ist die Idee, die Umformung jedes Schrittes nicht als Zeilenoperation umzusetzen, sondern eine Matrix zu finden, die multipliziert mit A dasselbe Ergebnis liefert.

Obigen Umformungsschritt kann man auch folgendermaßen schreiben:

$$\begin{bmatrix} 1 & 0 & 0 \\ -\frac{b}{a} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} = \begin{bmatrix} a & d & g \\ 0 & e - \frac{d \cdot b}{a} & h - \frac{g \cdot b}{a} \\ c & f & i \end{bmatrix}$$

Wenn man mehrere Beispiele aufschreiben würde, kann man sehen, dass alle konstruierten Matrizen ähnlich der obigen ist. Wichtige Werte wie bei dem Beispiel das $-\frac{b}{a}$ stehen immer im links unteren Bereich der Matrix und die Diagonaleinträge sind immer Einsen.

Die Idee der LR-Zerlegung ist es jetzt, alle diese konstruierten Matrizen aufeinander zu multiplizieren; wir nennen sie die inverse L Matrix, also L^{-1} . Nun beschreibt R die Matrix, die man bekommt, wenn man eine reguläre Vorwärtssubstitution ausführen würde, also eine rechts obere Dreiecksmatrix (wie wenn man Gauß Elimination anwenden würde). Nochmal in kurz:

$$L^{-1} \cdot A = R \iff A = L \cdot R \quad (9)$$

Zusammengefasst ist L eine links untere Dreiecksmatrix mit Einsen auf der Diagonalen und R eine rechts obere Dreiecksmatrix.

Nochmal zur Klarstellung: $A = L \cdot R$. Daher auch der Name LR-Zerlegung. Im Englischen spricht man von der LU-Zerlegung, da dort nicht links und rechts namensgebend war, sondern die untere (lower) L und die obere (upper) U Dreiecksmatrix. Es ist aber genau dasselbe!

VorgehenZerlegung der Matrix in L und R :

Herleitungen sind toll, bringen uns aber auch nicht viel weiter, wenn wir nicht wissen, wie man L und R überhaupt ausrechnet.

Es gibt mehrere Methoden, die Matrix zu zerlegen und L und R zu finden. Ich stelle euch jetzt meinen persönlichen Favoriten vor (das bedeutet nicht, dass es für euch unbedingt die beste Methode ist!).

Schreiben wir auf, was wir über die Matrizen wissen:

$$\begin{bmatrix} & & \\ A & & \\ & & \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} & & \\ 0 & & \\ 0 & 0 & \end{bmatrix} \quad (10)$$

Diese Gleichung lässt sich lösen, da A bekannt ist und wir mittels Matrixmultiplikation auf die Werte in L und R schließen können.

Beispiel

Achtung, sehr lang!

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} & & \\ 0 & & \\ 0 & 0 & \end{bmatrix}$$

Schauen wir uns den ersten Wert an:

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} x & & \\ 0 & & \\ 0 & 0 & \end{bmatrix}$$

$$1 \cdot x + 0 \cdot 0 + 0 \cdot 0 = 1 \implies x = 1$$

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & \\ 0 & & \\ 0 & 0 & \end{bmatrix}$$

Alle weiteren Schritte analog (etwas schneller)

$$\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & \\ 0 & & \\ 0 & 0 & \end{bmatrix}$$

$$\text{weil } x \cdot 1 + 1 \cdot 0 + 0 \cdot 0 = 2 \implies x = 2$$

$$\begin{aligned}
\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \color{red}{2} \\ 0 & \\ 0 & 0 \end{bmatrix} \\
\text{weil } 1 \cdot x + 0 + 0 \cdot 0 &= 2 \implies x = 2 \\
\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & \color{red}{-3} \\ 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ \color{red}{3} & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & -3 \\ 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & \color{red}{2} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & -3 \\ 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & \color{red}{2} \\ 0 & -3 \\ 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 3 & 0 & 2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 2 \\ 0 & -3 & \color{red}{-6} \\ 0 & 0 & \color{red}{8} \end{bmatrix}
\end{aligned}$$

Somit ergibt sich:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -3 & -6 \\ 0 & 0 & 8 \end{bmatrix}$$

Erklärung

Vorwärts- und Rückwärtssubst.

Wenn wir jetzt die Vorwärts- und Rückwärtssubst. noch modifizieren, erhalten wir ein Verfahren, welches ein beliebiges LGS lösen kann.

Also vom Prinzip:

$$\begin{aligned}
Ax &= b \\
A &= L \cdot R \\
\iff LRx &= b
\end{aligned}$$

Substituieren wir Rx mit y , erhalten wir nachfolgende Methode.

Vorgehen

LR-Zerlegungs-Verfahren zum Lösen eines LGS der Form $Ax = b$:

1. Zerlegung der Matrix:

$$A = L \cdot R \quad (11)$$

2. Vorwärtssubstitution:

$$Ly = b \quad (12)$$

3. Rückwärtssubstitution:

$$Rx = y \quad (13)$$

Beispiel

Lösen wir vorheriges Beispiel zuende. Wir nehmen als b dasselbe wie beim Gauß-Beispiel:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -3 & -6 \\ 0 & 0 & 8 \end{bmatrix} \quad b = \begin{pmatrix} 3 \\ 2 \\ 6 \end{pmatrix}$$

Vorwärtssubstitution, also $Ly = b$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 6 \end{pmatrix}$$

Lösen, indem wir nacheinander von oben nach unten einsetzen:

$$1y_0 = 3$$

$$2y_0 + 1y_1 = 2$$

$$y_1 = -4$$

$$3y_0 + 2y_1 + y_2 = 6$$

$$9 + -8 + y_2 = 6$$

$$y_2 = 5$$

Rückwärtssubstitution, also $Rx = y$:

$$\begin{bmatrix} 1 & 2 & 2 \\ 0 & -3 & -6 \\ 0 & 0 & 8 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ -4 \\ 5 \end{pmatrix}$$

Jetzt haben wir wieder genau dasselbe System wie nach Gauß, die Lösung ist logischerweise dieselbe (siehe [Kapitel 7.1](#)).

Erklärung

Vorwärts und Rückwärtssubstitution aus dem Algorithmus sind dann auszurechnende LGSs mit der schönen Eigenschaft, dass man sie verhältnismäßig einfach ausrechnen kann (wegen den Dreiecksmatrizen).

Man kann jetzt sehr schnell sehen, warum das so effizient sein kann. Denn egal was b in der Gleichung ist, die Zerlegung für A bleibt dieselbe und muss nach dem ersten Mal nicht mehr ausgerechnet werden.

So liegt der Aufwand bei der Berechnung mit Zerlegung in $\mathcal{O}(n^3)$, ohne Berechnung der Zerlegung (wenn wir sie schon haben) nur in $\mathcal{O}(n^2)$ und damit schneller als Gauß-Elimination.

Wichtig

Die LR-Zerlegung ist eine schöne Anwendung, wenn wir mit derselben Matrix A viele LGS der Form $Ax = b$ lösen müssen!

7.4 Orthogonale Matrizen

Erklärung

Orthogonale Matrizen sind quadratische Matrizen mit orthogonalen Spalten der Länge Eins. Anders, orthogonale Matrizen sind solche, bei denen jede Zeilen- und Spaltenvektoren paarweise orthonormal zueinander (sprich das Skalarprodukt $= 0$ ist) sind.

Im Folgenden beschreibt $\mathbb{1}$ immer die Einheitsmatrix.

Für eine orthogonale Matrix Q gilt:

$$Q^T = Q^{-1} \quad (14)$$

Daraus wiederum folgt:

$$Q^T \cdot Q = \mathbb{1} \quad (15)$$

In der Tutorstunde beweisen wir auch, dass die Determinante jeder beliebigen orthogonalen Matrix ± 1 ist und die Spektralnorm Eins beträgt. In kurz:

$$\det(Q) = \pm 1 \quad (16)$$

$$\|Q\|_2 = 1 \quad (17)$$

Beispiel

Als Beispiel wäre die Matrix:

$$\frac{1}{5} \begin{bmatrix} 3 & 4 \\ -4 & 3 \end{bmatrix}$$

$$\frac{1}{5} \begin{bmatrix} 3 & -4 \\ 4 & 3 \end{bmatrix} \cdot \frac{1}{5} \begin{bmatrix} 3 & 4 \\ -4 & 3 \end{bmatrix} = \frac{1}{25} \begin{bmatrix} 9+16 & 12-12 \\ 12-12 & 16+9 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{1}$$

Jetzt noch die Determinante überprüfen:

$$\det\left(\frac{1}{5} \begin{bmatrix} 3 & 4 \\ -4 & 3 \end{bmatrix}\right) = \frac{1}{25} \cdot (9+16) = 1$$

Wichtig

Orthogonale Matrizen können unsere Kondition nie verändern. Wenn wir ja die Spektralnorm (siehe [Kapitel 8.2](#)) als Maß unseres Fehlers betrachten, muss dies stimmen, da wir schon bewiesen haben, dass jegliche orthogonale Matrix normiert eins ist!

7.5 QR-Zerlegung

Erklärung

Bei der LR Zerlegung trennten wir die Matrix A in eine linke Dreiecksmatrix L und in eine rechte Dreiecksmatrix R auf ($A = L \cdot R$), um ein LGS mit derselben Matrix A schneller zu lösen (siehe [Kapitel 7.3](#)).

Wir fanden heraus, dass die Spektralnorm einer Matrix für uns ein Maß der stärksten Verzerrung ist und dass diese bei einer orthogonalen Matrix immer 1 ist, sprich, es gibt keine Verzerrung (siehe [Kapitel 8.2](#)).

Also wäre es praktisch, die Matrix A so zu zerlegen, dass wir mit einer orthogonalen Matrix rechnen, damit der Algorithmus stabiler wird.

7.5.1 Givens-Rotation

Erklärung

Wir möchten nun die Matrix A in eine orthogonale Matrix Q und eine rechts obere Dreiecksmatrix R auftrennen. Damit reduziert sich das Lösen eines LGS auf eine einfache Rückwärtssubstitution: $A = QR$

$$Ax = b \iff Q \cdot Rx = b \iff Rx = Q^T b \quad (18)$$

Das funktioniert, da ja $Q^{-1} = Q^T$ gilt (siehe [Kapitel 7.4](#)).

Die Matrix Q wird hierbei durch eine Folge von »Drehungen« konstruiert. Jede Drehung (oder Rotation) entspricht hierbei der Eliminierung eines Eintrages unter der Diagonalen von A , ähnlich der konstruierten Matrizen bei der LR-Zerlegung. Für 2×2 Matrizen benötigt man also logischerweise nur eine einzige Drehung, da nur ein Eintrag unter der Diagonalen zu einer Null rotiert werden muss.

Ganz allgemein beschreibt eine Drehung im \mathbb{R}^2 um den Winkel φ folgende orthogonale Rotationsmatrix G_φ :

$$\text{Mit } c := \cos(\varphi) \text{ und } s := \sin(\varphi) : \quad G_\varphi = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

Aus der Grafik ([Figur 7.5.1](#)) können wir auch erkennen, warum c und s ihre Werte haben (rechtwinkliges Dreieck, Sinus und Cosinus jeweils Gegenkathete bzw. Ankathete durch Hypotenuse). Auf unsere LGS Problematik angewandt filtert man also die nachfolgende Methode heraus:

Vorgehen

Mit $(a, b)^T$ als die erste Spalte der 2×2 Matrix A ,

$$c = \frac{a}{\sqrt{a^2 + b^2}}; \quad s = \frac{b}{\sqrt{a^2 + b^2}}$$

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

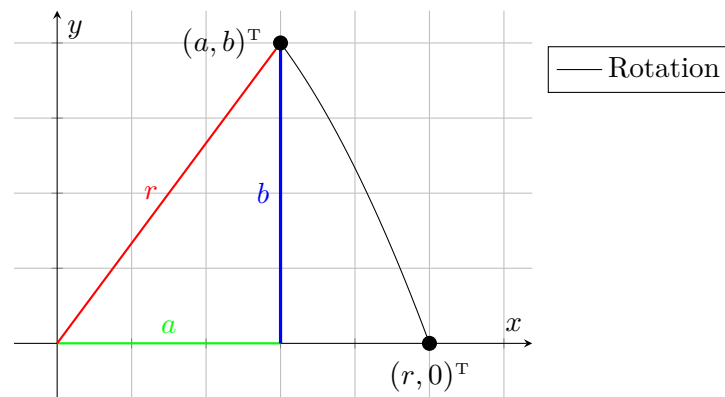


Abbildung 2: Beispiel einer Rotation

Jetzt müssen wir nur noch unser Wissen anwenden:

$$G_\varphi A = R$$

Genau wie bei der LR-Zerlegung nennen wir die Rotationsmatrix jetzt einfach die inverse Q Matrix, also Q^{-1} . Durch die Eigenschaft der Orthogonalität müssen wir das Inverse nicht ausrechnen, da das Inverse ja gleich der Transponierten ist.

Somit erhalten wir:

$$\begin{aligned} Q^{-1} A &= R \\ A &= QR \end{aligned}$$

$$\begin{aligned} Q^{-1} &= G_\varphi \\ Q^T &= G_\varphi \\ Q &= G_\varphi^T \end{aligned}$$

Beispiel

Gegeben sei folgendes LGS:

$$Ax = b$$

$$\begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Damit die Matrix A eine rechts obere Dreiecksmatrix ist, müssen wir die links untere Eins zu einer Null rotieren. Also den Vektor der ersten Spalte, $(2, 1)^T$ rotieren zu $(r, 0)^T$. Das r müssen wir noch ausrechnen, indem wir die Rotation ausführen.

Aus der ersten Spalte a und b auslesen: $a = 2$; $b = 1$

$$\sqrt{a^2 + b^2} = \sqrt{5}$$

$$c = \frac{2}{\sqrt{5}}; \quad s = \frac{1}{\sqrt{5}}$$

$$G_\varphi \cdot A = \frac{1}{\sqrt{5}} \cdot \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} = \frac{1}{\sqrt{5}} \cdot \begin{bmatrix} 5 & 7 \\ 0 & -1 \end{bmatrix}$$

Jetzt haben wir eine rechts obere Dreiecksmatrix und müssen es noch lösen:

$$R = \frac{1}{\sqrt{5}} \cdot \begin{bmatrix} 5 & 7 \\ 0 & -1 \end{bmatrix}$$

$$Rx = Q^T b = G_\varphi b$$

$$\frac{1}{\sqrt{5}} \cdot \begin{bmatrix} 5 & 7 \\ 0 & -1 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \frac{1}{\sqrt{5}} \cdot \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} \cdot \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

$$\begin{bmatrix} 5 & 7 \\ 0 & -1 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$$

Das können wir wie üblich (also wie bei LR sowie Gauß Elimination) lösen.

7.5.2 Weiterführung

Erklärung

Diese Zerlegung ist natürlich auch für allgemeine $n \times n$ Matrizen gültig, für eine 3×3 Matrix A würden drei Rotationsmatrizen benötigt werden.

Die Positionen vom links oberen c und dem $-s$ entspricht dabei der gleichen Position aus der Matrix A , wo man a und b abliest (im Folgenden ist dies als Index markiert). Das bedeutet nicht, dass sich die von dem anderen c oder s zahlentechnisch unterscheiden!

$$G_{2,1} = \begin{bmatrix} c_a & s & 0 \\ -s_b & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \implies G_{2,1} \cdot A = A_1 \quad (19)$$

$$G_{3,1} = \begin{bmatrix} c_a & 0 & s \\ 0 & 1 & 0 \\ -s_b & 0 & c \end{bmatrix} \implies G_{3,1} \cdot A_1 = A_2 \quad (20)$$

$$G_{3,2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_a & s \\ 0 & -s_b & c \end{bmatrix} \implies G_{3,2} \cdot A_2 = R \quad (21)$$

$G_{2,1}$ entspricht einer Rotation um die z -Achse, $G_{3,1}$ um y -Achse und $G_{3,2}$ dann um die x -Achse. Insgesamt kann man dann Q berechnen mit:

$$Q^T = G_{3,2} \cdot G_{3,1} \cdot G_{2,1} \quad (22)$$

$$Q = G_{2,1}^T \cdot G_{3,1}^T \cdot G_{3,2}^T \quad (23)$$

$$Rx = Q^T \cdot b \quad (24)$$

Wichtig

Schaut bitte nach jeder Drehung, ob nicht vielleicht schon eine weitere Null erzeugt wurde. Damit erspart ihr euch Rotationen und die Rechnungen gehen viel schneller und einfacher!

Beispiel

Wir möchten gerne folgende Matrix in Q und R auftrennen:

$$A = \begin{bmatrix} 3 & 1 & 2 \\ 4 & 3 & 1 \\ 3 & 0 & 2 \end{bmatrix}$$

Erste Rotation:

$$a = 3; \quad b = 4$$

$$c = \frac{3}{\sqrt{3^2 + 4^2}} = \frac{3}{5}; \quad s = \frac{4}{5}$$

$$G_{2,1} = \frac{1}{5} \cdot \begin{bmatrix} 3 & 4 & 0 \\ -4 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

$$G_{2,1} \cdot A = \frac{1}{5} \cdot \begin{bmatrix} 3 & 4 & 0 \\ -4 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 & 2 \\ 4 & 3 & 1 \\ 3 & 0 & 2 \end{bmatrix} = \frac{1}{5} \cdot \begin{bmatrix} 25 & 15 & 10 \\ 0 & 5 & -5 \\ 15 & 0 & 10 \end{bmatrix} = A_1 = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 1 & -1 \\ 3 & 0 & 2 \end{bmatrix}$$

Zweite Rotation:

$$a = 5; \quad b = 3$$

$$c = \frac{5}{\sqrt{5^2 + 3^2}}; \quad s = \frac{3}{\sqrt{25 + 9}}$$

$$G_{3,1} = \frac{1}{\sqrt{34}} \cdot \begin{bmatrix} 5 & 0 & 3 \\ 0 & \sqrt{34} & 0 \\ -3 & 0 & 5 \end{bmatrix}$$

$$G_{3,1} \cdot A_1 = \frac{1}{\sqrt{34}} \cdot \begin{bmatrix} 5 & 0 & 3 \\ 0 & \sqrt{34} & 0 \\ -3 & 0 & 5 \end{bmatrix} \cdot \begin{bmatrix} 5 & 3 & 2 \\ 0 & 1 & -1 \\ 3 & 0 & 2 \end{bmatrix} = \frac{1}{\sqrt{34}} \begin{bmatrix} 34 & 15 & 16 \\ 0 & \sqrt{34} & -\sqrt{34} \\ 0 & -9 & 4 \end{bmatrix}$$

Die Zahlen sind jetzt ein bisschen arg hässlich geworden, aber hier würde man nur noch die dritte Rotation machen müssen (mit $G_{3,2}$) und man wäre fertig.

7.6 Lineares Ausgleichsproblem

Erklärung

Unter einem linearen Ausgleichsproblem versteht man z.B. die Problemstellung, eine Gerade zu finden, die durch alle Stützpunkte hindurchgeht. Das Problem: Wenn man mehr als 2 Punkte hat und diese nicht zufälligerweise genau eine Gerade bilden ist das unlösbar. Man sucht also eine Gerade, die **möglichst nah** durch alle Stützpunkte geht (aka möglichst exakt ist).

Wiederholung Geraden Gleichung:

$$y = m \cdot x + t \quad (25)$$

Unser Problem lässt sich in ein überbestimmtes LGS darstellen: $Ax = b$ mit $A \in \mathbb{R}^{m \times n}$ und $m > n$. Solch ein überbestimmtes LGS können wir nicht (perfekt) lösen. $Ax = b \implies$ nicht direkt lösbar!

Wir können jedoch das Ergebnis annähern: $Ax' \approx b$.

Mit $a := Ax'$ als skalierbarer Vektor kann man das Problem grafisch folgendermaßen darstellen:

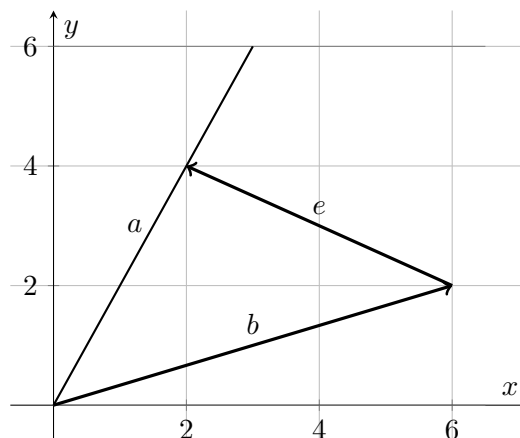


Abbildung 3: Veranschaulichung eines linearen Ausgleichsproblems

Wir möchten also ein x' so wählen, dass $Ax' \approx b$. Als Maß dient uns (auf die Grafik bezogen!) die Distanz des Vektors b zu einer beliebigen Stelle von a (also $e = b - Ax'$). Die kürzeste Distanz haben wir, wenn wir vom Vektor a senkrecht zu b »laufen«, wir nennen diesen Vektor e (siehe Grafik). Wir versuchen also e , der ja die Distanz und damit auch den Fehler angibt, zu **minimieren**.

Weil e senkrecht auf a steht, wissen wir, dass $e \cdot a = 0$ (Skalarprodukt!) gilt.

Folgender (rechnerischer) Gedankengang:

$$\begin{aligned}
 e &= b - x' \cdot a \\
 \min \|e\|_2 &= \min \|b - x' \cdot a\|_2 \\
 \implies e \cdot a &= 0 \\
 (b - x' \cdot a) \cdot a &= 0 \\
 a \cdot b - a \cdot x' \cdot a &= 0 \\
 a \cdot b &= a \cdot x' \cdot a \\
 a^T \cdot b &= a^T \cdot a \cdot x'
 \end{aligned}$$

Man nennt sie die **Normalengleichung**:

$$A^T \cdot b = A^T \cdot A \cdot x' \quad (26)$$

Das ist natürlich allgemeingültig, sprich für höhere Dimensionen ist A halt eine größere Matrix etc. . . ; Die Rechnung und Formel bleibt aber dieselbe!

Vorgehen

Lösen von linearen Ausgleichsproblemen

Die z.B. drei Punkte p_1, p_2 und p_3 haben die Koordinaten (x_1, y_1) bis (x_3, y_3) .

Folgende Matrix aufstellen:

$$\begin{bmatrix} A \end{bmatrix} \cdot \begin{pmatrix} t \\ m \end{pmatrix} = \begin{pmatrix} b \end{pmatrix}$$

Die erste Spalte von A sind nur Einsen (in der Geradengleichung ist t immer genau einmal vorhanden) und die zweite die jeweiligen x Koordinaten der Punkte. In b stehen jeweils die y Werte der Punkte.

Damit lässt sich obige Normalengleichung durch normale Operationen lösen:

$$A^T \cdot b = A^T \cdot A \cdot x'$$

In x' stehen dann unsere Ergebnisse für t und m drin. Wir stellen unsere Ergebnisgerade mit folgender Gleichung auf:

$$y = m \cdot x + t \quad (27)$$

Beispiel

Wir möchten eine Gerade so bestimmen, dass sie möglichst nahe die folgenden Punkte durchläuft:

$$P_1 = (1, 2); \quad P_2 = (2, 4); \quad P_3 = (3, 3)$$

Mit der Geradenfunktion stellen wir folgende Gleichungen auf:

$$y = m \cdot x + t$$

$$2 = m \cdot 1 + t$$

$$4 = m \cdot 2 + t$$

$$3 = m \cdot 3 + t$$

Aufstellen des LGS:

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \cdot \begin{pmatrix} t \\ m \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix}$$

Hier sehen wir schon, dass wir das nicht einfach mit Gauß oder LR Zerlegung lösen können!

Mit Normalengleichung:

$$A^T \cdot b = A^T \cdot A \cdot x'$$

$$\begin{aligned} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \cdot \begin{pmatrix} 2 \\ 4 \\ 3 \end{pmatrix} &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \cdot \begin{pmatrix} t \\ m \end{pmatrix} \\ \Rightarrow \begin{pmatrix} 9 \\ 19 \end{pmatrix} &= \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix} \cdot \begin{pmatrix} t \\ m \end{pmatrix} \end{aligned}$$

Und das können wir dann wieder mit einer beliebigen LGS-Lösungsmethode ausrechnen:

$$\left[\begin{array}{cc|c} 3 & 6 & 9 \\ 6 & 14 & 19 \end{array} \right] = \left[\begin{array}{cc|c} 3 & 6 & 9 \\ 0 & 2 & 1 \end{array} \right]$$

Rückwärtssubstitution ergibt den Lösungsvektor:

$$(2, 0.5)^T$$

Damit hätten wir als Lösung folgende Gerade:

$$y = \frac{1}{2}x + 2$$

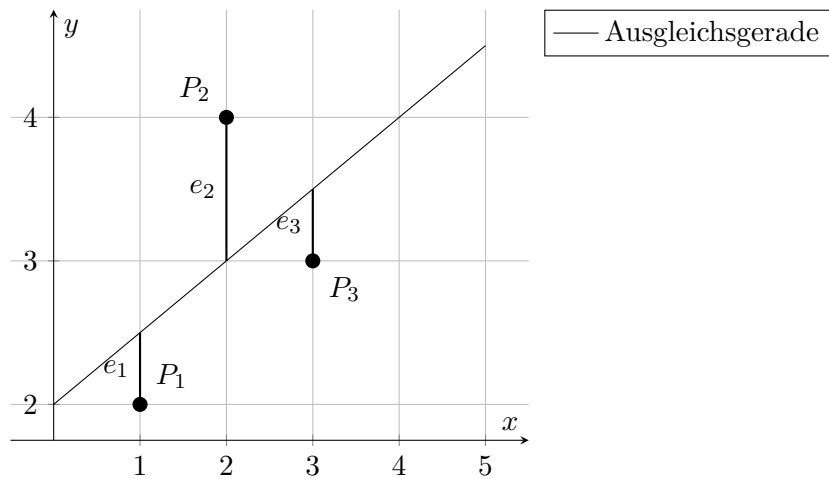


Abbildung 4: Die Ausgleichsgerade sowie die drei für die Rechnung verwendeten Punkte und die einzelnen Fehler (nur die y -Differenz!).

Wichtig

$$\text{cond}(A^T \cdot A) = \text{cond}(A)^2 \quad (28)$$

Obwohl wir einen stabilen Algorithmus haben kann sich auf dem Weg dahin die Kondition enorm verschlechtert haben!

7.6.1 Lösung mit QR-Zerlegung

Erklärung

Wenn man die QR-Zerlegung für das lineare Ausgleichsproblem nutzt, erkennt man, dass es viel stabiler wird (wir arbeiten mit orthogonalen Matrizen, das ist immer super ☺). Im Endeffekt rotiert man hier genauso wie bei der normalen QR-Zerlegung so, dass alles unter der Diagonalen zu null wird.

Dadurch ergibt sich dasselbe Minimierungsproblem und die Lösung ergibt sich durch:

$$Rx = \beta_1, \text{ wobei } \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} := Q^T \cdot b$$

Die rechte Seite der Gleichung entspricht **NICHT** einem Bruch, sondern der Aufteilung des Vektors in einen oberen ($= \beta_1$) und unteren ($= \beta_2$) Teil.

Dabei kann in β_2 zwar etwas anderes stehen als Nullen, es interessiert uns aber gar nicht.

Denn sobald wir die QR-Zerlegung gemacht haben, können wir den Bereich streichen, den das LGS überbestimmt macht. Sprich wir streichen den unteren Teil der Matrizen, sodass sie wieder quadratisch wird und wir das Ergebnis ausrechnen können.

Wichtig

Dieses Prinzip sowie die Normalengleichung funktioniert mit beliebigen überbestimmten LGS (weil das alle lineare Ausgleichsprobleme im Grunde sind).

Beispiel

Wir möchten folgendes überbestimmtes LGS mithilfe von QR-Zerlegung lösen:

$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 0 & 4 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 6 \end{pmatrix}$$

Wir konstruieren uns nach [Kapitel 7.5.2](#) die Matrix $G_{3,2}$, weil die anderen Positionen schon Nullen sind und wir nur noch die Zahl aus der zweiten Spalte, dritten Zeile »wegrotieren« müssen.

$$G_{3,2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_a & s \\ 0 & -s_b & c \end{bmatrix}$$

a und b auslesen:

$$\begin{aligned} a &= 3; \quad b = 4 \\ c &= \frac{a}{\sqrt{a^2 + b^2}} = \frac{3}{\sqrt{3^2 + 4^2}} = \frac{3}{\sqrt{25}} \\ s &= \frac{b}{\sqrt{a^2 + b^2}} = \frac{4}{5} \end{aligned}$$

Jetzt rotieren mit Matrix A :

$$G_{3,2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix}$$

$$G_{3,2} \cdot A = A_1$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 5 \\ 0 & 0 \end{bmatrix}$$

Jetzt haben wir schon die Matrix A in Q und R zerteilt (da einfach gewählt ☺)

$$Q = G_{3,2}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & -4/5 \\ 0 & 4/5 & 3/5 \end{bmatrix}; \quad R = \begin{bmatrix} 1 & 2 \\ 0 & 5 \\ 0 & 0 \end{bmatrix}$$

Jetzt noch lösen:

$$Q^T \cdot b = Rx$$

$$Q^T \cdot b = G_{3,2} \cdot b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/5 & 4/5 \\ 0 & -4/5 & 3/5 \end{bmatrix} \cdot \begin{pmatrix} 4 \\ 2 \\ 6 \end{pmatrix} = \begin{pmatrix} 4 \\ 30/5 \\ 10/5 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 2 \end{pmatrix}$$

Gesamte Gleichung aufstellen:

$$Rx = Q^T \cdot b$$

$$\begin{bmatrix} 1 & 2 \\ 0 & 5 \\ 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 2 \end{pmatrix}$$

Genau hier (und sonst nie) dürfen wir alle Zeilen streichen, die das Problem überbestimmt machen:

$$\begin{bmatrix} 1 & 2 \\ 0 & 5 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix}$$

Das ist nach Gauß lösbar und wir erhalten als Ergebnis unser x_1 und ein x_2 . Wenn das ein lineares Ausgleichsproblem wäre, dann entspräche x_1 dem t und x_2 dem m , mit der wir dann unsere Geradenfunktion aufstellen könnten.

Vertiefung

Beweis, dass die QR-Zerlegung hier nichts anderes ausrechnet als die Normalengleichung:

$$A^T A x = A^T b$$

$$A = QR$$

$$Q^T Q = \mathbb{1}$$

Wenn wir das jetzt miteinander einsetzen:

$$R^T Q^T Q R x = R^T Q^T b$$

$$R^T R x = R^T Q^T b$$

$$R x = Q^T b$$

$$R x = \beta_1$$

8 Matrixnorm

Eine Matrix kann den relativen Eingabefehler beeinflussen. Um Fehler bei Vektoren und Matrizen messen zu können, brauchen wir Normen.

8.1 Euklidische Norm

Erklärung

Die euklidische Norm wurde früher schon einmal beigebracht, deswegen nur eine kurze Wiederholung. Diese Norm gilt nur für Vektoren und entspricht der Länge eines Vektors:

$$\|x\|_2 = \sqrt{x_0^2 + \dots + x_n^2}$$

So wissen wir zum Beispiel, wenn wir mit einem Vektor der Länge eins normiert rechnen, eine Operation ausführen die die Länge nicht ändert und diese trotzdem nicht mehr 1 beträgt, dass sich ein Arithmetik Fehler eingeschlichen haben muss.

8.2 Spektralnorm

Erklärung

Oder auch Matrix-Grenzen-Norm-2 genannt, beschreibt folgende Normierung für beliebige Matrizen:

$$\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \quad (29)$$

Vorgehen

Kleiner Trick:

Das werden wir zwar nicht beweisen, aber folgende Gleichungen gelten:

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} \quad (30)$$

Für symmetrische Matrizen:

$$\|A\|_2 = \max_{i \in [n]} |\lambda_i| \quad (31)$$

Für Diagonalmatrizen:

$$\|A\|_2 = \max_{i \in [n]} |A_{i,i}| \quad (32)$$

Eigenwert der Inversen:

$$\lambda_i(A) \neq 0 \quad \forall i \implies \lambda_i(A^{-1}) = \frac{1}{\lambda_i} \quad (33)$$

8.3 Rechen Eigenschaften

Erklärung

Mit Normierungen müssen wir natürlich auch rechnen können, deswegen hier eine schnelle Übersicht über alle Eigenschaften.

Mit Skalar a , Vektor v , Matrix A, B und X, Y als Vektor oder Matrix gilt

$$\begin{aligned} \|aX\|_2 &= \|a\| \cdot \|X\|_2 \\ \|X + Y\|_2 &\leq \|X\|_2 + \|Y\|_2 \\ \|A \cdot B\|_2 &\leq \|A\|_2 \cdot \|B\|_2 \\ \|A \cdot v\|_2 &\leq \|A\|_2 \cdot \|v\|_2 \end{aligned}$$

8.4 Kondition einer Matrix

Erklärung

Mit der Einführung von Normen können wir nun auch noch die Kondition bzw. die Konditionszahl (siehe [Kapitel 5](#)) einer invertierbaren Matrix A bestimmen.

Vorgehen

Die Kondition der Matrix A bezüglich der euklidischen Norm ist gegeben durch

$$\text{cond}(A) = \|A^{-1}\|_2 \cdot \|A\|_2 \quad (34)$$

Außerdem kann man die Kondition der Matrix A berechnen, indem man den größten Singulärwert der Matrix durch den kleinsten teilt.

Falls die Matrix symmetrisch positiv definit ist, kann man obiges vereinfachen zu:

$$\text{cond}(A) = \left| \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \right| \quad (35)$$

wobei λ hierbei Eigenwerte beschreiben. $\lambda_{\max}(A)$ ist also der größte Eigenwert der Matrix A , $\lambda_{\min}(A)$ der kleinste.

8.5 Kondition eines LGS

Erklärung

Ein LGS hat die Form $Ax = b$. Auch hiervon können wir die Kondition mithilfe der Normen bestimmen. Wir definieren $\tilde{x} := x + \delta x$ als eine gestörte bzw. Näherungslösung des LGS. $\tilde{b} := b + \delta b$ ist eine gestörte rechte Seite des LGS.

Vorgehen

Störung der rechten Seite

Sei die rechte Seite des LGS gestört, erhalten wir eine Näherungslösung:

$$A\tilde{x} = \tilde{b} \quad (36)$$

Nun kann man die Fehler e und r beschreiben mit

$$e := x - \tilde{x} \quad (37)$$

$$r := b - \tilde{b} = Ae \quad (38)$$

Dann gilt:

$$f_{\text{rel}} = \frac{\|e\|}{\|x\|} \quad (39)$$

$$\frac{\|r\|}{\|b\| \cdot \text{cond}(A)} \leq \frac{\|e\|}{\|x\|} \leq \text{cond}(A) \frac{\|r\|}{\|b\|} \quad (40)$$

9 Interpolation

Erklärung

Interpolation bedeutet im Grunde, eine unbekannte Funktion aufgrund von gegebenen Teilinformationen zu erraten bzw. mathematisch würde man natürlich von annähern sprechen.

Formal: Finde zu n Stützstellen $(x_i, f(x_i))$ einer unbekannten Funktion $f(x)$ eine Funktion $p(x)$, mit

$$p(x_i) = f(x_i) \quad \forall x_i \quad (i = 0, \dots, n-1)$$

Das bedeutet nur, dass eine Interpolation von Punkten $p(x)$ einer Funktion $f(x)$ durch alle Stützpunkte »hindurchgeht«. Da wir ja sonst von keinerlei Informationen über $f(x)$ ausgehen ist das natürlich der beste Weg, weil zumindest diese Punkte stimmen und der Rest approximiert wird.

9.1 Polynominterpolation

Erklärung

Jetzt soll es erst einmal nur um die Polynominterpolation gehen, also eine Annäherung einer unbekannten Funktion durch eine Polynomfunktion. Andere Methoden wären mit trigonometrischen Funktionen (siehe [Kapitel 11](#)) und linearer Interpolation (siehe [Kapitel 9.3](#)).

Unsere Ergebnisse sind, wie der Name schon impliziert, immer Polynome und damit im Gegensatz zur linearen Interpolation stetig und beliebig häufig stetig differenzierbar.

Wiederholung Polynom k 'ten Grades:

$$c_k x^k + c_{k-1} x^{k-1} + \dots + c_1 x^1 + c_0$$

Wir können das Problem leicht in ein LGS übersetzen, indem wir einfach ein allgemeines Polynom $(n-1)$ 'ten Grades aufstellen und unsere Stützpunkte einsetzen. Wir definieren mit $y_i = f(x_i)$ die y Koordinaten der Stützpunkte (auch Stützwerte genannt).

$$\begin{bmatrix} x_0^0 & x_0^1 & \dots & x_0^{n-1} \\ x_1^0 & x_1^1 & \dots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1}^0 & x_{n-1}^1 & \dots & x_{n-1}^{n-1} \end{bmatrix} \cdot \begin{pmatrix} c_0 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

Die Lösungsfunktion (Interpolationsfunktion genannt) erhalten wir, indem wir die ausgerechneten Koeffizienten c_0 bis c_{n-1} in unsere allgemeine Polynomfunktion $(n -$

1)'ten Grades einsetzen. Also erhalten wir für n Stützstellen immer ein Polynom $(n-1)$ 'ten Grades!

9.1.1 Basisfunktionen

Erklärung

Das Kalkulieren mit Basisfunktionen (und damit mithilfe eines LGS) liegt in $\mathcal{O}(n^3)$, für die Auswertung der entstehenden Funktion $\mathcal{O}(n)$. Unter den Standard Polynom Basisfunktionen versteht man

$$\begin{array}{ll} g_0(x) = 1 & g_1(x) = x \\ g_2(x) = x^2 & g_3(x) = x^3 \\ \vdots & \end{array}$$

Vorgehen

Unser Problem können wir als LGS mit allgemeinen Basisfunktionen darstellen. Wir haben immer genauso-viele Gleichungen (n Stützstellen) wie wir Unbekannte ausrechnen möchten, da der Grad des resultierenden Polynoms mit steigendem n linear korreliert.

Umgerechnet mit beliebigen Basisfunktionen g_0 bis g_{n-1} :

$$\begin{bmatrix} g_0(x_0) & g_1(x_0) & \dots & g_{n-1}(x_0) \\ g_0(x_1) & \ddots & \ddots & g_{n-1}(x_1) \\ \vdots & \ddots & \ddots & \vdots \\ g_0(x_{n-1}) & \dots & \dots & g_{n-1}(x_{n-1}) \end{bmatrix} \cdot \begin{pmatrix} c_0 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} \quad (41)$$

Die Interpolationfunktion und damit Lösung des Problems entspricht dann:

$$p(x) = c_0 \cdot g_0(x) + c_1 \cdot g_1(x) + \dots + c_{n-1} \cdot g_{n-1}(x) \quad (42)$$

Beispiel

Eine unbekannte Funktion f durchläuft folgende Punkte:

$$f(1) = 0; \quad f(2) = 2$$

Wir haben also ein $n = 2$ und berechnen ein Polynom $(n - 1) = 1$ 'ten Grades. Wir sollen das Interpolationspolynom mithilfe der Standard Polynom Basisfunktionen aufstellen und damit den Funktionswert an der Stelle 1.5 abschätzen. Bis Grad 1 sind die Basisfunktionen also $g_0(x) = 1$ und $g_1(x) = x$.

Stellen wir unseres LGS auf:

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

Mit Gauß aufgelöst erhalten wir

$$c = (-2, 2)^T$$

Das wiederum (mit den Basisfunktionen) ergibt die Funktion

$$p(x) = c_0 \cdot g_0(x) + c_1 \cdot g_1(x)$$

$$p(x) = c_0 \cdot 1 + c_1 \cdot x$$

$$p(x) = (-2) + 2 \cdot x$$

Jetzt noch einsetzen:

$$p(1.5) = -2 + 3 = 1$$

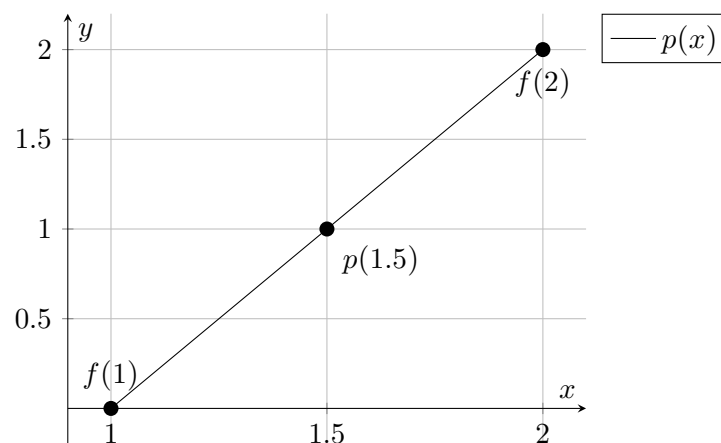


Abbildung 5: Das Beispiel grafisch veranschaulicht.

9.1.2 Lagrange-Basis**Erklärung**

Die Lagrange-Basis soll eine Möglichkeit sein, »schlaue« Basisfunktionen zu erstellen. Folgendermaßen wird sie berechnet:

$$L_j = \prod_{i=0; i \neq j}^{n-1} \frac{x - x_i}{x_j - x_i} \quad (43)$$

Wenn man die L_j nun als Basisfunktionen g_0, \dots, g_{n-1} verwendet und das LGS lösen will, stellt man fest, dass die linke Matrix immer der Einheitsmatrix $\mathbb{1}$ entspricht und das Lösen des LGS trivial macht. Das liegt daran, dass die »schlau« Basisfunktionen eben nur an der Stelle der jeweiligen Stützstelle 1 beträgt und an allen anderen 0. Ohne das LGS zu berechnen entsteht also die Interpolationspolynomfunktion

$$p(x) = \sum_{j=0}^{n-1} y_j \cdot L_j(x) \quad (44)$$

Hier sehen wir, dass das Aufstellen der Funktion faktisch gar nichts kostet (bzw. $\mathcal{O}(n)$), das Auswerten der Interpolationsfunktion jedoch $\mathcal{O}(n^2)$!

Beispiel

Wir haben die Punktemenge von Stützstellen gegeben mit:

$$P : \{(0, 1), (1, 4), (3, -2)\}$$

Wir möchten mithilfe von Lagrange-Basisfunktionen das Interpolationspolynom aufstellen:

$$\begin{aligned} L_0 &= \frac{(x-1)(x-3)}{(0-1)(0-3)} = \frac{(x-1)(x-3)}{3} \\ L_1 &= \frac{(x-0)(x-3)}{(1-0)(1-3)} = \frac{x \cdot (x-3)}{-2} \\ L_2 &= \frac{(x-0)(x-1)}{(3-0)(3-1)} = \frac{x \cdot (x-1)}{6} \end{aligned}$$

Jetzt noch die Interpolationsfunktion ausrechnen:

$$\begin{aligned} p(x) &= 1 \cdot L_0 + 4 \cdot L_1 - 2 \cdot L_2 \\ &= 1 \cdot \frac{(x-1)(x-3)}{3} + 4 \cdot \frac{x \cdot (x-3)}{-2} - 2 \cdot \frac{x \cdot (x-1)}{6} \\ &= 1 + 5 \cdot x - 2 \cdot x^2 \end{aligned}$$

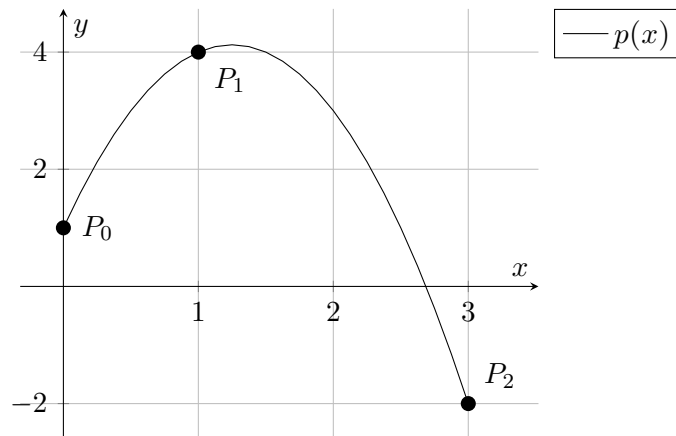


Abbildung 6: Das Beispiel grafisch veranschaulicht.

Wichtig

Ein Interpolationspolynom ist stets **eindeutig**! Egal ob man die Standard Polynom Basisfunktionen, die Lagrange Basis oder so verwendet kommt zu einem Problem immer dasselbe Polynom raus. Auf Genauigkeit brauchen wir also keinen dieser Algorithmen untersuchen, jedoch aber die Anzahl an darstellbaren Polynomgraden (im Moment $n - 1$ für n Stützstellen).

9.1.3 Interpolation nach Newton

Erklärung

Die Interpolation nach Newton ist ein effizienterer Algorithmus, ein Interpolationspolynom analytisch auszurechnen, denn das Verfahren braucht für das Berechnen der Koeffizienten nur $\mathcal{O}(n^2)$ und das Auswerten der Funktion auch nur $\mathcal{O}(n)$.

Vorgehen

Wir definieren:

$$c_{i,0} = f(x_i) = y_i \quad (45)$$

$$c_{i,k} = \frac{c_{i+1,k-1} - c_{i,k-1}}{x_{i+k} - x_i} \quad (46)$$

Informeller ist folgendes:

$$[\text{neu}]_c = \frac{f_{\text{linksUnten}} - f_{\text{links}}}{x_{\text{ganzLinksUnten}} - x_{\text{links}}} \quad (47)$$

Dabei sind die Richtungen immer in Relation zu der Stelle des zu ausrechnenden Wertes in der Tabelle angegeben. »ganzLinksUnten« meint, dass man in der Tabelle soweit nach ganz links unten zieht, bis man mit dem nächsten »Links-Unten-Move« nicht mehr in der rechten Seite der Tabelle wäre. Von dort aus nimmt man den x-Wert ganz links in derselben Zeile.

x_i	$i \setminus k$	0	1	2	...
x_0	0	$c_{0,0}$	$c_{0,1}$	$c_{0,2}$...
x_1	1	$c_{1,0}$	$c_{1,1}$	\ddots	
x_2	2	$c_{2,0}$	\ddots		
\vdots		\ddots			

Tabelle 9: Tabelle zum Ausrechnen

Wenn wir das fertig aufgestellt haben, können wir direkt alle Koeffizienten ablesen:

$$p(x) = c_{0,0} + c_{0,1} \cdot (x - x_0) + \cdots + c_{0,n-1} \cdot \prod_{i=0}^{n-2} (x - x_i) \quad (48)$$

Unsere Koeffizienten sind also einfach die Werte der ersten Zeile der Tabelle.

Beispiel

Wir haben:

x_i	$i \setminus k$	0	1	2
0	0	0	$c_{0,1}$	$c_{0,2}$
2	1	4	$c_{1,1}$	
3	2	9		

$$c_{0,1} = \frac{4 - 0}{2 - 0} = 2$$

$$c_{1,1} = \frac{9 - 4}{3 - 2} = 5$$

$$c_{0,2} = \frac{5 - 2}{3 - 0} = 1$$

x_i	$i \setminus k$	0	1	2
0	0	0	2	1
2	1	4	5	
3	2	9		

Also kommt als Interpolationsfunktion folgendes heraus:

$$\begin{aligned}
 p(x) &= c_{0,0} + c_{0,1} \cdot (x - x_0) + c_{0,2} \cdot (x - x_0)(x - x_1) \\
 &= 0 + 2 \cdot (x - 0) + 1 \cdot (x - 0)(x - 2) \\
 &= 0 + 2x + 1 \cdot x^2 - 2x \\
 &= x^2
 \end{aligned}$$

Würden wir hier die Punkte oben betrachten, sehen wir schon, dass es auf x^2 hinauslaufen wird.

Erklärung

Es ist vielleicht schon zu erkennen, aber wenn jetzt noch ein neuer Stützpunkt hinzugefügt werden soll, kann man diesen in der Tabelle einfach darunter schreiben und den neuen Koeffizienten relativ schnell ausrechnen. Dann muss nur noch das Ergebnis angepasst werden, indem ein weiteres Summenglied hinzugefügt wird und wir sind schon fertig. Das ist ein **großer Vorteil** dieses Verfahrens.

Beispiel

Wir erweitern letztes Beispiel um einen weiteren Stützpunkt, und zwar mit $(-2, 4)$.

x_i	$i \setminus k$	0	1	2	3
0	0	0	2	1	$c_{0,3} = ?$
2	1	4	5	?	
3	2	9	?		
-2	3	4			

Wir müssen die mit Fragezeichen markieren Stellen noch ausrechnen, um auf den neuen Koeffizienten $c_{0,3}$ schließen zu können.

$$c_{2,1} = \frac{4 - 9}{-2 - 3} = 1$$

$$c_{1,2} = \frac{1 - 5}{-2 - 2} = 1$$

$$c_{0,3} = \frac{1 - 1}{-2 - 0} = 0 = c_3$$

Insgesamt sieht die Tabelle also folgendermaßen aus:

x_i	$i \setminus k$	0	1	2	3
0	0	0	2	1	0
2	1	4	5	1	
3	2	9	1		
-2	3	4			

Jetzt noch die Interpolationsfunktion ausrechnen:

$$\begin{aligned}
 p(x) &= c_{0,0} + c_{0,1} \cdot (x - x_0) + c_{0,2} \cdot (x - x_0)(x - x_1) + c_{0,3} \cdot (x - x_0)(x - x_1)(x - x_2) \\
 &= 0 + 2 \cdot (x - 0) + 1 \cdot (x - 0)(x - 2) + 0 \\
 &= 0 + 2x + 1 \cdot x^2 - 2x \\
 &= x^2
 \end{aligned}$$

Das Ergebnis ist logisch, da der neue Punkt $(-2, 4)$ auch auf der Funktion x^2 liegt. Die Hinzunahme dieses Stützpunktes dürfte unser Ergebnis gar nicht abändern! Das können wir außerdem daher ablesen, dass der neu berechnete Koeffizient 0 ist und damit keinen Einfluss auf die Ergebnisfunktion hat.

9.1.4 Aitken-Neville

Erklärung

Dieses Verfahren dient dazu, ein Interpolationspolynom **direkt an einer Stelle x auszuwerten**, obwohl wir das Polynom selbst nie ausgerechnet haben. Aitken-Neville rechnet das Polynom auch nicht direkt aus.

Das machen wir, indem wir eine Tabelle ähnlich der bei Newton aufstellen:

Vorgehen

x_i	$i \setminus k$	0	1	2	...
x_0	0	$p[0,0] = y_0$	$p[0,1]$	$p[0,2]$...
x_1	1	$p[1,0] = y_1$	$p[1,1]$	\ddots	
x_2	2	$p[2,0] = y_2$	\ddots		
\vdots		\ddots			

Tabelle 10: Berechnungstabelle für diese Methode

mit der Formel:

$$p[i, k] := p[i, k-1] + \frac{x - x[i]}{x[i+k] - x[i]} \cdot (p[i+1, k-1] - p[i, k-1]) \quad (49)$$

Die **Lösung der Auswertung** steht in der Tabelle dann **rechts oben!**

Eine informelle Variante der Formel (mit derselben Notation wie bei Newton):

$$p_{\text{neu}} := p_{\text{links}} + \frac{x - x_{\text{links}}}{x_{\text{ganzLinksUnten}} - x_{\text{links}}} \cdot (p_{\text{linksUnten}} - p_{\text{links}}) \quad (50)$$

Erklärung

Wie bei Newton (und in diesem Fach bei jedem Dreiecksschema) können wir wieder einfach neue Stützpunkte unten an die Tabelle einfügen und müssen nicht neu anfangen.

Wenn wir das an einem Computer implementieren, können wir eine Laufzeit von $\mathcal{O}(n^2)$ herauslesen. Dieses Verfahren ist jedoch nur interessant, wenn man nur wenige Punkte auswerten will und die Funktion explizit nicht benötigt!

Beispiel

Es seien die Stützpunkte einer unbekannten Funktion $f(x)$ gegeben:

$$f(0) = 0; \quad f(1) = 1; \quad f(2) = 4$$

Wir möchten den Funktionswert an der Stelle $x = 0.5$ auswerten:

x_i	$i \setminus k$	0	1	2
$x_0 = 0$	0	$y_0 = 0$	$p[0, 1]$	$p[0, 2]$
$x_1 = 1$	1	$y_1 = 1$	$p[1, 1]$	
$x_2 = 2$	2	$y_2 = 4$		

Ausrechnen der unbekannten Werte in der Tabelle:

$$p_{\text{neu}} := p_{\text{links}} + \frac{x - x_{\text{links}}}{x_{\text{ganzLinksUnten}} - x_{\text{links}}} \cdot (p_{\text{linksUnten}} - p_{\text{links}})$$

$$\begin{aligned} p[0, 1] &= p[0, 0] + \frac{0.5 - x[0]}{x[1] - x[0]} \cdot (p[1, 0] - p[0, 0]) = \frac{1}{2} \\ p[1, 1] &= p[1, 0] + \frac{0.5 - x[1]}{x[2] - x[1]} \cdot (p[2, 0] - p[1, 0]) = -\frac{1}{2} \\ p[0, 2] &= p[0, 1] + \frac{0.5 - x[0]}{x[2] - x[0]} \cdot (p[1, 1] - p[0, 1]) = \frac{1}{4} \end{aligned}$$

Das Ergebnis ist also $f(0.5) = 0.25$, was auch Sinn ergibt, da unsere Stützpunkte recht eindeutig auf $f(x) = x^2$ hinweisen. Bei schwierigeren Stützpunkten wird Aitken-Neville aber ungenauer (genau so ungenau wie andere Polynominterpolation Methoden, siehe 9.1.2).

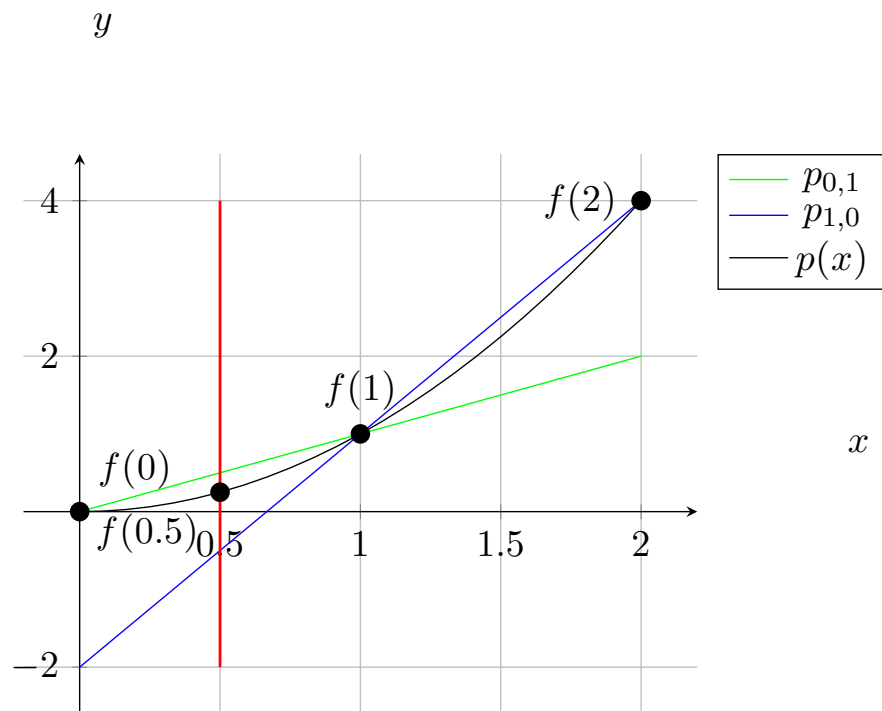


Abbildung 7: Das Ergebnis des Beispiels grafisch. Interpolation erst einmal zwischen jeweils zwei der Punkte ($p[0, 1]$ ist hierbei der Punkt bei $x = 0.5$ der grünen Linie, $p[1, 0]$ die blaue Linie).

Vertiefung

Wenn man in der Aitken-Neville Formel das x einfach als Variable lässt anstelle von einer bestimmten Stelle einzusetzen, wird das Ergebnis äquivalent zu der Interpolationsfunktion von Newton, Lagrange, ...

In diesem Fach behandeln wir Aitken-Neville aber so, wie wenn das nicht möglich wäre. Fragt mich nicht warum. ☺

9.1.5 Fehlerabschätzung

Vorgehen

So kann man den Interpolationsfehler an einer Stelle x abschätzen:

$$|f(x) - p(x)| = \left| \frac{f^{(n)}(\xi)}{(n)!} \cdot \prod_{k=0}^{n-1} (x - x_k) \right| \quad (51)$$

mit n Stützstellen und ξ entspricht einem Wert im Interpolationsbereich (kommt aus dem Mittelwertsatz). Man wählt ξ eigentlich immer so, dass der abgeschätzte Fehler maximal wird. Die Abschätzung des worst-case hat für uns die größte Aussagekraft.

$f^{(n)}$ steht hierbei für die n -te Ableitung von f .

Beispiel

Wir interpolieren eine Funktion mithilfe von 2 Stützpunkten an den Stellen 1 und 4. Damit ist der Interpolationsbereich definiert mit $x \in [1, 4]$ und wir wollen den Fehler an der Stelle $x = 2$ abschätzen.

Wir setzen in obige Formel ein:

$$\begin{aligned} |f(x) - p(x)| &= \left| \frac{f^{(2)}(\xi)}{(2)!} \cdot \prod_{k=0}^{2-1} (x - x_k) \right| \\ &= \left| \frac{f^{(2)}(\xi)}{(2)!} \cdot (2 - x_0)(2 - x_1) \right| \end{aligned}$$

Sei die zweite Ableitung gegeben, können wir das ganze ausrechnen:

$$f''(x) = \frac{21}{2} \cdot x$$

Da wir eine worst-case Abschätzung machen wollen, wählen wir $\xi \in [1, 4]$ so, dass die Fehlerabschätzung größtmöglich wird. Hier ist das offensichtlich mit $\xi = 4$ der Fall:

$$\begin{aligned} |f(x) - p(x)| &= \left| \frac{21 \cdot 4}{2 \cdot 2} \cdot (2 - 1)(2 - 4) \right| \\ &= |21 \cdot 1 \cdot -2| \\ &= 42 \end{aligned}$$

(Dass 42 rauskommt ist kein Zufall ☺)

9.2 Runge-Effekt

Erklärung

Wir haben uns noch nicht mit Problemen der Polynominterpolation beschäftigt. Ein klassisches Beispiel wäre der Runge Effekt.

Der Runge Effekt tritt dann auf, wenn man sich mit der Polynominterpolation von Polynomen hohen Grads beschäftigt und äquidistante Stützstellen verwendet. Äquidistant bedeutet hierbei einfach nur, dass diese gleich verteilt sind bzw. der Abstand von jeder Stützstelle zu ihren Nachbarn gleich groß ist. Bisher dachten wir, je mehr Stützstellen wir hinzufügen, desto besser wird auch die Qualität des Interpolationspolynoms.

Das ist nicht der Fall. Gerade die Ränder von Interpolationspolynomen mit vielen Stützstellen werden extrem ungenau, das nennen wir den Runge Effekt.

Folgende Erklärung: Wenn wir eine Funktion mit vielen äquidistanten Stützstellen haben, hat unser Interpolationspolynom zwangsläufig einen hohen Grad (ignorieren wir mal triviale Fälle). Wenn dieser hohe Grad informell aber nicht benötigt wird, kommt es zu Schwingungen, speziell an den Rändern des Interpolationsbereiches.

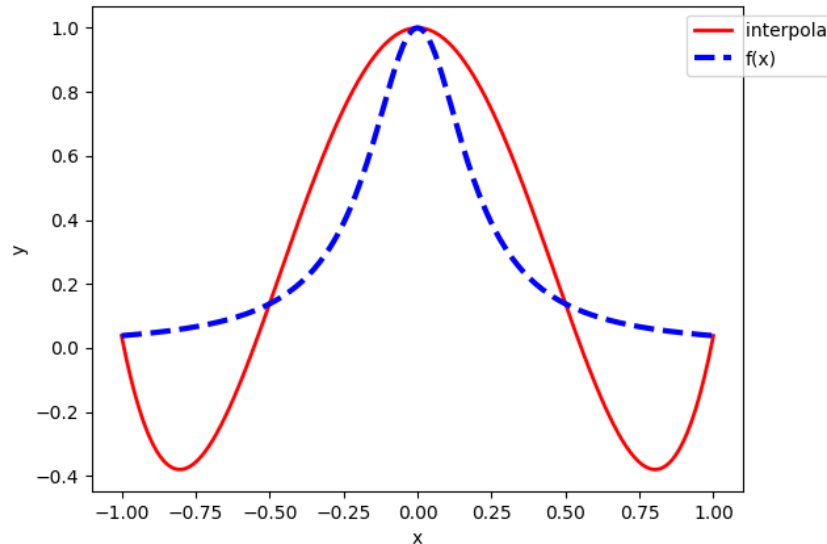


Abbildung 8: Interpolation mit wenigen Stützpunkten

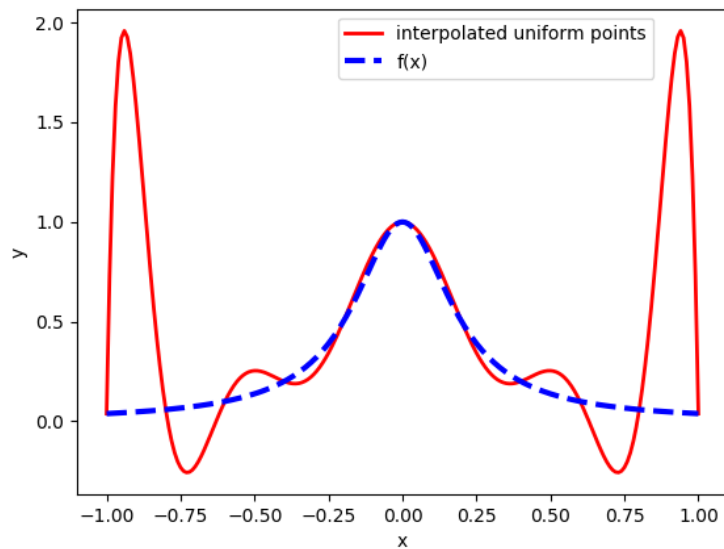


Abbildung 9: Dasselbe Problem mit vielen Stützpunkten

Erklärung

Welches der obigen Bilder zeigt die bessere Annäherung der Funktion (gestrichelte Linie) mithilfe von Polynominterpolation (durchgezogene Linie)?

Genau, das obere, welches deutlich weniger Stützpunkte nutzt...

9.3 Lineare Interpolation

Erklärung

Die lineare Interpolation ist denkbar einfach. Wir sollen ja eine Funktion finden, die durch alle Stützpunkte hindurchgeht. Also können wir doch benachbarte Stützpunkte einfach mit Geraden verbinden und fertig.

Das Ergebnis ist eine Sammlung an Teilfunktionen, deren Funktionsintervalle genau an den Rändern jeweils anknüpfen.

Vorgehen

Zwischen zwei benachbarten Stützpunkten (x_i, y_i) und (x_{i+1}, y_{i+1}) definieren wir eine lineare Verbindung mit

$$p_i(x) = (x - x_i) \cdot \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + y_i \quad x \in [x_i, x_{i+1}] \quad (52)$$

Machen wir das für alle $i = 0, \dots, n-1$, erhalten wir all solche Geraden, welche wir durch die Intervall Definitionen aneinander »kleben« können.

Beispiel

Wir haben die Stützpunkte $(0, 1)$, $(1, 3)$ und $(3, 0)$. Wir möchten diese stückweise linear interpolieren. Wir definieren unsere zwei Teilfunktionen mit obiger Methode:

$$\begin{aligned} p_0(x) &= (x - x_0) \cdot \frac{y_1 - y_0}{x_1 - x_0} + y_0 & x \in [x_0, x_1] \\ &= (x - 0) \cdot \frac{3 - 1}{1 - 0} + 1 & x \in [0, 1] \\ &= x \cdot 2 + 1 & x \in [0, 1] \\ \\ p_1(x) &= (x - x_1) \cdot \frac{y_2 - y_1}{x_2 - x_1} + y_1 & x \in [x_1, x_2] \\ &= (x - 1) \cdot \frac{0 - 3}{3 - 1} + 3 & x \in [1, 3] \\ &= (x - 1) \cdot -\frac{3}{2} + 3 & x \in [1, 3] \\ &= -\frac{3}{2}x + \frac{9}{2} & x \in [1, 3] \end{aligned}$$

Insgesamt könnten wir unsere Interpolationsfunktion (unser Ergebnis nennt man so) also folgendermaßen definieren:

$$g(x) = \begin{cases} p_0(x); & x \in [0; 1[\\ p_1(x); & x \in [1; 3] \end{cases}$$

Das bedeutet nichts anderes als, wenn man ein bestimmtes x in $g(x)$ einsetzt, die Teilfunktion (also p_0 oder p_1) ausgesucht wird, in welchem Intervall sich das x befindet. Wenn man beispielsweise jetzt $g(2)$ ausrechnen möchte, wird die zweite Teilfunktion ausgesucht, da $2 \in [1; 3]$ gilt.

$$g(2) = p_1(2) = -\frac{3}{2} \cdot 2 + \frac{9}{2}$$

Klar könnte man in der Definition von $g(x)$ oben noch $p_0(x)$ sowie $p_1(x)$ einsetzen aber das spare ich mir mal ☺.

Grafisch sähe das dann so aus:

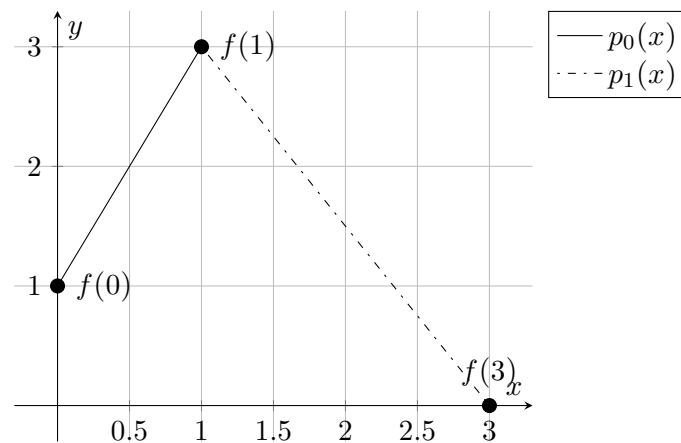


Abbildung 10: Stückweise lineare Interpolation grafisch.

Wichtig

Die erstellten Funktionen durch stückweise lineare Interpolation haben immer Knicke! Das bedeutet die Funktion ist nicht stetig differenzierbar!

9.4 Hermite Interpolation

Erklärung

Ein Problem, welches wir mit einer Interpolationsfunktion bekommen könnten: Sie ist nicht stetig differenzierbar (wie stückweise lineare Interpolation, siehe [Kapitel 9.3](#)).

Dieses Problem löst die Hermite Interpolation als netten Nebeneffekt, da sie für die Interpolation auch die Ableitung der Stützstellen einberechnet. Eigentlich ist sie »nur« eine genauere Interpolations Möglichkeit.

Schonmal gleich im Voraus, die Hermite Interpolation ist mit einer Laufzeit von $\mathcal{O}(n^3)$ aber relativ kostenaufwändig.

Mit der Standard Polynom Funktion dritten Grades

$$p(t) = c_3 \cdot t^3 + c_2 \cdot t^2 + c_1 \cdot t + c_0 \quad (53)$$

und dessen Ableitung können wir ein LGS aufstellen und lösen. Also wir verwenden nicht nur die Position von den Stützpunkten, sondern auch deren Ableitungen. Im konkreten Hermite Fall zwei Stützstellen und deren Ableitungen, weswegen wir auf einen Grad von 3 kommen.

Vorgehen

- LGS mithilfe von Stützstellen und deren Ableitung aufstellen.
- LGS lösen.
- Ergebnis des LGS in allgemeine Polynomfunktion einsetzen.
- Nach allen y und y' sortieren, also sodass da $y_0 \cdot \text{»irgendwas«} + y_1 \cdot \text{»irgendwas«} + \dots$ steht.
- Mithilfe von Koeffizientenvergleich auf die Basisfunktionen schließen, nach dem letzten Punkt entsprechen die »irgendwas« den Basisfunktionen.
- Hermite-Funktion (sortiert mit kubischen Basispolynomen):

$$p(t) = y_0 \cdot H_0(t) + y_1 \cdot H_1(t) + y'_0 \cdot H_2(t) + y'_1 \cdot H_3(t) \quad \in [0; 1] \quad (54)$$

Beispiel

Wir haben eine unbekannte Funktion $f(x)$ gegeben.
 Folgende Informationen seien über $f(x)$ bekannt:

$$f(0) = 0; \quad f(1) = 2; \quad f'(0) = 2; \quad f'(1) = 1$$

Terme aufstellen (mithilfe der Funktion eines allg. Polynom dritten Grades):

$$f(t) = c_3 \cdot t^3 + c_2 \cdot t^2 + c_1 \cdot t + c_0$$

$$f(0) = c_0 \stackrel{!}{=} y_0 = 0$$

$$f(1) = c_3 + c_2 + c_1 + c_0 \stackrel{!}{=} y_1 = 2$$

Jetzt müssen wir in die Ableitung einsetzen:

$$f'(t) = 3c_3 \cdot t^2 + 2c_2 \cdot t + c_1$$

$$f'(0) = c_1 \stackrel{!}{=} y'_0 = 2$$

$$f'(1) = 3 \cdot c_3 + 2 \cdot c_2 + c_1 \stackrel{!}{=} y'_1 = 1$$

Das entspricht folgender Vektor-Matrix-Schreibweise und damit LGS:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y'_0 \\ y'_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

LGS lösen (Gauß-Elimination) ergibt:

$$c_0 = 0; \quad c_1 = 2; \quad c_2 = 1; \quad c_3 = -1$$

Die Interpolationsfunktion entspricht also:

$$p(t) = -t^3 + t^2 + 2t + 0$$

Nach den Stützwerten und deren Ableitung sortieren, um auf die kubischen Basisfunktionen zu schließen wurde hier nicht (mehr) ausgeführt.

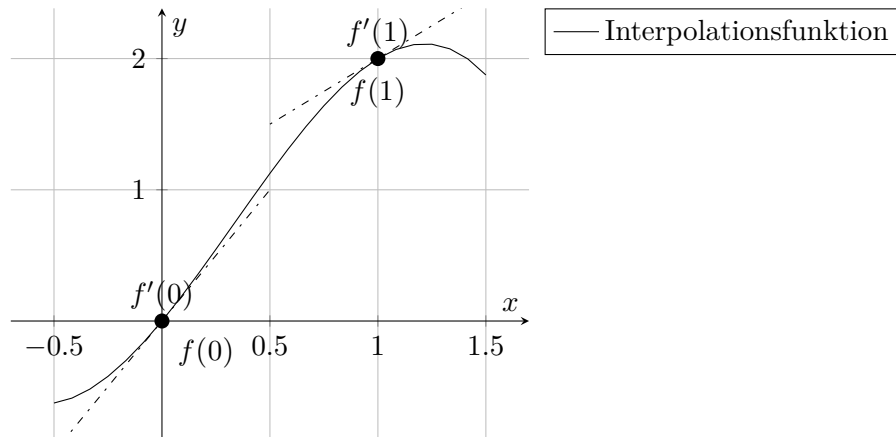


Abbildung 11: Obige Hermite-Interpolation grafisch. Wir verwenden nicht nur die Position der zwei Stützstellen, sondern auch deren Ableitung (hier als kurze gepunktete Linie eingezeichnet).

Vorgehen

Aus obigen Beispiel können wir eine neue Formel herleiten. Für zwei Stützstellen an den Positionen $x_0 = 0$, $x_1 = 1$ und deren Ableitungen entspricht das LGS:

$$\begin{bmatrix} 3 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} y'_1 \\ y_1 \\ y'_0 \\ y_0 \end{pmatrix} \quad (55)$$

Die berechneten c_0 bis c_3 können wir dann in unser Schema einsetzen

$$p(t) = c_3 \cdot t^3 + c_2 \cdot t^2 + c_1 \cdot t + c_0$$

und erhalten damit unser Ergebnis.

Vertiefung

Den letzten Methodenblock können wir natürlich jetzt etwas optimieren, da er immer gleich bleibt.

LR Zerlegung wollt ihr? Kein Problem! Ich erspar euch den langen Rechenweg, Fakt ist das Ergebnis entspricht

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 0 & 1/3 & 2/3 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Jetzt könnt ihr einfach das und euer Wissen aus [Kapitel 7.3](#) nutzen um so schneller Hermite Interpolationen durchzuführen. Für die Laufzeit eines Computerprogrammes hilft das natürlich herzlich wenig.

Die Zahlen der QR-Zerlegung sind leider weniger hübsch, sonst würde ich sie hier auch aufführen.

Um obige Korrektheit zu überprüfen, rechnen wir einfach die letzte Beispielbox mit-hilfe von LR Zerlegung. Zunächst müssen wir dafür $Ly = b$ lösen, also

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot y = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 0 \end{pmatrix}$$

Das Zwischenergebnis (hier ungünstig y genannt, obwohl unsere y -Koordination auch den Namen haben) ist nach Gauß:

$$y = \begin{pmatrix} 1 \\ 5/3 \\ 2 \\ 0 \end{pmatrix}$$

Jetzt noch $Rx = y$ berechnen, wobei x in dem Fall unseren c_0 bis c_3 entsprechen:

$$\begin{bmatrix} 3 & 2 & 1 & 0 \\ 0 & 1/3 & 2/3 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 5/3 \\ 2 \\ 0 \end{pmatrix}$$

Wenn man das mit Gauß berechnet, kommt in der Tat

$$c_0 = 0; \quad c_1 = 2; \quad c_2 = 1; \quad c_3 = -1$$

wie bei obiger Beispielbox heraus.

Erklärung

Kubische Polynome nennt man übrigens ganzrationale Polynome dritten Grades.

Das schöne bei der Hermite Interpolation ist es, dass man die kubischen Basisfunktionen für jeden (Interpolations-)Bereich derselben Funktion immer wiederverwenden kann und nicht mehr auszurechnen ist. Damit dies funktioniert müssen wir lediglich das neue Intervall (=Interpolationsbereich) auf das Intervall abbilden, in dem wir die Basisfunktionen ausgerechnet haben.

Wir rechnen also die kubischen Basisfunktionen immer im Intervall $[0, 1]$ aus und müssen nur noch beliebige andere Intervalle, die wir ausrechnen wollen dahin transformieren.

Formal ausgedrückt müssen wir das neue (hier allgemeine) Intervall $[x_i, x_{i+1}]$ auf das vorherige $t \in [0; 1]$ abbilden.

Das geht mit folgender Funktion:

$$t_i(x) := \frac{x - x_i}{x_{i+1} - x_i} \in [0; 1] = \frac{x - x_i}{h_i} \in [0; 1] \quad (56)$$

Informell ist h_i die Länge des Intervalls.

Wenn wir die transformierte (also auf das allg. Intervall abgebildete) Funktion ableiten wollen, müssen wir die Kettenregel berücksichtigen.

$$\frac{d}{dx}(f(t_i(x))) = \frac{d}{dt}(f(t_i(x))) \cdot \frac{dt}{dx} = \frac{d}{dt}(f(t_i(x))) \cdot \frac{1}{x_{i+1} - x_i} \quad (57)$$

Dadurch wird das sogenannte »kubische Teilpolynom« $p_i(t(x))$ auch noch folgendermaßen erweitert: Bei den Summanden der Ableitungen kommt der Faktor h_i hinzu, weil dort bei der Ableitung sonst ein $\frac{1}{h_i}$ auftauchen würde, welche die Interpolation verfälschen würde.

Damit haben wir die Formel für eine beliebige Hermite-Funktion:

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h_i \cdot y'_i \cdot H_2(t_i(x)) + h_i \cdot y'_{i+1} \cdot H_3(t_i(x)). \quad (58)$$

Beispiel

Wir wollen das Intervall $[3, 4]$ auf das Intervall $[0, 1]$ transformieren. Wir müssen also die Transformation nach obigem Schema aufstellen:

$$\begin{aligned} t(x) &= \frac{x-3}{4-3} \\ &= \frac{x-3}{1} \end{aligned}$$

Jetzt müssen wir nur schreiben, dass $x \in [3, 4]$ gilt und wir sind fertig, $t(x) \in [0, 1]$ wie wir es wollten.

Vertiefung

Die Formel aus dem letzten Erklärblock ist um auf $[0, 1]$ zu transformieren. Wir können das natürlich für beliebige Intervalle ausbauen. Nützlich wenn wir beispielsweise die kubischen Basisfunktionen gar nicht für das Intervall $[0, 1]$ ausrechnen können (warum auch immer).

Wenn wir ein beliebiges Intervall $[a, b]$ auf ein anderes beliebiges Intervall $[c, d]$ transformieren wollen, bilden wir folgende Transformationfunktion:

$$\begin{aligned} h_{ab} &:= b - a \\ h_{cd} &:= d - c \\ t(x) &:= \frac{x}{h_{ab}/h_{cd}} + \left(c - \frac{a}{h_{ab}/h_{cd}} \right) \\ &= \frac{x \cdot h_{cd}}{h_{ab}} + \left(c - \frac{a \cdot h_{cd}}{h_{ab}} \right) \end{aligned}$$

Beispiel

Wir wollen das Intervall $[2, 4]$ auf $[8, 14]$ transformieren. Wir nutzen obige Formel:

$$\begin{aligned} h_{ab} &= 4 - 2 = 2 \\ h_{cd} &= 14 - 8 = 6 \\ t(x) &= \frac{x \cdot 6}{2} + \left(8 - \frac{2 \cdot 6}{2} \right) \\ &= 3x + 2 \end{aligned}$$

Und fertig ☺.

9.5 Interpolation mit Splines

Erklärung

Die Interpolation mit Splines ist eine stückweise Interpolation. Sprich wir unterteilen den zu interpolierenden Bereich in diesem Fall bei jedem Stützpunkt, der sich nicht am Rand befindet. Jeder Bereich bildet hier ein sogenanntes »Spline«.

Bei der Interpolation mit Splines wollen wir den gleichen Polynomgrad wie bei der Hermite-Interpolation erreichen, nur dass wir keine Ableitungen zur Verfügung haben. Aus diesem Grund verwenden wir als zusätzliche Bedingung, dass die Verknüpfungsstellen zwischen 2 Polynomen sowohl in der ersten als auch in der 2. Ableitung übereinstimmen. Diese Eigenschaft nennt man C2-Stetigkeit.

Dies generiert 2 weitere Bedingungen, fehlen noch zwei, um auf den gewünschten Grad zu gelangen. Wir nehmen deshalb die Ableitungen an den Rändern, um die letzten Gleichungen zu erhalten.

Vorgehen

Mit

$$x_{i+1} - x_i = h := \frac{x_n - x_0}{n}; \quad i = 0, 1, \dots, n-1$$

Löse das System

$$\begin{bmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{bmatrix} \cdot \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_{n-2} \\ y'_{n-1} \end{pmatrix} = \frac{3}{h} \cdot \begin{pmatrix} y_2 - y_0 \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \end{pmatrix} - \begin{pmatrix} y'_0 \\ 0 \\ \vdots \\ 0 \\ y'_n \end{pmatrix} \quad (59)$$

Bestimme mithilfe dieser Gleichung die Ableitungen aller Stützpunkte außer den an den Rändern. Dann hat man alle Informationen, um für jedes Paar benachbarter Stützstellen die entsprechenden Hermite-Funktionen aufzustellen.

Für jedes Spline muss eine Transformationsfunktion aufgestellt werden, um die Intervalle zu normieren (siehe [Kapitel 9.4](#)). Danach kann man die Spline-Funktion definieren, die alle Splines vereinigt.

Aufgrund dieser Tri-diagonalmatrix (nennt man so) ist die Laufzeit von Splines in $\mathcal{O}(n)$.

Beispiel

Beispiel von oben erweitert:

$$f(0) = 0; \quad f(1) = 2; \quad f(2) = 4; \quad f(3) = 8$$

Die Ableitung an den Rändern sei vorgegeben:

$$f'(0) = 2; \quad f'(3) = 4$$

Wir haben also 3 Splines mit einer Länge von eins ($h = 1$).

In die obige Formel einsetzen, um die restlichen zwei Ableitungen zu approximieren:

$$\begin{aligned} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \cdot \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} &= \frac{3}{1} \cdot \begin{bmatrix} 4 - 0 \\ 8 - 2 \end{bmatrix} - \begin{pmatrix} 2 \\ 4 \end{pmatrix} \\ \Rightarrow \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \cdot \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} &= \begin{bmatrix} 12 - 2 \\ 18 - 4 \end{bmatrix} = \begin{pmatrix} 10 \\ 14 \end{pmatrix} \end{aligned}$$

Dies entspricht einem LGS; die Lösung mithilfe von Gauß ist:

$$y'_1 = \frac{26}{15}; \quad y'_2 = \frac{46}{15}$$

Damit lässt sich dann mithilfe der kubischen Basispolynome von Hermite die Spline-Funktion aufstellen, die alle drei Splines vereint. Dafür müssen wir unter anderem noch die t -Funktionen setzen. Beispiel dafür folgt noch.

Wichtig

Ein paar wichtige Dinge zu Splines:

Wenn sich ein Stützpunkt ändert, wirkt sich das immer nur lokal auf einen kleinen Bereich der Kurve aus (halt nur ein bis zwei Splines), der Großteil der Kurve bleibt unverändert.

Außerdem sind Spline Kurven so konstruiert, dass sie möglichst glatt durch die Stützstellen laufen, da die Krümmung der Gesamtfunktion minimiert wird.

Starke Oszillationen wie bei Polynominterpolation hohen Grades (sprich Runge-Effekt) können nicht auftreten!

Beispiel

Wir haben folgende Stützpunkte über eine unbekannte Funktion $f(x)$ gegeben und alle fehlenden Ableitungen nach der eingeführten Formel korrekt ausgerechnet:

i	0	1	2	3
x_i	-2	0	2	4
y_i	12	4	6	3
y'_i	-4	0.5	-1	1

Tabelle 11: Alle Informationen als Tabelle

Da wir vier Stützpunkte (an den Stellen -2 , 0 , 2 und 4) haben, erhalten wir drei Splines. Deren Intervall ist in allen Fällen zwei lang (von -2 bis $0 \dots$).

Gegeben sei außerdem, dass die kubischen Basispolynome der unbekannten Funktion $f(x)$ im Intervall $[0; 1]$ schon korrekt ausgerechnet worden sind.

Wir definieren unsere Spline-Funktion mit allen drei Splines:

$$s(x) = \begin{cases} p_0(t_0(x)); & x \in [-2; 0[\\ p_1(t_1(x)); & x \in [0; 2[\\ p_2(t_2(x)); & x \in [2; 4] \end{cases}$$

Die einzelnen p -Funktionen sind die einzelnen Splines und damit Hermite-Funktionen (die wir noch definieren müssen). Erst einmal müssen wir jetzt aber die Transformationsfunktionen $t_0(x)$, $t_1(x)$ und $t_2(x)$ definieren, die das Intervall des jeweiligen Splines nach $[0; 1]$ transformiert (weil dort die kubischen Basispolynome schon ausgerechnet worden sind).

Fangen wir vorne an:

$$\begin{array}{lll} t_i(x) := \frac{x - x_i}{x_{i+1} - x_i} & & \\ t_0(x) = \frac{x + 2}{2} & x \in [-2; 0] & t_0(x) \in [0; 1] \\ t_1(x) = \frac{x}{2} & x \in [0; 2] & t_1(x) \in [0; 1] \\ t_2(x) = \frac{x - 2}{2} & x \in [2; 4] & t_2(x) \in [0; 1] \end{array}$$

Jetzt müssen wir noch die einzelnen Splines definieren. Wir nutzen obige allgemeine Hermite-Funktion:

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h_i \cdot y'_i \cdot H_2(t_i(x)) + h_i \cdot y'_{i+1} \cdot H_3(t_i(x))$$

Jetzt müssen wir nur noch aus unserer Tabelle die Werte einsetzen:

$$\begin{aligned} p_0(t_0(x)) &= y_0 \cdot H_0(t_0(x)) + y_1 \cdot H_1(t_0(x)) + h_0 \cdot y'_0 \cdot H_2(t_0(x)) + h_0 \cdot y'_1 \cdot H_3(t_0(x)) \\ &= 12 \cdot H_0(t_0(x)) + 4 \cdot H_1(t_0(x)) + 2 \cdot (-4) \cdot H_2(t_0(x)) + 2 \cdot \frac{1}{2} \cdot H_3(t_0(x)) \end{aligned}$$

Der Rest geht analog:

$$\begin{aligned} p_1(t_1(x)) &= 4 \cdot H_0(t_1(x)) + 6 \cdot H_1(t_1(x)) + 2 \cdot \frac{1}{2} \cdot H_2(t_1(x)) + 2 \cdot (-1) \cdot H_3(t_1(x)) \\ p_2(t_2(x)) &= 6 \cdot H_0(t_2(x)) + 3 \cdot H_1(t_2(x)) + 2 \cdot (-1) \cdot H_2(t_2(x)) + 2 \cdot 1 \cdot H_3(t_2(x)) \end{aligned}$$

Wir definieren mit obigen Hermite-Funktionen unsere fertige Spline-Funktion:

$$s(x) = \begin{cases} p_0(t_0(x)); & x \in [-2; 0[; & t_0(x) = (x+2)/2 \\ p_1(t_1(x)); & x \in [0; 2[; & t_1(x) = x/2 \\ p_2(t_2(x)); & x \in [2; 4]; & t_2(x) = (x-2)/2 \end{cases}$$

Erklärung

Ihr fragt euch sicher, warum wir oben als Randbedingung Ableitungen zur Verfügung haben, wenn die Idee doch eigentlich ist, keine zu benötigen.

Die Antwort ist einfach: Dadurch, dass sich eine Änderung nur lokal auswirkt und in der Praxis die Interpolationsbereiche riesig sind, interessiert es uns nicht so sehr, wenn die Ränder der Interpolation ungenau sind.

Also wir brauchen die Randableitungen dann doch nicht mehr, wir setzen sie einfach auf 0. Sie verfälschen zwar dadurch unsere Ergebnisse (gerade an den Rändern), aber nicht viel. Bei einer niedrigen Spline Anzahl wäre das natürlich fataler.

Wie zur Hölle kommt man auf das Gleichungssystem, welches man für Splines verwendet...?

Vertiefung**Spline-Interpolation Herleitung:**

Wir stellen wieder nach dem Hermite-Ansatz unsere Interpolationsfunktionen auf:

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h \cdot y'_i \cdot H_2(t_i(x)) + h \cdot y'_{i+1} \cdot H_3(t_i(x))$$

Dafür benötigen wir wieder obige Transformationsfunktion und die Kettenregel, die schon oben eingebaut ist.

Nun können wir alle y_i bestimmen, indem wir die letzte Bedingung der C2-Stetigkeit nutzen. Wir setzen einfach die zweite Ableitung der Funktionen $p_i(t(x) = 1)$ und $p_{i+1}(t(x) = 0)$ an den Klebestellen gleich.

$$p_i''(t = 1) = p_{i+1}''(t = 0); \quad i = 0, 1, \dots, n-2$$

Durch zweimaliges Ableiten der Basispolynome erhält man mit Kettenregel den Faktor $1/h^2$, also:

$$\begin{aligned} & \frac{1}{h^2} \cdot [y_i \cdot H_0''(1) + y_{i+1} \cdot H_1''(1) + h \cdot y'_i \cdot H_2''(1) + h \cdot y'_{i+1} \cdot H_3''(1)] \\ &= \frac{1}{h^2} \cdot [y_{i+1} \cdot H_0''(0) + y_{i+2} \cdot H_1''(0) + h \cdot y'_{i+1} \cdot H_2''(0) + h \cdot y'_{i+2} \cdot H_3''(0)] \end{aligned}$$

Jetzt sortieren wir y' auf die linke und y auf die rechte Seite:

$$\begin{aligned} & \frac{h}{h^2} \cdot [y'_i \cdot H_2''(1) + y'_{i+1} \cdot (H_3''(1) - H_2''(0)) - y'_{i+2} \cdot H_3''(0)] \\ &= \frac{1}{h^2} \cdot [-y_i \cdot H_0''(1) + y_{i+1} \cdot (-H_1''(1) + H_0''(0)) + y_{i+2} \cdot H_1''(0)] \end{aligned}$$

Wenn wir hier nun unsere kubischen Basisfunktionen H_i'' mit $i = 0, \dots, n-2$ einsetzen, kommen wir zwangsläufig auf:

$$[y'_i + 4y'_{i+1} + y'_{i+2}] = \frac{3}{h} \cdot [y_{i+2} - y_i]$$

Und daraus können wir dann obiges Gleichungssystem ableiten:

$$\begin{bmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{bmatrix} \cdot \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_{n-2} \\ y'_{n-1} \end{pmatrix} = \frac{3}{h} \cdot \begin{pmatrix} y_2 - y_0 \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \end{pmatrix} - \begin{pmatrix} y'_0 \\ 0 \\ \vdots \\ 0 \\ y'_n \end{pmatrix}$$

10 Komplexe Zahlen

Erklärung

Komplexe Zahlen $z \in \mathbb{C}$ bestehen aus einem **Realteil** und einem **Imaginärteil**. Hier sind kurz und knapp alle wichtigen Fakten in Stichpunkten zusammengefasst.

- $z = x + iy$
- $x = \operatorname{Re}(z); \quad y = \operatorname{Im}(z)$
- $i = \sqrt{-1}$
- $\bar{z} = x - iy$ (komplex konjugiertes)
- $\overline{z \cdot w} = \bar{z} \cdot \bar{w}$
- $|z| = \sqrt{\operatorname{Re}(z)^2 + \operatorname{Im}(z)^2} = \sqrt{z \cdot \bar{z}}$
- Euler-Formel: $e^{i \cdot m} = \cos(m) + i \cdot \sin(m)$
- $e^z = e^{x+iy} = e^x (\cos(y) + i \cdot \sin(y))$
- $z_1 + z_2 = (x_1 + x_2) + i \cdot (y_1 + y_2)$
- $z_1 - z_2 = (x_1 - x_2) + i \cdot (y_1 - y_2)$
- $z_1 \cdot z_2 = (x_1 x_2 - y_1 y_2) + i \cdot (x_1 y_2 + x_2 y_1)$

Beispiel


$$a = 3 + 4i; \quad b = 5 + 3i$$

$$\bar{a} = 3 - 4i$$

$$\operatorname{Re}(a) = 3$$

$$\operatorname{Im}(a) = 4 \quad (\text{nicht } 4i \text{ !!!})$$

$$\begin{aligned} |a| &= \sqrt{3^2 + 4^2} \\ &= 5 \end{aligned}$$


$$\begin{aligned}e^a &= e^{3+4i} \\ &= e^3 \cdot (\cos(4) + i \cdot \sin(4))\end{aligned}$$

$$\begin{aligned}c &= a + b \\ &= 8 + 7i\end{aligned}$$

$$\begin{aligned}d &= a \cdot b \\ &= 3 \cdot 5 + 3 \cdot 3i + 4i \cdot 5 + 4i \cdot 3i \\ &= (15 - 12) + (9 + 20)i \\ &= 3 + 29i\end{aligned}$$

11 Frequenzanalyse

Erklärung

Trigonometrische Funktionen (aka solche, die Operationen wie $\sin()$ verwenden), besitzen immer eine Amplitude und eine Frequenz. Einige Probleme in der Praxis (wie z.B. Soundwellen) basieren auf Signalen, die man in ihre einzelnen Frequenzen auf trennen möchte. Solche Signale nimmt man als Summen trigonometrischer Funktionen an und lassen sich oft anschaulich durch ihr »Spektrum« darstellen.

Beispiel

Gegeben sei folgende »Grundfrequenz«:

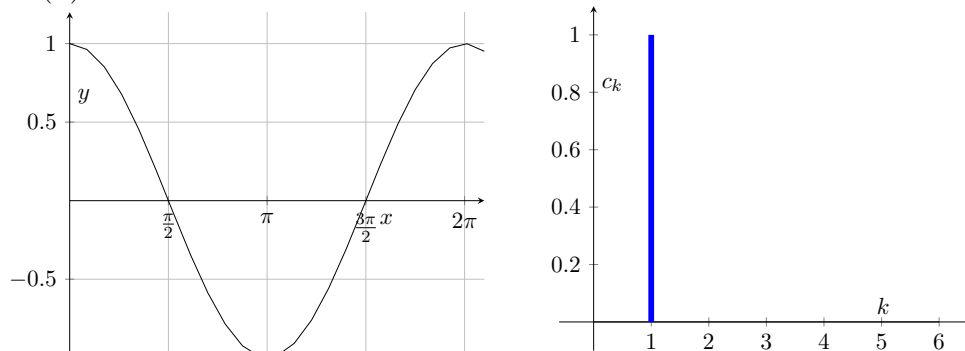
$$f(x) = \sum_{k=0}^{\infty} c_k \cdot \cos(kx)$$

Damit lassen sich alle Signale darstellen, die nur aus Kosinus Überlagerungen bestehen. In dem Fall würde ein Signal, welches nur aus der normalen Kosinusfunktion besteht, ausschließlich das Spektrum $k = 1$ aufweisen. Eine gerade Linie als Signal hätte das Spektrum: $k = 0$.

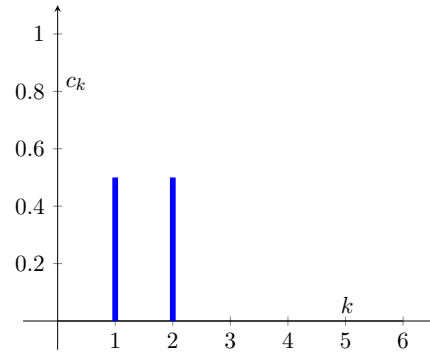
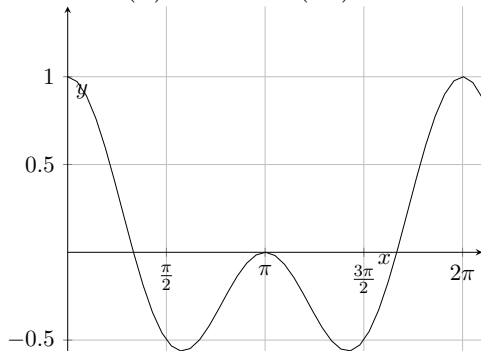
Also sind die unterschiedlichen k die unterschiedlichen Frequenzen, c_k beschreibt den Gewichtungskoeffizienten. Bei der Frequenzanalyse ist das äquivalent zu der Amplitude der jeweiligen Frequenz. Also der normale Kosinus hat nur die Frequenz $k = 1$ mit einer Amplitude von $c_k = 1$, da der Kosinus zwischen Eins und -1 schwingt.

Im Folgenden ein paar Beispiele von Schwingungen und ihren Frequenzspektren:

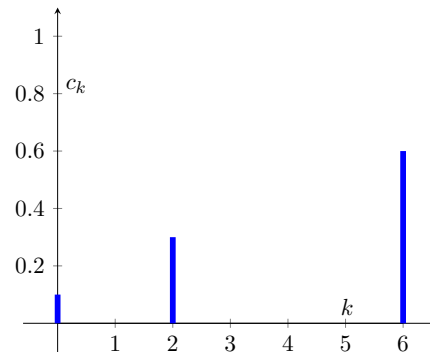
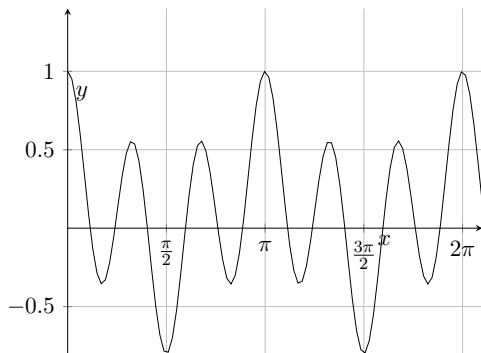
$\cos(x)$:



$$0.5 \cdot \cos(x) + 0.5 \cdot \cos(2x):$$



$$0.1 + 0.3 \cdot \cos(2x) + 0.6 \cdot \cos(6x):$$



11.1 Diskrete Fourier-Transformation

Erklärung

Die Idee hinter der Fourier-Transformation ist es, ein periodisches, diskretes Signal als Frequenzspektrum auszudrücken. Dies wird z.B. häufig im Zusammenhang mit Sound-Effekten ein »Muss«.

Die Eingabe einer Fourier-Transformation ist ein Vektor v , der aus Stützwerten (wie bei Interpolation) besteht. Diese Stützwerte sind immer äquidistant und im Intervall $[0, 2\pi)$. Warum dieses Intervall? Naja, alle trigonometrischen Funktionen sind 2π periodisch. Aus demselben Grund ist auch die obere Intervallgrenze 2π nicht mehr im Intervall, da es immer derselbe Wert sein muss wie bei $x = 0$. Außerdem ähnlich zu Interpolation ist jetzt das Prinzip, das wir anwenden. Im Grunde machen wir einfache Interpolation mithilfe von Basisfunktionen (siehe [Kapitel 9.1.1](#)), nur mit trigonometrischen Basisfunktionen im komplexen Bereich, um 3D Signale abdecken zu können.

Wenn man es implementiert, erkennt man, dass es sich lediglich um eine Vektor-Matrix Multiplikation handelt, sprich die Laufzeit entspricht $\mathcal{O}(n^2)$.

Vorgehen

Die Eingabe für die Diskrete Fourier Transformation (DFT) ist ein Vektor v . Mit

$$\omega = \exp\left(i \cdot \frac{2\pi}{n}\right) \quad (60)$$

und mit n als die Größe des Eingabevektors v ergibt sich die Formel:

$$\text{DFT}(v) := c_k = \frac{1}{n} \cdot \sum_{j=0}^{n-1} v_j \cdot \bar{\omega}^{jk} \quad (61)$$

Die Formel für die Inverse Diskrete Fourier Transformation (IDFT):

$$\text{IDFT}(c) := v_j = \sum_{k=0}^{n-1} c_k \cdot \omega^{jk} \quad (62)$$

In Matrix-Vektor Schreibweise schön darstellbar:

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} = c = M_{\text{DFT},n} \cdot v = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \bar{\omega}^1 & \bar{\omega}^2 & \dots & \bar{\omega}^{(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \bar{\omega}^{n-1} & \bar{\omega}^{2 \cdot (n-1)} & \dots & \bar{\omega}^{(n-1)(n-1)} \end{bmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \quad (63)$$

Das Inverse:

$$\begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} = v = M_{\text{IDFT},n} \cdot c = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^1 & \omega^2 & \dots & \omega^{(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2 \cdot (n-1)} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} \quad (64)$$

Erklärung

Außerdem können wir aus unserem Omega (also die Grundfrequenz) noch ein paar Infos ableiten:

$$\begin{aligned} \omega &= \exp\left(i \cdot \frac{2\pi}{n}\right) \\ &= \cos\left(\frac{2\pi}{n}\right) + i \cdot \sin\left(\frac{2\pi}{n}\right) \\ \bar{\omega} &= \cos\left(\frac{2\pi}{n}\right) - i \cdot \sin\left(\frac{2\pi}{n}\right) = \omega^{-1} \end{aligned}$$

Da es also eigentlich nur einer Kosinus Kurve (im Realteil) und einer Sinuskurve (im Imaginärteil) entspricht, können wir eine Symmetrie und Periodik ableiten:

$$\omega^{k \cdot (n-j)} = \omega^{-kj} \quad (\text{Symmetrie}) \quad (65)$$

$$w^{k \cdot j} = w^{k \cdot (n+j)} = w^{(k+n) \cdot j} \quad (\text{Periodik}) \quad (66)$$

Daraus können wir uns zum Beispiel für $n = 3$ herleiten:

$$\begin{aligned} \bar{\omega}^4 &= \omega^2 = \bar{\omega} \\ \omega^4 &= \bar{\omega}^2 = \omega \end{aligned}$$

Beispiel

$$v = (0, -1, 1)^T$$

$$\begin{aligned} \text{DFT}(v) &= \frac{1}{3} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & \bar{\omega}^1 & \bar{\omega}^2 \\ 1 & \bar{\omega}^2 & \bar{\omega}^4 \end{bmatrix} \cdot \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} \\ &= \frac{1}{3} \cdot \begin{pmatrix} 0 \\ -\bar{\omega}^1 + \bar{\omega}^2 \\ -\bar{\omega}^2 + \bar{\omega}^4 \end{pmatrix} = \frac{1}{3} \cdot \begin{pmatrix} 0 \\ \omega - \bar{\omega} \\ \bar{\omega} - \omega \end{pmatrix} \end{aligned}$$

Jetzt wenden wir die Euler-Formel an (siehe [Kapitel 10](#)):

$$\text{DFT}(v) = \frac{1}{3} \cdot \begin{pmatrix} 0 \\ (\cos(\frac{2\pi}{3}) + i \cdot \sin(\frac{2\pi}{3})) - (\cos(\frac{2\pi}{3}) - i \cdot \sin(\frac{2\pi}{3})) \\ (\cos(\frac{2\pi}{3}) - i \cdot \sin(\frac{2\pi}{3})) - (\cos(\frac{2\pi}{3}) + i \cdot \sin(\frac{2\pi}{3})) \end{pmatrix}$$

Mit

$$\cos\left(\frac{2\pi}{3}\right) = -0.5; \quad \sin\left(\frac{2\pi}{3}\right) = \frac{\sqrt{3}}{2}$$

$$\begin{aligned} \text{DFT}(v) &= \begin{pmatrix} 0 \\ (-0.5 + i \cdot \frac{\sqrt{3}}{6}) - (-0.5 - i \cdot \frac{\sqrt{3}}{6}) \\ (-0.5 - i \cdot \frac{\sqrt{3}}{6}) - (-0.5 + i \cdot \frac{\sqrt{3}}{6}) \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ (0 + i \cdot \frac{2\sqrt{3}}{6}) \\ (0 - i \cdot \frac{2\sqrt{3}}{6}) \end{pmatrix} \end{aligned}$$

11.2 Schnelle (Inverse) Fourier-Transformation

Erklärung

Die Fourier-Transformation lässt sich schneller implementieren. Die Idee hinter dieser schnellen Variante ist es, mithilfe des sogenannten »Butterfly-Operators« die Menge an redundanten Operationen durch Rekursion zu verkleinern. Dafür sortiert man in der sogenannten »Sortier-Phase« den Eingabevektor nach den Einträgen mit gerade und ungeraden Indizes rekursiv. Danach wendet man oben genannten Operator in der »Kombinationsphase« an und geht dabei die Rekursion Ebenen wieder nach oben. Es wird sogar deutlich schneller, die neue Laufzeit von IFFT (= Inverse Fast Fourier Transformation) liegt in $\mathcal{O}(n \cdot \log(n))$.

Wir nutzen dafür die zwei Eigenschaften von Omega:

$$\begin{aligned}\omega^{k \cdot (n-j)} &= \omega^{-kj} && \text{(Symmetrie)} \\ w^{k \cdot j} &= w^{k \cdot (n+j)} = w^{(k+n) \cdot j} && \text{(Periodik)}\end{aligned}$$

Vorgehen

Butterfly-Operator

Zwei Einträge eines Vektors a_j und b_j mit demselben Index j werden immer zusammen mit dem Butterfly-Operator verbunden. Mit BFO als Butterfly-Operator ist definiert:

$$\text{BFO}(a_j, b_j) := (a_j + \omega^j \cdot b_j, a_j - \omega^j \cdot b_j) \quad (67)$$

Das Ergebnis sind die Einträge der neuen Vektoren.

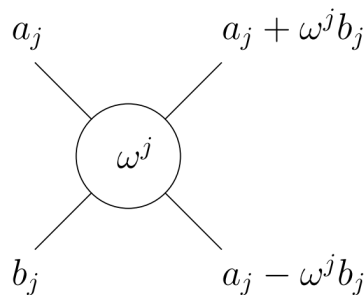


Abbildung 12: Der BFO schöner dargestellt. Links von dem Kreis ist die Eingabe (wobei streng genommen das ω^j auch eine Eingabe ist und rechts jeweils die Ausgaben, die jeweils die Vektor-Einträge der Eingabe überschreiben.

Beispiel

Gegeben seien zwei Arrays mit jeweils einem Eintrag:

$$a_0 = 3 \quad b_0 = 5$$

Nun wenden wir den Butterfly-Operator an. Wir merken uns, dass unser Index $j = 0$ ist.

Unser Ergebnis ist dann ein zweidimensionaler Vektor z mit:

$$\begin{aligned} z &= \begin{pmatrix} a_0 + w^0 \cdot b_0 \\ a_0 - w^0 \cdot b_0 \end{pmatrix} \\ &= \begin{pmatrix} 3 + 1 \cdot 5 \\ 3 - 1 \cdot 5 \end{pmatrix} \\ &= \begin{pmatrix} 8 \\ -2 \end{pmatrix} \end{aligned}$$

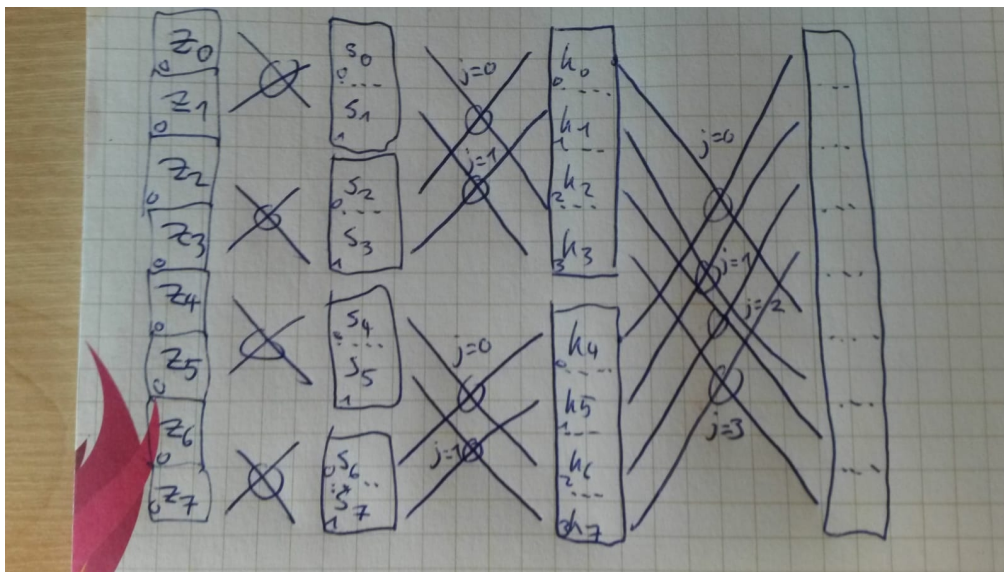


Abbildung 13: Typische Skizze einer Fast Fourier Transformation mit 8 Einträgen.

Erklärung

Obiges Bild zeigt die Kombinationsphase, in der die einzelnen Buckets mithilfe des BFO miteinander kombiniert werden. Links unten von jedem Eintrag steht dabei der Index, den der Eintrag in seinem Bucket hat. Das sind nämlich genau die j Werte

beim Kombinieren, wie die Einträge neben den BFO Strichen zeigt (also dieses Kreuz mit dem Kreis entspricht dem BFO).

Dass die Einträge in jedem Schritt anders benannt wurden, spielt nicht wirklich eine Rolle und soll nur zeigen, dass sich die Werte ändern, nicht aber die Positionen.

Wichtig: ω ist weiterhin $\exp\left(\frac{2\pi i}{n}\right)$, aber n ist bei jedem Kombinationsschritt anders! So ist $n = 2$ bei dem ersten Kombinationsschritt, $n = 4$ bei dem danach und so weiter...

Auf dem Musterlösungsblatt kann man auch ein rasches Codebeispiel für FFT von der Übungsleitung finden.

12 Quadratur

Erklärung

Nachdem wir herausgefunden haben, wie wir eine Funktion mithilfe von Interpolation annähern können, drängt sich sofort die Frage auf, wie das mit der Integration einer Funktion aussieht.

Es geht nicht nur, sondern auch noch recht simpel. Wir interpolieren die Funktion mithilfe von schon besprochenen Methoden und integrieren dann einfach analytisch. So integrieren wir eine Annäherung der Funktion und erhalten damit auch eine Approximation des Integrals. Dies ist die Kategorie der Quadratur-Probleme.

Vorgehen

Allgemeines Quadraturproblem:

$$\int_a^b f(x) \, dx \approx \sum_{i=1}^n f(x_i) \cdot w_i$$

Die w_i nennt man Gewichtungen, also wir gewichten $f(x_i)$ Werte, die ja den Stützpunkten entsprechen. Aus der Kombination an Stützpunkten und deren Gewichtungen erhalten wir eine approximative Lösung.

Sehr einfach mit Lagrange (siehe [Kapitel 9.1.2](#)):

$$\begin{aligned} f(x) &\approx p(x) = \sum_{i=1}^n f(x_i) \cdot L_i(x) \\ \longrightarrow \int_a^b f(x) \, dx &\approx \int_a^b p(x) \, dx = \int_a^b \sum_{i=1}^n f(x_i) \cdot L_i(x) \, dx \end{aligned}$$

Für 2 Stützpunkte erhalten wir die Formel:

$$Q_T(f) = H \cdot \frac{1}{2} \cdot (f(a) + f(b)) \quad (68)$$

Wir nennen diese Formel die **Trapezregel**.

Für 3 Stützpunkte erhalten wir die Formel:

$$Q_S(f) = H \cdot \frac{1}{6} \cdot \left(f(a) + 4 \cdot f\left(\frac{a+b}{2}\right) + f(b) \right) \quad (69)$$

Diese Formel nennt man die **Simpsonregel** (oder nach Johannes Kepler gibt es das Synonym **Keplersche Fassregel**) und ist genauer als die Trapezregel.

Die Variable H entspricht hierbei der Länge des zu integrierenden Bereichs, also $H = b - a$.

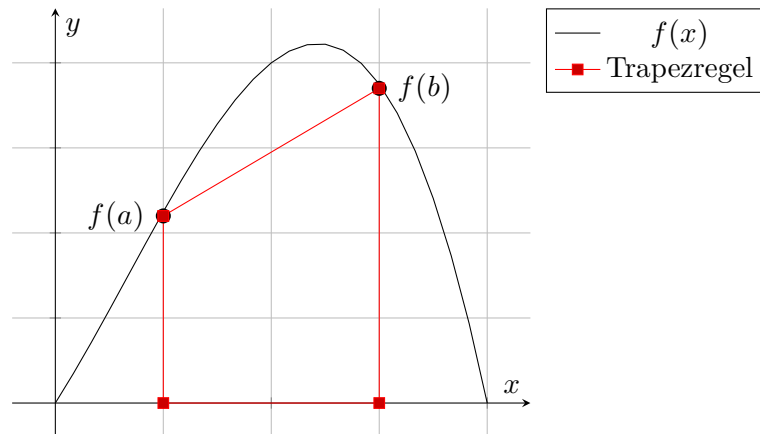


Abbildung 14: Ein zufälliges Beispiel einer Trapezregel. a und b jeweils wie in der Formel rechnet die Trapezregel die umrandete Fläche aus.

Beispiel

Wir möchten $\int_1^2 f(x) \, dx$ einer Funktion $f(x)$ ausrechnen, bei denen wir nur die Stützpunkte $s_0 : f(1) = 2$ und $s_1 : f(2) = 4$ kennen.

Nach der Trapezregel ergibt sich also:

$$Q_T(f) = 1 \cdot \frac{1}{2} \cdot (2 + 4) = 3$$

Beispiel

Wir haben die Funktion $f(x) = \sin(x)$. Wir sollen mit den Stellen $x_0 = 0$ und $x_1 = \pi$ und der Trapezregel die Fläche in diesen Bereich annähern.

$$\begin{aligned} Q_T(f) &= \pi \cdot 0.5 \cdot (f(0) + f(\pi)) \\ &= \pi \cdot 0.5 \cdot (\sin(0) + \sin(\pi)) \\ &= \pi \cdot 0.5 \cdot (0) \\ &= 0 \end{aligned}$$

Die Trapezregel schätzt, dass die Fläche 0 ist. Wir wissen aber, dass der Sinus zwischen 0 und π eine Fläche deutlich größer 0 hat.

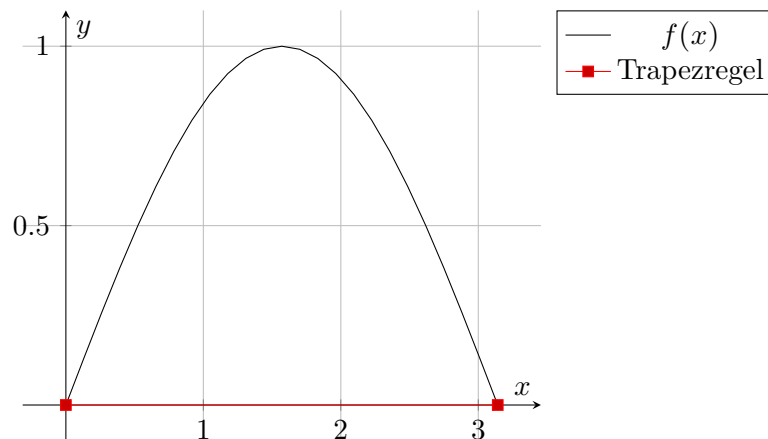


Abbildung 15: Das Problem grafisch. Die Fläche unter dieser Sinuskurve ist definitiv nicht 0!

Hieran kann man sehen, dass es Fälle gibt, wo die Abschätzung schlecht bis sehr schlecht sein kann.

Wichtig

Wir können mit einem Interpolationspolynom vom Grad $2 \cdot n$ ein Polynom vom Grad $2 \cdot n + 1$ noch exakt integrieren. (ein gerader Grad integriert den nächst höheren ungeraden Grad noch exakt)

Beispiel

Eine Konstante (Grad 0) kann eine Gerade (Grad 1) exakt integrieren
Aber eine Gerade (Grad 1) kann keine Parabel (Grad 2) exakt integrieren.

Erklärung

Warum erstellt man nicht eine »Regel« mit sehr viel mehr Stützpunkten?

Das Problem hierbei ist die Kondition. Die entspricht nämlich der Summe aller Beträge der Gewichtungen ω .

$$\sum_i |\omega_i| \tag{70}$$

In aller Regel sind die Gewichtungen positiv und ergeben aufaddiert immer 1. Bei Trapezregel ist es $0.5 + 0.5 = 1$, bei Simpsonregel $1/6 \cdot (1 + 4 + 1) = 1$ und so weiter. Bei deutlich mehr Stützpunkten pro »Regel« (etwa ab 8) werden manche Gewichtungen negativ und somit die Summe der Beträge nicht mehr eins und damit schlechter.

Wir probieren das Problem der Ungenauigkeit auf andere Weise zu verringern. Das behandeln wir im nächsten Kapitel.

12.1 Klassische Quadratur

Erklärung

Komplizierte Funktionen liefern meist schlechte Ergebnisse. Die Idee diese zu lösen ist denkbar einfach: Wir wenden die Quadraturregeln stückweise an und unterteilen unser Problem in viele Teilprobleme.

Vorgehen

Beschreibt H den zu integrierenden Bereich (also zwischen a und b), h die Länge der einzelnen Stücke, in der wir das Problem aufteilen und N die Anzahl jener.

Formell:

$$H = b - a; \quad h = \frac{b - a}{N}$$

$$f_i = f(x_i); \quad x_i = a + i \cdot h$$

$$Q_T(f) = H \cdot \frac{f(a) + f(b)}{2}$$

$$\rightarrow Q_{TS}(f; h) = h \cdot \left(\frac{f_0}{2} + f_1 + \dots + f_{N-1} + \frac{f_N}{2} \right) \quad (71)$$

Q_{TS} entspricht dann der **Trapezsumme**.

$$Q_S(f) = H \cdot \frac{f(a) + 4 \cdot f\left(\frac{a+b}{2}\right) + f(b)}{6}$$

$$\rightarrow Q_{SS}(f; h) = \frac{h}{3} \cdot (f_0 + 4f_1 + 2f_2 + 4f_3 + \dots + 2f_{N-2} + 4f_{N-1} + f_N) \quad (72)$$

Q_{SS} nennt man die **Simpsonsumme**. Benannt ist dies nach Thomas Simpson.

Beispiel

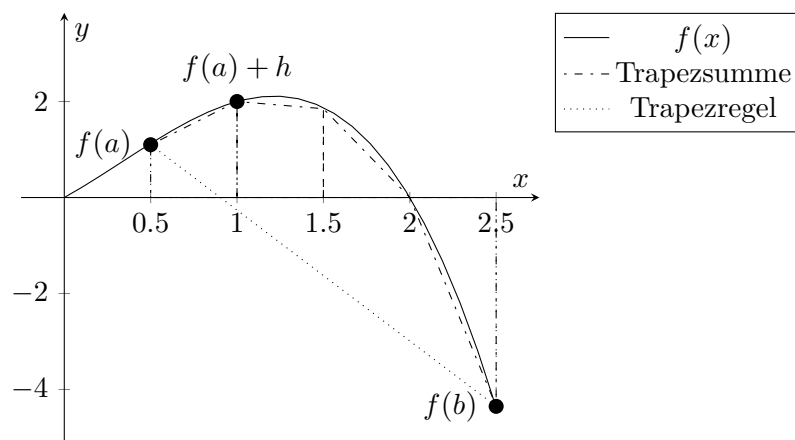


Abbildung 16: Beispiel einer Trapezsumme, bei der $N = 4$ (in dieser Grafik die gestrichelte Linie) Trapeze konstruiert worden sind, um die Fläche besser anzunähern. Im Vergleich zeigt die gepunktete Linie die Annäherung durch die Trapezregel, also einem Trapez.

Beispiel

Und wenn wir uns auch noch einmal das Problem vorher mit Sinus anschauen (siehe [Figur 15](#)) sehen wir, dass mehr Trapeze zu einer höheren Genauigkeit führt.

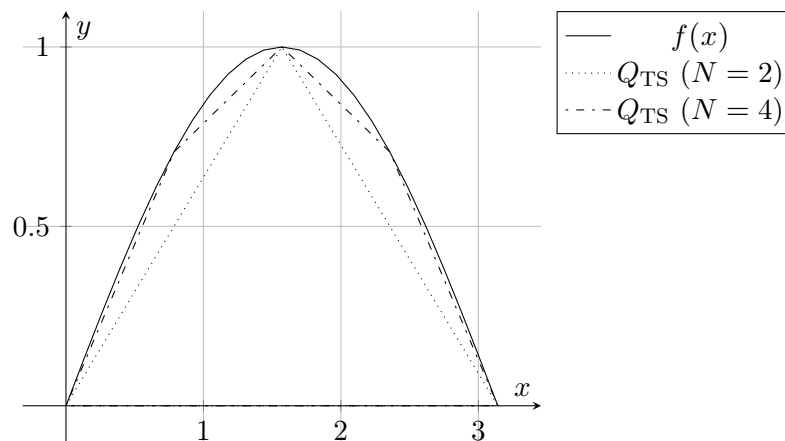


Abbildung 17: Die Approximation ist definitiv besser geworden.

12.2 Restglied

Erklärung

Unter dem sogenannten »Restglied« z.B. einer Trapezsumme beschreibt man die Fehlerabschätzung. Im Grunde also $R_{\text{TS}} \approx Q_{\text{TS}} - I(f)$. Diesen Fehler schätzt man nach folgender Formel ab:

Restglied R_{TS} :

$$R_{\text{TS}}(f; h) = h^2 \cdot H \cdot \frac{f^{(2)}(\xi)}{12} \quad (73)$$

Das ganze gibt es dann noch für die Simpsonsumme.

$$R_{\text{SS}}(f; h) = h^4 \cdot H \cdot \frac{f^{(4)}(\xi)}{180} \quad (74)$$

Das ξ kommt aus dem Mittelwertsatz. Man nimmt hierfür, wenn nötig, meist den Wert aus dem Intervall, dass die Fehlerabschätzung den maximalen Fehler abschätzt und damit den »worst-case«.

Es ist ziemlich analog zur Fehlerabschätzung einer Polynom-Interpolation, siehe [Kapitel 9.1.5](#).

12.3 Quadratur nach Romberg

Erklärung

Quadratur nach Romberg kombiniert zwei Trapezsummen unterschiedlicher Schrittweiten, um das exakte Ergebnis besser zu approximieren. Wir können das iterativ mit jeweils zwei und insgesamt mehr als zwei Trapezsummen machen, um eine höhere Genauigkeit zu erhalten.

Wir konstruieren uns dafür Trapezsummen mit immer kleinerem h bzw. größerem N , die wir auch inkrementell bestimmen können. Es gilt nämlich:

$$Q_{\text{TS}}(f; h) = \frac{Q_{\text{TS}}(f; 2h)}{2} + h \cdot (f_1 + f_3 + \cdots + f_{N-3} + f_{N-1}) \quad (75)$$

Die Romberg-Quadratur ist im Grunde eine **Extrapolation**. Mithilfe der Kombination von Trapezsummen schätzt die Romberg-Quadratur den Wert an der Stelle $h = 0$ ab, was im Prinzip dem analytischen Ergebnis entsprechen sollte (schrittweite von 0 wäre ja theoretisch die perfekte Approximation).

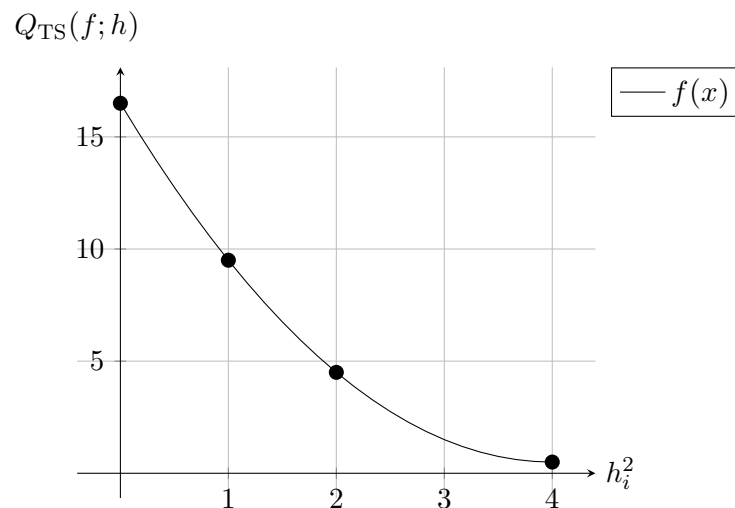


Abbildung 18: Die rechten drei Punkte sind hier die Werte von drei Trapezsummen. Durch diese kann der Punkt an der Stelle $h^2 = 0$ abgeschätzt werden. Das ist dann das Ergebnis der Romberg-Quadratur.

Vorgehen

Zum Kombinieren können wir ähnlich dem Newton und Aitken-Neville Verfahren eine Tabelle schreiben, die nach **rechts unten** determiniert.

h_k	k	Q_{k0}	Q_{k1}	\dots	Q_{kn}
$\frac{b-a}{1}$	0	$Q_{\text{TS}}(f; \frac{b-a}{1}) = Q_{00}$			
$\frac{b-a}{2}$	1	$Q_{\text{TS}}(f; \frac{b-a}{2}) = Q_{10}$	Q_{11}		
$\frac{b-a}{4}$	2	$Q_{\text{TS}}(f; \frac{b-a}{4}) = Q_{20}$	Q_{21}	Q_{22}	
\vdots		\vdots			\ddots

Tabelle 12: Tabelle zum ausrechnen

Mit folgender Gleichung kombinieren wir zwei beliebige Trapezsummen $Q_{\text{TS}}(f, h_1)$ und $Q_{\text{TS}}(f, h_2)$:

$$Q_{\text{TS}}(f, h_{\text{neu}}) = Q_{\text{TS}}(f, h_2) + \frac{Q_{\text{TS}}(f, h_2) - Q_{\text{TS}}(f, h_1)}{\left(\frac{h_1^2}{h_2^2}\right) - 1} \quad (76)$$

Informell und direkt auf das Dreiecksschema angepasst:

$$Q_{\text{neu}} = Q_{\text{links}} + \frac{Q_{\text{links}} - Q_{\text{linksOben}}}{\left(\frac{h_{\text{ganzLinksOben}}}{h_{\text{links}}}\right)^2 - 1} \quad (77)$$

Von der Notation her wie bei Newton-Verfahren (siehe [Kapitel 9.1.3](#)), nur halt jetzt mit »links« statt »rechts« und anstelle von »unten« ist es halt darüber, also »oben«.

Beispiel

Wir möchten folgende Berechnung annähern:

$$\int_1^3 f(x) \, dx$$

Folgende Stützpunkte seien bekannt:

x	1	1.5	2	2.5	3
$f(x)$	-3	-1.75	0	2.25	5

Damit können wir Romberg-Quadratur bis zu drei Trapezsummen aufstellen:

h_k	k	Q_{k0}	Q_{k1}	Q_{k2}
$\frac{b-a}{1}$	0	$Q_{\text{TS}}(f; \frac{b-a}{1}) = Q_{00}$		
$\frac{b-a}{2}$	1	$Q_{\text{TS}}(f; \frac{b-a}{2}) = Q_{10}$	Q_{11}	
$\frac{b-a}{4}$	2	$Q_{\text{TS}}(f; \frac{b-a}{4}) = Q_{20}$	Q_{21}	Q_{22}

$$Q_{\text{TS}}(f; h) = h \cdot \left(\frac{f_0}{2} + f_1 + \dots + f_{N-1} + \frac{f_N}{2} \right)$$

$$Q_{00} = Q_{\text{TS}}(f, \frac{b-a}{1}) = Q_{\text{T}}(f) = 2 \cdot \frac{f(1) + f(3)}{2} = 2 \cdot \frac{-3 + 5}{2} = 2$$

$$Q_{10} = Q_{\text{TS}}(f, \frac{b-a}{2}) = Q_{\text{TS}}(f, 1) = 1 \cdot \left(\frac{-3}{2} + 0 + \frac{5}{2} \right) = 1$$

$$Q_{20} = Q_{\text{TS}}(f, \frac{b-a}{4}) = Q_{\text{TS}}(f, 0.5) = 0.5 \cdot \left(\frac{-3}{2} - 1.75 + 0 + 2.25 + \frac{5}{2} \right) = 0.75$$

Jetzt geht es an das Kombinieren mit der Formel:

$$Q_{\text{neu}} = Q_{\text{links}} + \frac{Q_{\text{links}} - Q_{\text{linksOben}}}{\left(\frac{h_{\text{ganzLinksOben}}}{h_{\text{links}}} \right)^2 - 1}$$

$$Q_{11} = 1 + \frac{1 - 2}{\left(\frac{2}{1} \right)^2 - 1} = \frac{2}{3}$$

$$Q_{21} = 0.75 + \frac{0.75 - 1}{\left(\frac{1}{0.5} \right)^2 - 1} = \frac{8}{12}$$

h_k	k	Q_{k0}	Q_{k1}	Q_{k2}
2	0	2		
1	1	1	2/3	
0.5	2	0.75	8/12	Q_{22}

Jetzt noch das Ergebnis ausrechnen:

$$Q_{22} = \frac{8}{12} + \frac{\frac{8}{12} - \frac{2}{3}}{\left(\frac{2}{0.5}\right)^2 - 1} = \frac{8}{12} + \frac{\frac{8}{12} - \frac{8}{12}}{16 - 1} = \frac{8}{12} = \frac{2}{3}$$

In diesem Fall ist das Ergebnis schon exakt, da die unbekannte Funktion f in Wahrheit $f(x) = x^2 - 4$ ist und die Fläche von 1 bis 3 analytisch exakt $\frac{2}{3}$ entspricht.

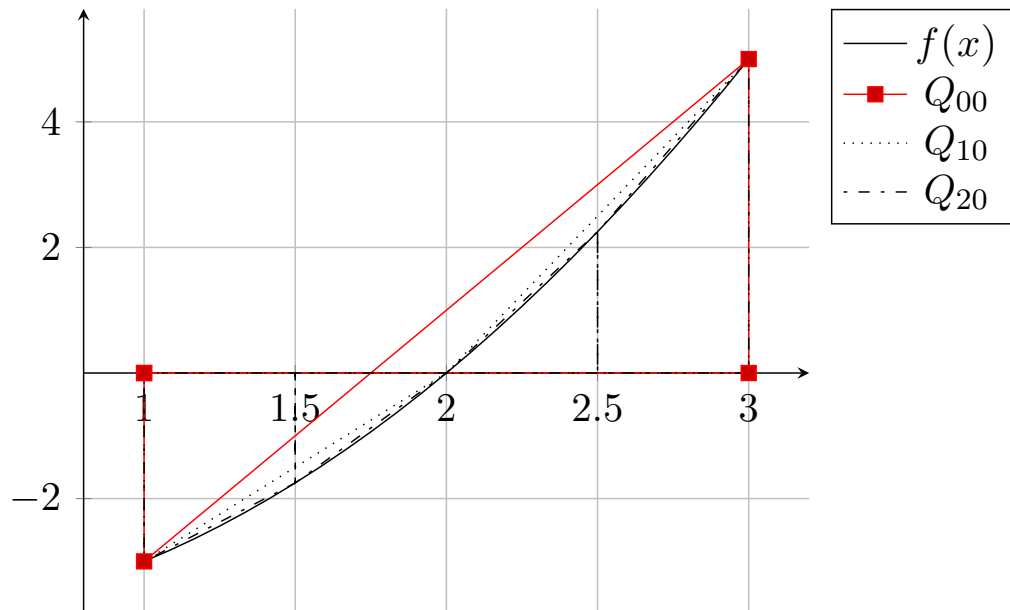


Abbildung 19: Die verschiedenen Trapezsummen, die wir in dem Beispiel genutzt haben sowie die exakte Funktion.

12.4 Gauß Quadratur

Erklärung

Bisher haben wir bei der Interpolation mit den verschiedenen Methoden hauptsächlich die sogenannte Gewichtung einzelner Stützstellen ausgerechnet. Z.B. hatte die Trapezregel (siehe [Kapitel 12](#)) eine Gewichtung von 0.5 bei beiden Stützstellen.

Die Gauß Quadratur gibt jetzt nicht nur die Gewichtung der einzelnen Stützstellen, sondern auch die Position der Stützstellen selbst vor, um die Anzahl der Bedingungen zu verdoppeln. Der erreichbare Polynomgrad liegt also bei $(2 \cdot n - 1)$ bei n Stützstellen, was ziemlich hoch ist.

Eine Einschränkung ist, dass das Integrationsgebiet immer von -1 bis 1 geht. Andere Intervalle müssen wir also ähnlich wie bei Hermite-Interpolation (siehe [Kapitel 9.4](#)) erst einmal strecken und verschieben bzw. transformieren, also ist diese Einschränkung eigentlich kein Thema.

Außerdem erhalten wir bei der Gauß-Quadratur nur positive Gewichte w_i , wodurch die Quadraturformel immer stabil bleibt, die Kondition immer gleich $b - a$ ist (siehe [Kapitel 12](#)). Um eine Gauß-Quadratur Formel herzuleiten, wird die »Methode der unbestimmten Koeffizienten« verwendet.

Vorgehen

Methode der unbestimmten Koeffizienten:

Wir können folgende Gleichung aufstellen, die im Grunde unser allgemeines Quadraturproblem zusammenfasst. Nämlich die Summe von Funktionswerten mal Gewichtungen soll gleich dem Integralwert der Funktion entsprechen.

$$\sum_{i=0}^n f(x_i) \cdot w_i \stackrel{!}{=} \int_a^b f(x) \, dx \quad (78)$$

Diesen Term kann man nun für unterschiedliche Grade aufschreiben, wo diese gelten sollen (bei n Stützstellen darf ja nach Gauß-Quadratur der bis zum $(2 \cdot n - 1)$ 'ten Grad erreicht werden). Danach versuchen wir, dass alle aufgestellten Gleichungen erfüllt werden (ähnlich zu einem LGS). Damit kann man dann auf alle x_i und alle w_i schließen und damit die Regel der entsprechenden Gauß-Quadratur.

Die Funktion $f(x)$ entspricht natürlich den Standard Polynom Grad Funktionen, also für Grad 0 wäre $f(x) = 1$, für Grad 1 ist es $f(x) = x$ usw.

Beispiel

Es soll eine unbekannte Funktion mithilfe von einer Stützstelle x_0 und der Gauß-Quadratur bestimmt werden. Mit einer Stützstelle, also $n = 1$, müssen wir die Grade von 0 bis $(2 \cdot n) - 1 = 1$ aufstellen.

Alle Informationen mithilfe von der obigen Formel aufschreiben:

Grad 0 mit $f(x) = 1$:

$$f(x_0) \cdot w_0 = w_0 \stackrel{!}{=} \int_{-1}^1 f(x) \, dx = 2$$

Grad 1 mit $f(x) = x$:

$$f(x_0) \cdot w_0 = w_0 x_0 \stackrel{!}{=} \int_{-1}^1 f(x) \, dx = 0$$

Daraus können wir also folgern:

$$\begin{aligned} w_0 &= 2 \\ w_0 x_0 &= 0 \\ \implies x_0 &= 0 \end{aligned}$$

Das ist natürlich ein etwas bescheuertes Beispiel, da man mit einer Stützstelle niemals Gauß-Quadratur machen würde. Trotzdem interessant, dass man im Falle einer Stützstelle im besten Falle die in der Mitte des Bereiches nehmen sollte.

Jetzt gibt es noch ein komplexeres Beispiel.

Beispiel

Gauß-Quadratur mit 3 Stützstellen an den Stellen $x_0 = -1$; $x_1 = 0$; $x_2 = 1$. Alle Informationen mithilfe von der obigen Formel aufschreiben:

Grad 0 mit $f(x) = 1$:

$$f(x_0) \cdot w_0 + f(x_1) \cdot w_1 + f(x_2) \cdot w_2 = w_0 + w_1 + w_2 \stackrel{!}{=} \int_{-1}^1 f(x) \, dx = 2$$

Grad 1 mit $f(x) = x$:

$$f(x_0) \cdot w_0 + f(x_1) \cdot w_1 + f(x_2) \cdot w_2 = w_0 x_0 + w_1 x_1 + w_2 x_2 \stackrel{!}{=} \int_{-1}^1 f(x) \, dx = 0$$

Grad 2 mit $f(x) = x^2$:

$$f(x_0) \cdot w_0 + f(x_1) \cdot w_1 + f(x_2) \cdot w_2 = w_0 x_0^2 + w_1 x_1^2 + w_2 x_2^2 \stackrel{!}{=} \int_{-1}^1 f(x) \, dx = \frac{2}{3}$$

Weil wir oben schon die Positionen der Stützstellen (vereinfachtes Beispiel) gegeben haben, müssen wir die Gleichung noch höherer Grade (wir könnten bis Grad 5 aufschreiben) nicht mehr verwenden, weil das genügend Informationen sind (noch drei Unbekannte, drei Gleichungen).

$$w_0 + w_1 + w_2 = 2 \quad (I)$$

$$-w_0 + w_1 \cdot 0 + w_2 = 0 \quad (II)$$

$$w_0 + w_1 \cdot 0 + w_2 = \frac{2}{3} \quad (III)$$

Wir können schließen:

$$-w_0 + w_2 = 0 \quad (II)$$

$$w_0 = w_2$$

$$w_0 + w_2 = \frac{2}{3} \quad (III)$$

$$w_0 = \frac{2}{3} - w_2$$

Einsetzen in (II):

$$-\frac{2}{3} + w_2 + w_2 = 0$$

$$2w_2 = \frac{2}{3}$$

$$w_2 = w_0 = \frac{1}{3}$$

Den Rest erschließen:

$$w_0 + w_1 + w_2 = 2 \quad (I)$$

$$w_1 = 2 - w_0 - w_2 = 2 - 2w_2$$

$$w_1 = 2 - \frac{2}{3} = \frac{4}{3}$$

Also zusammenfassend könnte man sagen, dass die Fläche einer unbekannten Funktion mit Gauß-Quadratur mit Stützstellen $-1, 0$ und 1 folgendermaßen ausrechnet:

$$Q = f(-1) \cdot \frac{1}{3} + f(0) \cdot \frac{4}{3} + f(1) \cdot \frac{1}{3}$$

Das könnte bekannt vorkommen, da das einfach **Simpsonregel** (siehe [Gleichung 69](#)) für unser spezifisches Beispiel ist ☺.

Es ist äquivalent zu:

$$Q_S = 2 \cdot \frac{1}{6} \cdot (f(-1) + 4f(0) + f(1))$$

Erklärung

Sobald wir alle x_i und w_i aufgestellt haben, können wir die entstandene Formel nutzen, um das Integral einer Funktion anzunähern.

Das geht dann wie gewohnt mit folgender Formel:

$$\sum_{i=0}^{n-1} f(x_i) \cdot w_i \quad (79)$$

Die $f(x_i)$ müssen wir unter Umständen noch ausrechnen. In der Praxis ist das deswegen insofern relevant, weil die Gauß-Quadratur im Grunde über **Gewichtung und Stützposition** optimiert.

Sprich wenn ein Stützwert auszurechnen kostenaufwändig ist (weil man die Funktion zwar gegeben hat, diese aber sehr komplex ist auszuwerten), dann könnte man zu einer bestimmten Anzahl beliebig zu wählender Stützstellen die Quadratur mit Gauß optimieren.

Die Einschränkung bezüglich des Integrationsgebietes hebt sich da natürlich auf! Kann man sich eigentlich sowieso überlegen, wenn wir mit letztem Beispiel auf die Simpsonregel kamen, die ja auch allgemeingültig ist.

13 Fixpunkte

Erklärung

Was ist nochmal ein Fixpunkt?

Ein Fixpunkt ist ein Punkt x einer Funktion f , sodass $f(x) = x$ gilt. Der Punkt bildet über die Funktion auf sich selbst ab.

Einen solchen »Fixpunkt« zu finden ist bei einer beliebigen Funktion f nicht trivial.

13.1 Newton Verfahren

Erklärung

Das Verfahren von Newton kann die Nullstellen von $f(x)$ approximieren und bewerkstelligt dies mithilfe von Tangenten. Das Problem:

$$f(x) = a; \quad x \approx a$$

Herleitung: Approximation von Nullstellen mithilfe von Tangenten t in x_n

$$t(x) = f(x_n) + f'(x_n) \cdot (x - x_n)$$

$$t(x_{n+1}) = 0$$

$$f(x_n) + f'(x_n) \cdot (x_{n+1} - x_n) = 0$$

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

Vorgehen

Die Formel für das Newton Verfahren:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{80}$$

Informell bzw. graphisch bedeutet das: Man legt eine Tangente auf den Startwert x_0 und bestimmt davon die Nullstelle. Dann geht man an den Punkt der Funktion mit derselben x -Koordinate wie die Nullstelle und beginnt von vorne (Tangente anlegen, davon NST ausrechnen etc. ...).

Was hat das mit Fixpunkten zu tun? Naja, wenn man eine Nullstelle in das Newton Verfahren eingibt, kommt derselbe Punkt der Funktion wieder heraus. Nullstellen sind also Fixpunkte der Iterationsvorschrift vom Newton Verfahren.

Beispiel

$$f(x) = x^3 - 2x^2 - 1; \quad f'(x) = 3x^2 - 4x$$

Der Startpunkt sei $x_0 = 1$. Ich führe drei Schritte von Newton aus:

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= 1 - \frac{1 - 2 - 1}{3 - 4} = -1 \end{aligned}$$

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} \\ &= (-1) - \frac{(-1) - 2 - 1}{3 + 4} = -\frac{3}{7} \end{aligned}$$

$$x_3 = \left(-\frac{3}{7}\right) - \frac{\left(\frac{27}{343}\right) + \left(\frac{18}{49}\right) - 1}{\left(\frac{27}{49}\right) + \left(\frac{12}{7}\right)} = \dots$$

Dieses Verfahren würde sich der Nullstelle bei $x = 2.2056$ annähern.

Keine Sorge, in der Klausur werden nur solche Funktionen und Zahlen gewählt werden, dass ihr alles im Kopf ausrechnen könnt (also hier ist x_3 deutlich zu schwer im Kopf auszurechnen).

13.2 Eigenschaften

Erklärung

Allgemein versuchen Fixpunktverfahren (wie eben Newton) mit einem Fixpunkt x^* , der Iterationsvorschrift $\Phi(x)$ und einem Startwert x_0

$$x^* = \Phi(x^*) \quad (81)$$

mit Gleichungen der folgenden Form zu lösen:

$$x_{i+1} = \Phi(x_i) \quad (82)$$

Aber wo sind grafisch von der Iterationsfunktion $\Phi(x)$ die Fixpunkte der Funktion $f(x)$?

Man könnte es sich nicht einfacher vorstellen: Die Fixpunkte sind die Schnittpunkte der Graphen von $\Phi(x)$ und $f(x)$.

Die Iterationen kann man graphisch auch schön veranschaulichen: man bildet eine Treppe und springt zwischen Iterationsfunktion und der eigentlichen Funktion f hin und her. Also man fängt auf der Funktion am Startwert an und zieht eine gerade Linie nach links bzw. rechts, bis man auf $\Phi(x)$ trifft. Dann zieht man eine gerade Linie nach oben bzw. unten, bis man wieder auf $f(x)$ landet. Das wäre dann eine Iteration.

Also gerade Linien nach oben bzw. unten zeichnen, wenn man von $\Phi(x)$ kommt und wieder nach rechts bzw. links »abbiegen«, wenn man von $f(x)$ kommt.

Es gibt außerdem verschiedene Arten von Fixpunkten (von Φ abhängig!):

- $|\Phi'(x^*)| < 1 \implies$ anziehender Fixpunkt (konvergiert)
- $|\Phi'(x^*)| > 1 \implies$ abstoßender Fixpunkt (divergiert)
- $|\Phi'(x^*)| = 1 \implies$ indifferent

Anziehend bedeutet im Endeffekt (informell), dass Verfahren eine Tendenz haben, auf diesen Fixpunkt zuzugehen. Abstoßend ist hierbei verständlicherweise das Gegenteil. Grafisch würde die Treppe bei dem Fixpunkt konvergieren (anziehender Fixpunkt) oder davon weg divergieren (abstoßender Fixpunkt).

Beispiel

Es sei die Iterationsvorschrift $\Phi(x) = x^2$ gegeben. Wir sollen alle Fixpunkte bestimmen und zeigen, ob diese anziehend oder abstoßend sind.

Fixpunkte bestimmen:

$$\begin{aligned} x &= x^2 \\ 0 &= x^2 - x \\ \text{p-q-Formel: } x_{0,1} &= -\frac{-1}{2} \pm \sqrt{\frac{1}{4} - 0} \\ &= \frac{1}{2} \pm \frac{1}{2} \end{aligned}$$

Die Fixpunkte sind $x_0 = 0$ und $x_1 = 1$.

Welcher Art gehören die Fixpunkte an?

$$\begin{aligned} \Phi'(x) &= 2x \\ \Phi'(0) &= 0 < 1 \\ \Phi'(1) &= 2 > 1 \end{aligned}$$

Ergo ist $x_0 = 0$ ein anziehender Fixpunkt und $x_1 = 1$ ein abstoßender!

Erklärung

Im Normalfall konvergiert das Newton-Verfahren mit quadratischer Konvergenz. Bei mehrfachen Nullstellen besteht aber nur noch lineare Konvergenz. Wir können dieses Problem vorbeugen, indem wir das Näherungsverfahren modifizieren.

Bei einer k -fachen Nullstelle modifiziert man das newton'sche Näherungsverfahren mit einem Faktor k :

$$x_{n+1} = x_n - k \cdot \frac{f(x_n)}{f'(x_n)} \quad (83)$$

Erklärung

Wdh. mehrfache Nullstellen:

Falls jemand nicht weiß, was eine mehrfache Nullstelle x_0 von f ist, hier eine kurze Erklärung:

Für jedes mal, dass man f ableiten kann und x_0 immer noch eine Nullstelle ist, ist die Vielfachheit dieser Nullstelle x_0 um eins erhöht.

Formal:

Eine Nullstelle x_0 ist eine k -fache Nullstelle, wenn

$$f^{(j)}(x_0) = 0; \quad \forall j \in [0, k-1] \quad (84)$$

Sei f nun mindestens k -mal differenzierbar. Ist x_0 eine k -fache Nullstelle, aber keine $(k+1)$ -fache, so ist k die Vielfachheit der Nullstelle x_0 .

Beispiel

Gegeben sei

$$\begin{aligned} f(x) &= x^2 - 2x + 1 \\ x_0 &= 1 \end{aligned}$$

x_0 ist eine doppelte Nullstelle, da

$$\begin{aligned} f(1) &= 1^2 - 2 + 1 = 0 \\ f'(x) &= 2x - 2 \\ f'(1) &= 2 - 2 = 0 \end{aligned}$$

x_0 ist keine dreifache Nullstelle, weil

$$\begin{aligned} f''(x) &= 2 \\ f''(1) &= 2 \neq 0 \end{aligned}$$

Wichtig

Wir sagen, dass ein Iterationsverfahren $\Phi(x)$ genau **dann gut** ist, wenn die Lösung des Iterationsverfahren ein **anziehender Fixpunkt** ist.

Ansonsten wird der Fixpunkt halt nie durch das Verfahren erreicht oder es ist nicht einmal ein Fixpunkt, das wäre natürlich Quatsch!

Beispiel

Wir sollen zeigen, ob folgende Iterationsvorschriften sich zum Ausrechnen von Nullstellen einer beliebigen Funktion $f(x)$ eignen:

- Gerade, die nach oben verschoben ist:

$$\Phi_0(x) = x + 1$$

Wir schauen nach, ob Nullstellen Fixpunkte der Iterationsfunktion sind:

$$\Phi_0(0) = 0 + 1 = 1$$

Das ist nicht der Fall und somit eignet sich diese Vorschrift nicht für das Ausrechnen von Nullstellen!

- Eine Gerade durch den Ursprung:

$$\Phi_1(x) = 2x$$

$$\Phi_1(0) = 0$$

Null ist ein Fixpunkt der Vorschrift, ist dieser auch anziehend?

$$|\Phi_1'(0)| = 2 > 1$$

\implies abstoßend \implies eignet sich nicht

- Potenzfunktion:

$$\Phi_2(x) = x^3$$

$$\Phi_2(0) = 0$$

Null ist ein Fixpunkt, jetzt noch Eigenschaft des Fixpunktes überprüfen:

$$|\Phi_2'(0)| = 3 \cdot 0^2 = 0 < 1$$

0 ist ein anziehender Fixpunkt von $\Phi_2(x)$ und somit eignet sich die Vorschrift hierfür (da die Iterationsfunktion aber gar nicht von $f(x)$ abhängig ist lässt sich darüber streiten, wie sinnvoll das wirklich ist).

13.3 Banach'scher Fixpunktsatz

Erklärung

Der Banach'sche Fixpunktsatz sagt folgendes aus:

Es existiert genau ein anziehender Fixpunkt in einem Intervall I genau dann, wenn

1. I abgeschlossen
2. $\forall x \in I : \Phi(x) \in I$
3. Φ ist »Lipschitz-stetig« im Intervall I mit $0 < L < 1$:

$$\exists L : |\Phi(x) - \Phi(y)| \leq L \cdot |x - y| \quad (85)$$

Wenn die Bedingungen erfüllt sind, hat das auch noch eine andere Folgerung. Nicht nur ist in dem Intervall I dann genau ein anziehender Fixpunkt, sondern egal welchen Startwert $x_0 \in I$ man wählt, die Fixpunktiteration $\Phi(x)$ würde auf ebenjenen anziehenden Fixpunkt garantiert konvergieren.

Vorgehen

1. Ein Intervall I ist abgeschlossen, wenn es beide Grenzen enthält.
 $0 \leq x \leq 1$ wäre abgeschlossen,
 $0 < x < 1$ ist ein offenes Intervall.
2. Herausfinden, wann in dem Intervall die Funktionswerte am kleinsten und am größten sind, einsetzen und schauen, ob sich diese noch im Intervall befinden (Weil alle Werte dazwischen müssen sich dann auch im Intervall befinden)
 Gute Startüberlegung wäre »monoton steigend/fallend« und ähnliches.
3. Für diesen Punkt verwenden wir immer den Mittelwertsatz (MWS):

$$\exists \xi \in [a, b] : f(b) - f(a) = f'(\xi) \cdot (b - a) \quad (86)$$

Wenn man dies jetzt auf Φ anwendet gilt dann:

$$|\Phi(x) - \Phi(y)| \leq |x - y| \cdot \max_{z \in [x, y]} |\Phi'(z)| \quad (87)$$

Das gilt daher, weil man durch den Mittelwertsatz weiß, dass die zwei Gleichungen für ein bestimmtes ξ gleich ist und für den maximalen Wert der Ableitung im Intervall I also größer oder gleich der linken Seite sein muss.

Beispiel

Es sei die Iterationsvorschrift $\Phi(x) = x^2$ vom obigem Beispiel gegeben. Jetzt sollen wir mithilfe vom Fixpunktsatz beweisen, dass 0 ein anziehender Fixpunkt ist (bzw. wir zeigen, dass in einem Intervall, wo auch die 0 drin ist, ein anziehender Fixpunkt existiert!).

Sei

$$I = \left[-\frac{1}{4}; \frac{1}{4}\right]$$

1. I ist abgeschlossen, da trivial (bzw. das Intervall hat keine runde Klammern mit Inhalten ungleich Unendlich)
2. $\Phi(x)$ ist im Bereich von $[-0.25; 0]$ und $[0; 0.25]$ monoton fallend/steigend (Ableitung ist $2x$), daraus wissen wir, dass der kleinste/größte Funktionswert bei $x = 0$ bzw. $x = \pm 0.25$ liegen muss.

$$\Phi(-0.25) = 0.125$$

$$\Phi(0) = 0$$

Diese Funktionswerte liegen immer noch im Intervall I .

3. Auf Φ angewendet gilt:

$$|\Phi(x) - \Phi(y)| \leq |x - y| \cdot \max_{z \in [x, y]} |\Phi'(z)|$$

Wir wissen, dass $\Phi'(z) = 2z$ ist und der maximale Wert dieser Funktion in unserem Intervall I sich an der Stelle $z = 0.25$ befindet (die Ableitung ist streng monoton steigend).

$$|\Phi(x) - \Phi(y)| \leq \max_{z \in I} (2z) \cdot |x - y| = \frac{1}{2} \cdot |x - y|$$

Die »Lipschitz-stetigkeit« ist also mit $L = \frac{1}{2}$ erfüllt.

Die Fixpunktiteration konvergiert also für alle Startwerte aus dem Intervall $I := [-0.25; 0.25]$ gegen den Fixpunkt $x_0 = 0$!

14 Fixpunktiteration (LGS)

Erklärung

Beim letztem Mal haben wir Iterationsfunktionen kennengelernt und deren Nutzen im Bezug auf Fixpunkte erlernt.

Jetzt geht es darum, solch eine Iterationsfunktion für das Lösen von linearen Gleichungssystemen (=LGS) zu nutzen. Zu diesem Zwecke möchten wir ähnlich wie bei den Fixpunkten eine Iterationsvorschrift $\Phi(x)$ definieren, sodass wir iterativ die Lösung des Gleichungssystems annähern können.

14.1 Vektoriteration

Erklärung

Wir probieren uns eine Iterationsfunktion herzuleiten.

Die Idee ist, dass wir uns eine Matrix M definieren, die möglichst gut A approximiert, damit sie bei jedem Schritt die Lösung immer besser annähern kann.

Vorgehen

Herleitung:

$$Ax = b$$

$$0 = Ax - b$$

$$0 = M^{-1}(b - Ax)$$

$$x = x + M^{-1}(b - Ax)$$

$$\Phi(x_k) = x_k + M^{-1}(b - Ax_k) = x_{k+1} \quad (88)$$

Erklärung

Für M schauen wir uns folgende drei Vorschläge an. Wir haben im Hinterkopf, dass M möglichst gut invertierbar sein muss, weil wir ja M^{-1} brauchen und sie sollte die Matrix A gut approximieren können (damit das Verfahren schnell konvergiert).

- Richardson: $M = \mathbb{1}$
Perfekt invertierbar aber approximiert die (beliebige) Matrix A seeehr schlecht.
- Jacobi: $M = \text{diag}(A)$
Approximiert besser aber ist auch schlechter invertierbar.
- Gauß-Seidel: $M = A \cdot R$ (R ist links untere Dreiecksmatrix)
Approximiert die Matrix A am besten und ist immer noch gut invertierbar.

Wichtig

Eine volle Matrix zu invertieren kostet $\mathcal{O}(n^3)$. Das ist der Grund, warum wir als M nicht einfach die Matrix A nehmen, obwohl diese natürlich zur schnellsten Konvergenz führen würde. Irgendwie logisch, wir würden ja sonst das analytische ausrechnen und benötigten all das hier nicht.

Erklärung

Dadurch, dass wir beim ausrechnen immer $(b - Ax)$ benötigen, nennen wir dies ab sofort das **Residuum**

$$r^{(k)} = b - Ax^{(k)} \quad (89)$$

Lasst euch von dieser Potenzschreibweise nicht ablenken. Dadurch, dass wir nachher mehrere Indizes brauchen, beschreibt hier die »Potenz« mit Klammern unseren Iterationsschritt und der »normale« Index unter der Zahl geschrieben beschreibt quasi die »Zeile« (also von oben nach unten um welchen Eintrag des Vektors / der Matrix es sich handelt).

Wir schreiben a_{ii} für einen Diagonaleintrag der Matrix A , weil ii die Position des Eintrages in der Matrix beschreibt (i nach rechts und dann i nach unten beschreibt für beliebiges i die Diagonale der Matrix).

Wichtig ist, dass dies natürlich für jeglichen Größen $n \times n$ der Matrix funktioniert.

14.1.1 Jacobi Verfahren

Vorgehen

Die k -te »Jacobi« Iteration:

$$x^{(k+1)} = x^{(k)} + \text{diag}(A)^{-1} (b - Ax^{(k)}) \quad (90)$$

Wir teilen das auf in folgende Rechenschritte:

$$r^{(k)} = b - Ax^{(k)} \quad (91)$$

$$y_i^{(k)} = \frac{1}{a_{ii}} \cdot r_i^{(k)} \quad \forall i \in [1, \dots, n] \quad (92)$$

$$x^{(k+1)} = y^{(k)} + x^{(k)} \quad (93)$$

Beispiel

Jacobi Verfahren: $M = \text{diag}(A)$

Sei folgendes LGS gegeben:

$$\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

1. Iteration von Jacobi mit dem Startvektor $x^{(0)} = (0, 0)^T$

$$r^{(0)} = b - Ax^{(0)} = (2, 2)^T - (0, 0)^T = (2, 2)^T$$

$$y_1^{(0)} = \frac{1}{a_{11}} \cdot r_1^{(0)} = \frac{1}{1} \cdot 2 = 2$$

$$y_2^{(0)} = \frac{1}{a_{22}} \cdot r_2^{(0)} = \frac{1}{2} \cdot 2 = 1$$

$$x^{(1)} = x^{(0)} + y^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Jetzt käme die zweite Iteration, sprich man rechnet wieder alles von vorne nur jetzt mit dem neuen $x^{(1)}$ anstelle vom alten $x^{(0)}$

Und wenn man dies häufig genug macht, nähert sich das x der Lösung unseres LGS an!

14.1.2 Gauß-Seidel Verfahren

Erklärung

Von Jacobi zu Gauß-Seidel:

Der Unterschied von Jacobi zum Gauß-Seidel Verfahren ist das sogenannte »in-place« Prinzip. Das bedeutet, dass wir im selben Iterationsschritt schon Informationen / Werte verwenden, die wir im selben Schritt erst ausgerechnet haben. Lustigerweise ist das dasselbe wie wenn wir einfach für M die links untere Seite von A nehmen würden (anstelle der Diagonalmatrix wie bei Jacobi).

Heißt, wenn wir den i -ten Eintrag von r ausrechnen, verwenden wir für alle Einträge von 1 bis $(i - 1)$ nicht die Einträge vom alten Vektor, sondern direkt die neuen.

Wir trennen also $r = b - Ax^{(k)}$ in zwei Teile auf.

$$r_i^{(k)} = b_i - \sum_{m=1}^{i-1} a_{im}x_m^{(k+1)} - \sum_{m=i}^n a_{im}x_m^{(k)} \quad \forall i = 1, \dots, n \quad (94)$$

Diese Formel sieht viel komplizierter aus, als sie in der Praxis ist. Ich persönlich stelle es mir ähnlich eines Programmcodes vor, in dem neue Werte nicht separat abgespeichert werden, sondern direkt den jeweiligen Eintrag in unserer Lauf-variable x überschreiben.

Vorgehen

Die Methodik ändert sich also folgendermaßen: Die k -te »Gauß-Seidel« Iteration:

Für alle i von 1 bis n :

$$\begin{aligned} r_i^{(k)} &= b_i - \sum_{m=1}^{i-1} a_{im}x_m^{(k+1)} - \sum_{m=i}^n a_{im}x_m^{(k)} \\ y_i^{(k)} &= \frac{1}{a_{ii}} \cdot r_i^{(k)} \\ x_i^{(k+1)} &= y_i^{(k)} + x_i^{(k)} \end{aligned}$$

Zurück zur Schleife...

Beispiel

Selbes Beispiel wie bei dem Jacobi-Verfahren:

$$\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

1. Iteration von Gauß-Seidel mit dem Startvektor $x^{(0)} = (0, 0)^T$
Also (Programmcode-mäßig und damit informell):

$$\begin{aligned} x_{\text{current}} &:= x_c = x^{(0)} = (0, 0)^T \\ r_1^{(0)} &= b_1 - (Ax_c)_1 = 2 - 0 = 2 \\ y_1^{(0)} &= \frac{1}{a_{11}} \cdot r_1^{(0)} = \frac{1}{1} \cdot 2 = 2 \\ x_1^{(1)} &= (x_c)_1 + y_1^{(0)} = 0 + 2 = 2 \end{aligned}$$

$$x_c = (2, 0)^T \quad \text{Alten Wert überschreiben}$$

$$\begin{aligned} r_2^{(0)} &= b_2 - (Ax_c)_2 = 2 - 4 = -2 \\ y_2^{(0)} &= \frac{1}{a_{22}} \cdot r_2^{(0)} = \frac{1}{2} \cdot (-2) = -1 \\ x_2^{(1)} &= (x_c)_2 + y_2^{(0)} = 0 - 1 = -1 \end{aligned}$$

$$x^{(1)} = x_c = (2, -1)^T \quad \text{Ein Schritt ist fertig}$$

Hier käme jetzt also der nächste Schritt, der dann analog funktioniert.

Erklärung

Wir sehen am obigen Beispiel, dass Gauß-Seidel (für uns) deutlich schwieriger auszurechnen ist. Was man leider nicht sieht, aber dennoch der Fall ist: Gauß-Seidel nähert sich durch das »in-place« Prinzip der Lösung in aller Regel schneller an als Jacobi.

14.2 Verfahren des steilsten Abstieges

Erklärung

Das sogenannte »Verfahren des steilsten Abstieges« (steepest descent) soll eine weitere Alternative sein, um iterativ ein LGS lösen zu können. Diese ist an den Fall angepasst, wenn A symmetrisch und »positiv definit« (alle Eigenwerte > 0) ist.

Die grundlegende Idee ist es, dass wir uns die Funktion (offiziell Hyperfläche genannt)

$$f(x) = x^T A x - b^T x + c \quad (95)$$

definieren. Weil wenn wir das ableiten, erhalten wir

$$\nabla f(x) = Ax - b \quad (96)$$

und die Nullstellen davon entsprechen dann $Ax = b$, unserem LGS.

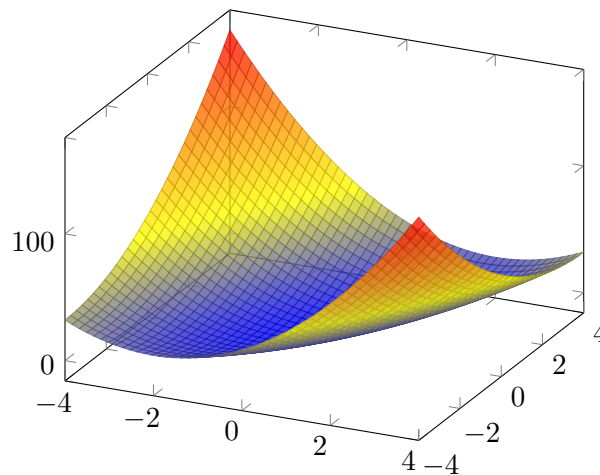


Abbildung 20: Beispiel einer Hyperfläche.

Vorgehen

In jeder Iteration dieses Verfahrens berechnet man folgende 3 Schritte:

1. Aktuelles Residuum: (»Schrittrichtung«)

$$r^{(k)} = b - Ax^{(k)} \quad (97)$$

2. Optimale Schrittweite:

$$\alpha^{(k)} = \frac{r^{(k)\top} \cdot r^{(k)}}{r^{(k)\top} \cdot Ar^{(k)}} \quad (98)$$

3. Aktuelles Zwischenergebnis:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \cdot r^{(k)} \quad (99)$$

Beispiel

Sei folgendes LGS (aus obigem Beispiel) gegeben und der Startvektor $x^{(0)} = (0, 0)^\top$

$$\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Wir berechnen die erste Iteration nach obiger Methode:

$$\begin{aligned} r^{(0)} &= b - Ax^{(0)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \\ \alpha^{(0)} &= \frac{r^{(0)\top} \cdot r^{(0)}}{r^{(0)\top} \cdot Ar^{(0)}} = \frac{8}{32} = \frac{1}{4} \\ x^{(1)} &= x^{(0)} + \alpha^{(0)} \cdot r^{(0)} = (0, 0)^\top + \frac{1}{4} \cdot \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{aligned}$$

Hier käme die nächste Iteration und erfolgt völlig analog...

Erklärung

Grafisch wird dieses Verfahren häufig so dargestellt, indem man die Höhenlinien der Hyperfläche als Kreise einzeichnet. Höhenlinien, weil ein Kreis bedeutet, dass alle Punkte auf dieser Kreislinie dieselbe Höhe (y-Wert) besitzen.

Das Verfahren konvergiert schneller, je gleichmäßiger es nach außen geht. Um genau zu sein braucht das Verfahren genau ein Schritt, falls die grafische Darstellung der Funktion (also die Höhenlinien) konzentrische Kreise bilden, also die Hyperfläche einem umgedrehten Kegel entspricht.

In der Praxis gleicht die grafische Darstellung meist eher weiten Ellipsen, weswegen dieses Verfahren genau so in der Praxis häufig recht langsam ist. Es gibt aber eine bessere Variante, das sogenannte »Conjugated gradient descent« (online nachschauen, wenn das Interesse geweckt hat).

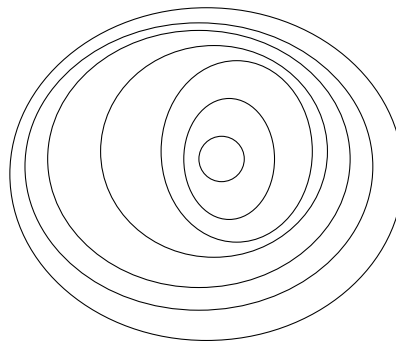


Abbildung 21: Beispielhafte grafische Darstellung eines »steepest descent« Problems.

Die Schrittrichtung in einem Schritt ist dann einfach immer senkrecht zur momentanen Höhenlinie, da es der jeweils steilste Abstieg in dem Punkt ist.

15 Eigenwertproblem

Erklärung

Eigenwerte kennt man vermutlich noch aus DS und Lineare Algebra. Wir wiederholen diese kurz und beschäftigen uns mit iterativen Verfahren, welche diese Art von Problem lösen.

Wir haben eine Matrix A , einen Vektor v und einen Skalar λ und

$$Av = \lambda v \quad (100)$$

v bezeichnet man hierbei als einen Eigenvektor und λ einen Eigenwert von A .

Formell bezeichnet ein Eigenvektor einer Abbildung ein vom Nullvektor verschiedener Vektor, dessen Richtung durch die Abbildung nicht verändert wird. Ein solcher Eigenvektor wird durch die Abbildung lediglich skaliert. Dieser Faktor ist der Eigenwert.

Ein typisches Beispiel wäre die Scherabbildung.

Vorgehen

Wie rechnet man pauschal einen Eigenvektor und -wert aus?

Gegeben sei eine Matrix A :

$$\det(A - \lambda_i \mathbb{1}) = 0 \quad (101)$$

Mit $\mathbb{1}$ als Einheitsmatrix, löse obige Gleichung, um auf alle Eigenwerte zu schließen. Mithilfe obiger Ausgangsbedingung

$$Av = \lambda v$$

leiten wir folgende Beziehung ab, um die Eigenvektoren auszurechnen:

$$(A - \lambda_i \mathbb{1}) \cdot v_i = 0 \quad (102)$$

Erklärung

Die Gleichung zum Ausrechnen des Eigenvektors können wir mithilfe von Gauß-Elimination lösen, da sie einem LGS entspricht. Beim Ausrechnen eines Eigenvektors haben wir immer **einen Freiheitsgrad**, es entsteht immer eine Nullzeile beim Gauß-Verfahren. Das liegt daran, dass wir nur an der Richtung des Eigenvektors interessiert sind, nicht dessen Länge.

Wir können also (wie immer bei Freiheitsgraden), sobald wir bei der Nullzeile sind, einen Wert des Vektors festlegen und alle anderen anhand des festgelegten berechnen. Es gibt also im Prinzip unendlich viele Lösungen, aber jeder Lösungsvektor ist nur ein Vielfaches eines anderen Lösungsvektors.

In der Praxis würde man den Vektor so bestimmen, dass die Norm Eins beträgt.

Beispiel

Eigenwerte:

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 2 \end{bmatrix}$$

$$\text{charakteristisches Polynom} := (A - \lambda \mathbb{1}) = \begin{bmatrix} 1 - \lambda & 0 \\ 2 & 2 - \lambda \end{bmatrix}$$

$$\det(A - \lambda \mathbb{1}) = (1 - \lambda) \cdot (2 - \lambda)$$

$$0 = 2 - \lambda - 2\lambda + \lambda^2$$

$$0 = 2 - 3\lambda + \lambda^2$$

P-Q Formel bzw. Mitternachtsformel...

$$\lambda_1 = 1$$

$$\lambda_2 = 2$$

Beispiel

Eigenvektoren:

Obiges Beispiel weitergeführt, wir möchten die Eigenvektoren zu obigem A erhalten.

$$(A - \lambda_i \mathbb{1}) v_i = 0$$

Für beliebigen Eigenwert umgesetzt:

$$\left[\begin{array}{cc|c} 1 - \lambda_i & 0 & 0 \\ 2 & 2 - \lambda_i & 0 \end{array} \right]$$

Für $\lambda_1 = 1$ eingesetzt:

$$\left[\begin{array}{cc|c} 0 & 0 & 0 \\ 2 & 1 & 0 \end{array} \right]$$

Hier sehen wir den einen Freiheitsgrad. Wir setzen (beispielsweise) die zweite Komponente auf 1. Es ergibt sich der Lösungsvektor und damit Eigenvektor des ersten Eigenwertes von A :

$$v_1 = \begin{pmatrix} -1/2 \\ 1 \end{pmatrix}$$

15.1 Kondition

Vorgehen

Um die Kondition eines Eigenwert- und Eigenvektorproblems zu bestimmen gehen wir davon aus, dass die Einträge der Matrix A einen fehlerhaften Offset von ϵ besitzen. Wie sonst auch immer rechnen wir den Algorithmus einfach mit beliebigem, nicht festem Fehler ϵ aus und schauen uns an, wie stark sich der Fehler auf die Ausgabe auswirkt.

15.2 Rayleigh Quotient

Erklärung

Der Rayleigh Quotient ist eine Möglichkeit, einen Eigenwert einer Matrix A mit gegebenem Eigenvektor v »einfach« auszurechnen.

Herleitung:

$$\begin{aligned} A \cdot v &= \lambda v \\ v^T \cdot A \cdot v &= \lambda \cdot v^T \cdot v \\ \lambda &= \frac{v^T \cdot A \cdot v}{v^T \cdot v} \end{aligned}$$

Vorgehen

Sei $v \in \mathbb{R}^n$ ein Eigenvektor einer Matrix $A \in \mathbb{R}^{n \times n}$. Der Eigenwert λ zu diesem Eigenvektor v der Matrix A kann wie folgt berechnet werden:

$$\lambda = \frac{v^T \cdot A \cdot v}{v^T \cdot v} \quad (103)$$

Beispiel

Obiges Beispiel:

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 2 \end{bmatrix} \quad v_1 = \begin{pmatrix} -1/2 \\ 1 \end{pmatrix}$$

Rayleigh Quotient nutzen, um auf den Eigenwert zu schließen:

$$\begin{aligned} \lambda &= \frac{\begin{pmatrix} -\frac{1}{2} & 1 \end{pmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 2 & 2 \end{bmatrix} \cdot \begin{pmatrix} -1/2 \\ 1 \end{pmatrix}}{\begin{pmatrix} -\frac{1}{2} & 1 \end{pmatrix} \cdot \begin{pmatrix} -1/2 \\ 1 \end{pmatrix}} \\ &= \frac{\begin{pmatrix} 1.5 & 2 \end{pmatrix} \cdot \begin{pmatrix} -1/2 \\ 1 \end{pmatrix}}{5/4} \\ &= \frac{(-3/4 + 2) \cdot 4}{5} \\ &= \frac{1}{4} \cdot 4 = 1 \end{aligned}$$

Stimmt, wie die vorherigen Beispielboxen zeigen ☺.

15.3 Vektor-/Poweriteration

Erklärung

Die sogenannte Poweriteration ist ein Verfahren, welches Eigenwerte annähern kann. Um genau zu sein, wirft man in diesen Algorithmus die Matrix A und einen Startvektor x_0 . Dieser x -Vektor wird sich mit jeder Iteration dem Eigenvektor für den betragsmäßig größten Eigenwert von A annähern.

15.3.1 Direct Power Iteration

Vorgehen

Die Iterationsvorschrift der Power-Iteration:

$$x_{k+1} = \frac{A \cdot x_k}{\|A \cdot x_k\|} \quad (104)$$

Der untere Teil des Bruchstrichs bedeutet einfach nur, dass nach jedem Schritt der Vektor wieder normalisiert wird.

Erklärung

Das Verfahren hat aus dem Grund seinen Namen, weil man das ganze auch rekursiv formulieren kann. Dann wäre es:

$$x_{k+1} = A^k \cdot x_0 \quad (105)$$

Dann würde man erst danach normalisieren. Die Normalisierung in jedem Schritt (siehe iterative Formel) macht man eh nur aus Stabilitätsgründen, da ansonsten die Zahlen immer größer werden was bei unserer Arithmetik tendenziell zu größeren Fehlern führt.

Da man auch am Eigenwert interessiert ist, nutzt man den Rayleigh **Quotient** um aus einer Power Iteration Approximation eines Eigenvektors den zugehörigen Eigenwert zu erhalten.

$$\lambda_{k+1} = \frac{x_k^T \cdot A \cdot x_k}{x_k^T \cdot x_k} \quad (106)$$

15.3.2 Shifted Power Iteration

Erklärung

Man kann aber auch noch einen sogenannten **Shift** μ einbauen. Mittels Shift kann die Matrix so manipuliert werden, dass die Eigenwerte um den Shift verschoben werden. Dadurch kann man erreichen, dass nun andere Eigenwerte der maximale Eigenwert sind und das Verfahren konvergiert gegen den entsprechend anderen Eigenwert.

Dieser Shift verschiebt im Grunde alle Eigenwerte um denselben Betrag nach unten. Also die Eigenwerte $\lambda_1 = 3$ und $\lambda_2 = 4$ mit einem Shift von $\mu = 4$ würden zu -1 und 0 werden. Mit diesem Shift würde die Power Iteration zu ersterem Eigenwert konvergieren, da dieser mit dem Shift der **betragsmäßig** größte ist.

Vorgehen

Shifted Power Iteration

$$x_{k+1} = \frac{(A - \mu \cdot \mathbb{1}) \cdot x_k}{\|(A - \mu \cdot \mathbb{1}) \cdot x_k\|} \quad (107)$$

Im Grunde genommen ist dieser Shift also ein konstanter Offset aller Diagonaleinträge von A jeder Iteration.

Erklärung

Wenn wir einen Shift haben müssen wir das beim Ausrechnen des Eigenwertes berücksichtigen und dessen Auswirkung rückwirkend mitberechnen. Wir müssen dann die Formel für den Rayleigh Quotient wie folgt anpassen:

$$\lambda_{k+1} = \frac{x_k^T \cdot (A - \mu \cdot \mathbb{1}) \cdot x_k}{x_k^T \cdot x_k} + \mu \quad (108)$$

Wichtig

Wenn zwei Eigenwerte durch den Shift **betragsmäßig** am größten sind, konvergiert die Power Iteration gar nicht mehr!

Beispiel

Wir führen eine Power-Iteration über die folgende Matrix A durch.

$$A = \begin{bmatrix} 2 & 2 \\ 4 & 0 \end{bmatrix}$$

Als Startvektor sei $x_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ gegeben.

Zwei Schritte mit einem Shift $\mu = 0$:

$$\begin{aligned} x_1 &= A \cdot x_0 \\ &= \begin{bmatrix} 2 & 2 \\ 4 & 0 \end{bmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 4 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} x_2 &= A \cdot x_1 \\ &= \begin{bmatrix} 2 & 2 \\ 4 & 0 \end{bmatrix} \cdot \begin{pmatrix} 2 \\ 4 \end{pmatrix} \\ &= \begin{pmatrix} 12 \\ 8 \end{pmatrix} \end{aligned}$$

Dies würde sich (normiert) dem Vektor $(1, 1)^T$ annähern, welche der Eigenvektor zu dem Eigenwert $\lambda_2 = 4$ ist, welcher der betragsmäßig größte von A ist.

Zwei Schritte mit einem Shift $\mu = 2$:

$$\begin{aligned} (A - \mu \mathbb{1}) &= \begin{bmatrix} 2-2 & 2 \\ 4 & 0-2 \end{bmatrix} \\ x_1 &= \begin{bmatrix} 0 & 2 \\ 4 & -2 \end{bmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 4 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} x_2 &= \begin{bmatrix} 0 & 2 \\ 4 & -2 \end{bmatrix} \cdot \begin{pmatrix} 0 \\ 4 \end{pmatrix} \\ &= \begin{pmatrix} 8 \\ -8 \end{pmatrix} \end{aligned}$$

15.3.3 Konvergenzrate

Erklärung

Die **Konvergenzrate** der direkten Power Iteration sagt aus, wie schnell sich der Algorithmus der exakten Lösung annähert. Sie ist definiert als:

$$q = \left| \frac{\lambda_2 - \mu}{\lambda_1 - \mu} \right| \in [0, 1] \quad (109)$$

Dabei ist λ_2 der betragsmäßig zweitgrößte und λ_1 der größte Eigenwert von A , den Shift jeweils schon einberechnet (verwirrend? Guck dir unten das Beispiel an!).

Die Eigenvektoren konvergieren linear (also quasi $\epsilon_{k+1} = \epsilon_k \cdot q$; mit ϵ als den Fehler) und die Eigenwerte quadratisch mit diesem Faktor. Aus diesem Grund ist das direkte Verfahren schlecht geeignet, wenn man viele Eigenwerte nahe dem maximalen Eigenwert hat. q geht hierbei gegen 1. Die Konvergenzrate liegt immer zwischen 1 und 0, wobei 0 Perfektion wäre, aka es konvergiert mit einem Schritt auf die analytisch korrekte Lösung.

Beispiel

Man hätte zwei Eigenwerte 3 und 12 und einen Shift von $\mu = 2$. Dadurch wäre der betragsmäßig größte Eigenwert $12 - 2$ und der zweitgrößte $3 - 2$ (mit dem Shift angegeben). Wir erhalten:

$$q = \left| \frac{3 - 2}{12 - 2} \right| = \frac{1}{10}$$

Das würde im Grunde also bedeuten, dass jede Power Iteration eine Dezimalstelle genauer wird.

Nehmen wir nun $\mu = 6$. Wir erhalten:

$$q = \left| \frac{3 - 6}{12 - 6} \right| = \frac{3}{6} = \frac{1}{2}$$

Obwohl mit beiden Shifts die Power Iteration zu demselben Eigenwert konvergiert, hat der Shift einen Einfluss auf die Konvergenzgeschwindigkeit!

Wählen wir dagegen $\mu > 7.5$, also beispielsweise $\mu = 10$, wird 3 zum betragsmäßig größtem Eigenwert. Wir erhalten:

$$q = \left| \frac{12 - 10}{3 - 10} \right| = \frac{2}{7}$$

15.3.4 Inverse Power Iteration

Erklärung

Natürlich gibt es dieses Verfahren noch invertiert. Hierbei konvergiert das Verfahren gegen den Eigenvektor des betragsmäßig **kleinsten** Eigenwertes von A .

Vorgehen

Mit

$$B = (A - \mu \cdot \mathbb{1})^{-1} \quad (110)$$

ergibt sich die Iterationsvorschrift:

$$x_{k+1} = \frac{B \cdot x_k}{\|B \cdot x_k\|} \quad (111)$$

Erklärung

Im obigen Methodenblock ist die Inverse Power-Iteration direkt mit Shift μ eingebaut worden, welcher analog zur nicht inversen Variante funktioniert.

Damit lassen sich dann ein paar nette Tricks machen. Möchte man den betragsmäßig kleinsten Eigenwert haben, lässt man die Inverse mit einem Shift von $\mu = 0$ laufen. Weiß man z.B., dass man drei Eigenwerte hat und den mittleren haben will, kann man die Inverse Power Iteration mit einem Shift von $0.5 \cdot (\lambda_1 + \lambda_2)$ laufen lassen. Durch das Benötigen der Inverse einer Matrix ist diese Methode aber laufzeit-technisch aufwändiger.

16 Differentialgleichungen

Erklärung

Differentialgleichungen, ernsthaft? Man weiß häufig nur, dass diese kompliziert, schwer zu lösen und nervig sind. Da dies (meiner Meinung nach) nicht stimmt, zeige ich dies folgendermaßen:

Eine »gewöhnliche Differentialgleichung« ist eine Gleichung, die eine Funktion durch ihre Ableitung beschreibt:

$$y'(t) = \varphi(y, t) \quad (112)$$

Ein Beispiel wäre eine Sonnenblume, die proportional zu ihrer momentanen Größe wächst:

$$y'(t) = 2y(t)$$

Die Lösung eines solchen Systems ist eine Funktion $y(t)$. In diesem Falle würde es die Größe der Blume in Abhängigkeit zu einem Zeitpunkt t angeben.

Doch wie löst man ein solches ODE? (»ordinary differential equation«)

16.1 Separation der Variablen

Vorgehen

Gegeben ist eine Differentialgleichung der Form

$$y'(t) = \varphi(y, t)$$

Methodik:

1. In Leibniz-Notation schreiben
2. Separieren
3. Integrieren
4. Integrale lösen
5. Nach y auflösen

Beispiel

Mit obiger Methode ergibt sich für unser Beispiel folgende Rechnung:

$$y'(t) = 2y(t)$$

In Leibniz Notation schreiben:

$$\frac{dy}{dt} = 2y$$

Separieren der Variablen:

$$\frac{dy}{2y} = dt$$

Integrieren:

$$\int_{y_0}^y \frac{1}{2\eta} d\eta = \int_{t_0}^t d\tau$$

Lösen und nach $y(t)$ auflösen:

$$\begin{aligned} \frac{1}{2}(\ln|y| - \ln|y_0|) &= t - t_0 \\ \left| \frac{y(t)}{y_0} \right| &= e^{2(t-t_0)} \\ y(t) &= \pm y_0 \cdot e^{2(t-t_0)} \end{aligned}$$

Dies nennt man die allgemeine Lösung des ODEs, da die Funktion für alle y_0 und alle t_0 die Differentialgleichung erfüllt.

16.2 Verkürzte Version**Vorgehen**

Folgende umgeschriebene Variante ist ein Versuch, die Separation der Variablen schöner darzustellen:

Gegeben ist eine Differentialgleichung der Form

$$y'(t) = f(t) \cdot g(y(t)) \tag{113}$$

Man stellt folgende Gleichung auf:

$$\int_{y_0}^y \frac{1}{g(y(\eta))} d\eta = \int_{t_0}^t f(\tau) d\tau \tag{114}$$

Nun löst man die entstandene Gleichung nach $y(t)$ auf und ist fertig.

Beispiel

Obiges Beispiel:

$$y'(t) = 2y(t)$$

Nach der Methodik ist $f(t) = 2$ und $g(y(t)) = y(t)$
Gleichung aufstellen:

$$\begin{aligned}\int_{y_0}^y \frac{1}{\eta} \, d\eta &= \int_{t_0}^t 2 \, d\tau \\ (\ln|y| - \ln|y_0|) &= 2 \cdot (t - t_0) \\ y(t) &= \pm y_0 \cdot e^{2(t-t_0)}\end{aligned}$$

Erklärung

y_0 und t_0 sind sogenannte Anfangsbedingungen.

Wenn man sie gegeben hat, löst man ein sogenanntes **AWP (=Anfangswertproblem)**. Hierzu muss man die allgemeine Lösung des ODEs in die spezifische übertragen, bei der man schlichtweg y_0 und t_0 einsetzt.

Solche Anfangsbedingungen könnten z.B. $t \geq 0$ und $y(0) = 1$ sein.

t_0 beschreibt den niedrigsten, noch gültigen Wert für t . Da t meist die Zeit beschreibt, ist es logisch, dass diese z.B. bei dem Wachstum einer Blume nicht im negativen anfängt.

Außerdem beschreibt y_0 den Wert am Startzeitpunkt t_0 :

$$y(t_0) = y_0 \tag{115}$$

Beispiel

Obiges Beispiel, obige Anfangsbedingungen, also:
ODE:

$$y'(t) = 2y(t)$$

Allgemeine, also Lösung des ODE:

$$y(t) = \pm y_0 \cdot e^{2(t-t_0)}$$

Anfangsbedingungen:

$$\begin{aligned} t \geq 0 &\longrightarrow t_0 = 0 \\ y(0) = 1 &\longrightarrow y_0 = 1 \end{aligned}$$

Spezifische, also Lösung des AWP:

$$\begin{aligned} y(t) &= 1 \cdot e^{2t} \\ &= e^{2t} \end{aligned}$$

16.3 Kleiner Trick**Vorgehen**

Wenn ihr ein ODE habt, welches nicht von t abhängig ist, sprich in folgender Form ist:

$$y'(t) = a \cdot y(t); \quad a \in \mathbb{R} \tag{116}$$

Dann ist die Lösung des ODEs **immer**:

$$y(t) = y_0 \cdot e^{a \cdot (t-t_0)} \tag{117}$$

Vertiefung

Für die Interessierten hier der Beweis für [16.3](#):

$$y'(t) = a \cdot y(t)$$

Mit obiger verkürzter Variante der Separation der Variablen:

$$\begin{aligned}\int_{y_0}^y \frac{1}{y(\eta)} \, d\eta &= \int_{t_0}^t a \cdot d\tau \\ \ln|y| - \ln|y_0| &= a \cdot \int_{t_0}^t d\tau = a \cdot (t - t_0) \\ \ln\left(\left|\frac{y}{y_0}\right|\right) &= a \cdot (t - t_0) \\ y(t) &= y_0 \cdot e^{a \cdot (t - t_0)}\end{aligned}$$

16.4 Kondition eines AWP

Erklärung

Um die Kondition eines Anfangswertproblems zu untersuchen, betrachten wir die Relation von einem Eingabe- und Ausgabefehler. Bei einem AWP wäre der Eingabewert der Startwert y_0 , während die Ausgabe die Funktion $y(t)$ entspricht.

Beispiel

Gegeben sei das exakte Ergebnis eines AWP:

$$y(t) = y_0 \cdot e^{4t}$$

mit $y_0 = 3$. Dann definieren wir uns eine verfälschte Eingabe $y_\epsilon = y_0 + \epsilon$. Dadurch wird die nun verfälschte Ausgabe zu:

$$\begin{aligned} y_\epsilon(t) &= y_\epsilon \cdot e^{4t} \\ &= (y_0 + \epsilon) \cdot e^{4t} \\ &= y_0 \cdot e^{4t} + \epsilon \cdot e^{4t} \\ &= y(t) + \epsilon \cdot e^{4t} \end{aligned}$$

Der Ausgabefehler an sich ist der Teil, der die korrekte Lösung verfälscht. Diese untersuchen wir dann für t geht gegen unendlich.

$$\lim_{t \rightarrow \infty} (\epsilon \cdot e^{4t}) = \infty$$

Das untersuchte AWP ist also schlecht konditioniert.

16.5 (Explizite) Einschrittverfahren

Erklärung

Einschrittverfahren bieten die Möglichkeit, ODEs schrittweise anzunähern. Im Folgenden sei δt immer unsere Schrittweite, also in wie große (oder grobe) Stücke wir (die Zeit) t unterteilen.

Bevor wir uns drei explizite Verfahren anschauen, wäre es wichtig, die grafische Darstellung eines ODE zu verstehen:

16.5.1 Richtungsfelder

Erklärung

Richtungsfelder bieten die Möglichkeit, das ODE graphisch anschaulich darzustellen. Hierfür stellen wir lediglich ein Graph mit x und $y(t)$ als Achsen auf. Punkte dieses Graphen stellen jeweils Werte der Ableitung da. Dafür ziehen wir bei jedem Punkt eine kleine Linie, welche die Steigung widerspiegelt.

Also wenn unser ODE $y'(t) = x^2 + y$ wäre, würden folgende Punkte folgende Steigungen haben:

$P(0, 1) \rightarrow 0^2 + 1 = 1$ hätte die Steigung 1.

$Q(1, 2) \rightarrow 1^2 + 2 = 3$ hätte die Steigung 3.

$R(2, 0) \rightarrow 2^2 + 0 = 4$ hätte die Steigung 4 und so weiter und so fort.

Zeichnet man genug dieser kleinen Linien in den Graphen kann man einen »Strom« erkennen, der auf die Lösung des ODEs vermuten lässt.

Wenn man jetzt (informell) einem Strom mit einem Stift folgt, zeichnet man eine mögliche Stammfunktion ein, welche das ODE erfüllt. Bedenke ja, dass man bei Stammfunktionen immer den konstanten Offset c hat, der beliebig sein kann. Und deswegen gibt es unendlich viele Pfeile im Richtungsfeld, denen man folgen kann und die trotzdem eine Lösung bilden.

Wenn man beispielsweise von der Funktion

$$y'(t) = 2t$$

das Richtungsfeld einzeichnen würde, könnte man ganz viele, vertikal verschobene Parabeln erkennen. Logisch, das Integral von $y'(t)$ ist:

$$y(t) = t^2 + c$$

Und genau dieses Prinzip nutzen wir jetzt für Schrittverfahren, die nichts anderes tun als Ableitungen (die Linien im Richtungsfeld) in Punkten zu nutzen, um einen Schritt auszurechnen (grafisch könnte man sagen, sie laufen den Pfeilen hinterher).

Erklärung

Für den Rest des Kapitels beschreibt $f(t_k, y_k)$ die Funktion, welche zu diskreten Zeitpunkten t_k und Werte y_k die Ableitung berechnet. Im Grunde ist es also nichts anderes als die rechte Seite des ODEs.

1. Beispiel:

$$\begin{aligned}y_0'(t) &= 2y(t) \\ f_0(t_k, y_k) &= 2y_k\end{aligned}$$

2. Beispiel:

$$\begin{aligned}y_1'(t) &= 4t \cdot 2y(t) \\ f_1(t_k, y_k) &= 4t_k \cdot 2y_k\end{aligned}$$

16.5.2 Explizites Euler-Verfahren (1. Ordnung)**Vorgehen**

Bei diesem Verfahren wird die Steigung im aktuellen Punkt t_k der Iteration zur Berechnung der nächsten Annäherung genutzt.

$$t_k = t_0 + k \cdot \delta t \quad (118)$$

$$y_{k+1} = y_k + \delta t \cdot f(t_k, y_k) \quad (119)$$

16.5.3 Heun-Verfahren (2. Ordnung)**Vorgehen**

Bei diesem Verfahren wird die Steigung in zwei Punkten (analog zur Trapezregel aus der Quadratur) der Iteration zur Berechnung der nächsten Annäherung genutzt.

$$t_k = t_0 + k \cdot \delta t \quad (120)$$

$$y_{k+1} = y_k + \frac{\delta t}{2} \cdot (f(t_k, y_k) + f(t_{k+1}, y_k + \delta t \cdot f(t_k, y_k))) \quad (121)$$

16.5.4 Runge-Kutta Verfahren (4. Ordnung)**Vorgehen**

Hier wird analog zur Simpsonregel aus der Quadratur noch mehr Punkte für die Iteration genutzt.

$$t_k = t_0 + k \cdot \delta t \quad (122)$$

$$T_1 = f(t_k, y_k) \quad (123)$$

$$T_2 = f\left(t_k + \frac{\delta t}{2}, y_k + \frac{\delta t}{2} T_1\right) \quad (124)$$

$$T_3 = f\left(t_k + \frac{\delta t}{2}, y_k + \frac{\delta t}{2} T_2\right) \quad (125)$$

$$T_4 = f(t_{k+1}, y_k + \delta t \cdot T_3) \quad (126)$$

$$y_{k+1} = y_k + \frac{\delta t}{6} \cdot (T_1 + 2T_2 + 2T_3 + T_4) \quad (127)$$

Beispiel

Euler-Verfahren:

Wir haben die Startwerte $y_0 = 1$; $t_0 = 0$, außerdem ist $\delta t = 1$ und unser ODE:

$$y'(t) = 2y(t)$$

Unsere Funktion ist also

$$f(t_k, y_k) = 2y_k$$

1. Schritt:

$$t_1 = t_0 + k \cdot \delta t = 0 + 1 \cdot 1 = 1$$

$$y_1 = y_0 + \delta t \cdot 2y_0 = 1 + 1 \cdot 2 \cdot 1 = 3$$

2. Schritt:

$$t_2 = t_0 + k \cdot \delta t = 0 + 2 \cdot 1 = 2$$

$$y_2 = y_1 + \delta t \cdot 2y_1 = 3 + 1 \cdot 2 \cdot 3 = 9$$

Und so weiter, und so fort ...

16.6 Diskretisierungsordnung**Erklärung**

Solche Schrittverfahren haben immer eine Diskretisierungsordnung p (die ich bei den Verfahren oben schon dazugeschrieben habe). Diese gibt im Grunde die Genauigkeit des Verfahrens an. Mit einer Schrittweite von δt liegt der globale Diskretisierungsfehler eines Verfahrens in

$$\mathcal{O}(\delta t^p) \tag{128}$$

Das bedeutet zum Beispiel auch, dass wenn man die Schrittweite bei einem Verfahren zweiter Ordnung halbieren würde, sich der Fehler vierteln würde.

Erklärung**Problem:**

Damit explizite Verfahren für uns genau genug werden, müsste man eine möglichst kleine Schrittweite wählen. Damit wiederum explodiert aber unser Rechenaufwand für den gleichen Bereich. Außerdem würden wir dann eine kleine Zahl mit »normal« großen Zahlen verrechnen was wiederum an unserer Maschinengenauigkeit nagt.

Aus diesem Grunde schauen wir uns noch ein Beispiel eines **impliziten** Einschrittverfahrens an.

16.7 Implizites Euler-Verfahren (1. Ordnung)**Erklärung**

Die Idee vom impliziten Euler Verfahren ist es, die Steigung im nächsten Schritt zu antizipieren und damit in der aktuellen Iteration zu rechnen.

Implizite Verfahren haben in der Regel eine wesentlich bessere Stabilität, da man meist größere Zeitschritte wählen kann als bei expliziten Verfahren. Jedoch muss man auch ein Gleichungssystem in jedem Iterationsschritt lösen, da auch auf der rechten Seite der Gleichung ein y_{k+1} steht.

Vorgehen

Wir wenden folgende Formel iterativ an:

$$y_{k+1} = y_k + \delta t \cdot f(t_{k+1}, y_{k+1}) \quad (129)$$

Tipp: Wenn man das umformt, kann es sein, dass man auf ein Nullstellen Problem kommt (p-q Formel bzw. Mitternachtsformel lässt grüßen).

$$0 = y_k + \delta t \cdot f(t_{k+1}, y_{k+1}) - y_{k+1} \quad (130)$$

Beispiel

Gegeben sei:

$$y_0 = -1 \quad \delta t = 1 \quad t_0 = 0 \quad y(t) < 0$$

mit der Funktion (schon aus dem ODE ausgelesen):

$$f(t_k, y_k) = 2 \cdot t_k \cdot y_k^2$$

1. Iteration:

$$y_{k+1} = y_k + \delta t \cdot f(t_{k+1}, y_{k+1})$$

$$y_1 = y_0 + \delta t \cdot 2 \cdot t_1 \cdot y_1^2$$

$$y_1 = -1 + 1 \cdot 2 \cdot y_1^2$$

$$0 = 2y_1^2 - y_1 - 1$$

Mit P-Q Formel erhält man die Lösungen $y_{1,1} = 1$ und $y_{1,2} = -\frac{1}{2}$. Da wir aber oben gegeben haben, dass $y(t) < 0$ gilt, kann 1 nicht die richtige Lösung sein. Also haben wir $y_1 = -\frac{1}{2}$.

2. Iteration:

$$y_{k+1} = y_k + \delta t \cdot f(t_{k+1}, y_{k+1})$$

$$y_2 = y_1 + \delta t \cdot 2 \cdot t_2 \cdot y_2^2$$

$$y_2 = -\frac{1}{2} + 1 \cdot 2 \cdot 2 \cdot y_2^2$$

$$0 = -\frac{1}{2} + 4 \cdot y_2^2 - y_2$$

$$0 = y_2^2 - \frac{1}{4}y_2 - \frac{1}{8}$$

P-Q Formel nutzen:

$$y_{2,1,2} = \frac{1}{8} \pm \sqrt{\frac{1}{64} + \frac{1}{8}}$$

Die Lösungen davon sind 0.5 und -0.25 . Auch hier können wir mit gegebener Bedingung schließen, dass $y_2 = -0.25$ sein muss.

16.8 Wichtige Begriffe

Erklärung

Lokaler Diskretisierungsfehler

Ist der Fehler, den wir in einem Schritt haben, wenn man annimmt, dass der ursprüngliche Wert y_k korrekt wäre (Quasi der Fehler von y_{k+1} zu der exakten Lösung mit Anfangswert y_k). Also quasi wieviel Fehler durch das Schrittverfahren in einem beliebigen Schritt hinzukommen kann.

Als Beispiel gilt bei der Eulermethode für den lokalen Diskretisierungsfehler:

$$\ell(\delta t) := \max_{a \leq t \leq b - \delta t} \left(\left| \frac{y(t + \delta t) - y(t)}{\delta t} - f(t, y(t)) \right| \right) \quad (131)$$

Eine nicht äquivalente, aber leichter (und damit häufiger genutzt) in der Praxis handhabbare Definition beschreibt den Fehler mit $y_k = y(t_k)$ als:

$$|y_{k+1} - y(t_{k+1})| \quad (132)$$

Erklärung

Globaler Diskretisierungsfehler

Ist der tatsächliche Fehler von einem y_k zu der eigentlichen Lösung mit Anfangswert y_0 . Also um wieviel unterscheidet sich die exakte Lösung an einer bestimmten Stelle zu der von einem Einschrittverfahren. Gibt also an, wie gut unser Verfahren am Ende ist und damit relevanter als der lokale Diskretisierungsfehler.

Mathematisch, mit Approximationen y_k und analytischen Werten $y(t_k)$ gilt:

$$e(\delta t) := \max_{k=0, \dots, N} (|y_k - y(t_k)|) \quad (133)$$

Eine nicht äquivalente aber gebräuchliche Form des globalen Diskretisierungsfehlers schaut sich lediglich die Differenz am Ende des betrachteten Zeitintervalls an:

$$|y_n - y(t_n = b)| \quad (134)$$

Erklärung

Konsistenz

Ein Schrittverfahren ist konsistent, wenn der lokale Diskretisierungsfehler für kleine Zeitschrittweiten gegen 0 geht.

$$\lim_{\delta t \rightarrow 0} \ell(\delta t) \rightarrow 0 \quad (135)$$

Erklärung**Konvergenz**

besagt, dass der globale Diskretisierungsfehler bei kleinen Zeitschritten auch gegen 0 geht. Konvergenz (der stärkere Ausdruck) impliziert Konsistenz aber nicht umgekehrt. (Nur weil wir lokal immer kleinere Fehler machen heißt es nicht, dass das auch global stimmt).

Es gilt:

Konvergenz \implies Konsistenz

Erklärung**Stabilität**

Stabil nennt man eine Lösung, welche gegenüber kleinen Störungen der Eingabe unempfindlich ist. Wenn sich kleine lokale Fehler nur zu kleinen globalen Fehlern aufsummieren, ist ein Verfahren stabil.

Es gilt:

Konsistenz + Stabilität \iff Konvergenz

Erklärung**Steifheit**

Eine Differentialgleichung nennt man **steif**, die zwar die Eigenschaft der Konsistenz und/oder Konvergenz aufweisen, diese jedoch nur für sehr kleine δt .

Ein zu kleines δt kann jedoch unpraktikabel sein, somit ist **Steifheit** eine Problemeigenschaft.

Erklärung

Wenn wir dieses Wissen auf unsere Verfahren anwenden, sehen wir folgendes:
Expliziter Euler ist Verfahren erster Ordnung, also gilt:

$$\ell(\delta t) = \mathcal{O}(\delta t), \quad e(\delta t) = \mathcal{O}(\delta t)$$

Bei Heun wäre es halt $\mathcal{O}(\delta t^2)$ und Runge Kutta $\mathcal{O}(\delta t^4)$ jeweils.

Alle drei Verfahren sind konsistent und konvergent!

16.9 Quadratur und AWP-Lösung

Erklärung

Wie schon in [Kapitel 16.5.2](#) kurz erwähnt lässt sich eine Analogie zwischen Quadratur und Lösen von AWP's feststellen.

Um das zu veranschaulichen werden einmal die »dumme Rechtecksregel« in das explizite Euler Verfahren umgewandelt sowie die Trapezregel zum Heun-Verfahren.

Dafür muss man nur die Integralgrenzen mit den jeweiligen Zeitwerten t_k ersetzen und das ganze schrittweise betrachten.

Beispiel

Die »dumme Rechtecksregel« ist einfach die normale Rechtecksregel, nur dass es den Punkt am linken Rand des Intervalls verwendet und nicht jenen in der Mitte.

$$\begin{aligned}\int_a^b f(t) \, dt &\approx (b-a) \cdot f(a) \\ y_{k+1} - y_k &= (t_{k+1} - t_k) \cdot f(t_k, y_k) \\ y_{k+1} &= y_k + \delta t \cdot f(t_k, y_k)\end{aligned}$$

Beispiel

Die Trapezregel:

$$\begin{aligned}\int_a^b f(t) \, dt &\approx (b-a) \cdot \frac{1}{2} \cdot (f(a) + f(b)) \\ y_{k+1} - y_k &= (t_{k+1} - t_k) \cdot \frac{1}{2} \cdot (f(t_k, y_k) + f(t_{k+1}, y_{k+1})) \\ y_{k+1} &= y_k + \delta t \cdot \frac{1}{2} \cdot (f(t_k, y_k) + f(t_{k+1}, y_{k+1}))\end{aligned}$$

Da wir $f(t_{k+1}, y_{k+1})$ nicht besitzen, müssen wir es mit explizitem Euler Verfahren annähern. Ergo erhalten wir:

$$y_{k+1} = y_k + \delta t \cdot \frac{1}{2} \cdot (f(t_k, y_k) + f(t_{k+1}, y_k + \delta t \cdot f(t_k, y_k)))$$

Das entspricht genau unserer Heun-Verfahren Definition ☺.

16.10 Mehrschrittverfahren

Erklärung

Es gibt nicht nur explizite und implizite Einschrittverfahren, sondern auch noch die Klasse der Mehrschrittverfahren.

Deren Grundidee ist es, die in früheren Schritten approximierten Werte für den nächsten Schritt mitzubenutzen. Also quasi nicht die Werte des letzten Schrittes nutzen, sondern nur oder auch von Schritten davor.

16.10.1 Mittelpunktsregel

Vorgehen

Formel zur Berechnung eines Schrittes mit der MPR (=Mittelpunktsregel)

$$y_{k+1} = y_{k-1} + 2\delta t \cdot f(t_k, y_k) \quad (136)$$

Um damit starten zu können brauchen wir allerdings nicht nur einen Startwert y_0 , sondern auch schon y_1 . Das macht man in aller Regel mit einem explizitem Einschrittverfahren (z.B. Euler-Verfahren).

Beispiel

Gegeben sei:

$$f(t_k, y_k) = 2t_k \cdot y_k \quad ; \quad y_0 = 1 \quad ; \quad t_0 = 0 \quad ; \quad \delta t = 1$$

Wir berechnen y_1 mithilfe von explizitem Euler-Schritt:

$$\begin{aligned} y_1 &= y_0 + \delta t \cdot f(t_0, y_0) \\ &= 1 + 1 \cdot (2 \cdot 0 \cdot 1) \\ &= 1 \end{aligned}$$

Erster Schritt mit der Mittelpunktsregel:

$$\begin{aligned} y_2 &= y_0 + 2\delta t \cdot f(t_1, y_1) \\ &= 1 + 2 \cdot (2 \cdot 1 \cdot 1) \\ &= 1 + 4 \\ &= 5 \end{aligned}$$

Zweiter Schritt mit der Mittelpunktsregel:

$$\begin{aligned}y_3 &= y_1 + 2\delta t \cdot f(t_2, y_2) \\&= 1 + 2 \cdot (2 \cdot 2 \cdot 5) \\&= 1 + 40 \\&= 41\end{aligned}$$

Erklärung

Wir klären in der Tutorstunde, warum dieses Verfahren instabil ist.

17 Tipps für die Klausur

Hier sind ein paar Erfahrungen, die ich beim Korrigieren von Klausuren gemacht habe. Aber keine Garantie, solche Dingen ändern sich gerne mal!

- **Aufgaben beliebig lösen:**

Wenn **nicht klar gesagt** wird, wie man eine Aufgabe zu lösen hat, löse sie so, wie du es willst (und vorallem am schnellsten bist). Pass aber auf! Es könnte sein, dass indirekt ein Verfahren deutlich schneller ist und das nicht in der Aufgabe steht!

- **Logiklücken und Fehler in Aufgaben**

Wenn du einen Fehler in einer Aufgabe findest oder denkst, dass du einen Weg genommen hast, der nicht bedacht war und der die Aufgabe z.B. deutlich vereinfacht hat, schreibe einen Satz, wie du darauf kommst. Das wird mit einbezogen. Logischerweise gibt es aber keine Punkte, wenn dieser Weg dann falsch ist.

- **Überprüft eure Lösungen mit Bauchgefühl:**

Es gibt Aufgaben, da kann man aus einer Grafik ablesen, was die Lösung einer Rechenaufgabe (vielleicht auch nur grob) sein müsste. In einer Klausur gab es z.B. eine einfach Quadratur Aufgabe und man konnte aus der Grafik sehen, dass die Fläche unter der Funktion negativ und betragsmäßig recht klein sein musste. Wenn man dann halt auf Zahlen wie 12 kommt, sollte man sich seine Rechnung vielleicht noch einmal durchgehen. Das gleiche gilt auch für Verständnisaufgaben! Wenn man nämlich (als Beispiel) erkennt, dass seine Lösung falsch ist, aber nicht die Zeit hat, diese zu korrigieren, schreibt als Text hin, dass ihr erkannt habt, dass die Lösung falsch sein muss. Dafür gibt es Punkte, weil wir sehen, dass ihr es verstanden habt.

- **Aufbau:**

- In der Klausur gibt es normal 5-6 Abschnitte mit jeweils mehreren Teilaufgaben. Jeder Abschnitt repräsentiert meist ein Oberkapitel. Dementsprechend gibt es mehrere Themen, die nicht abgefragt werden.
- Es werden **mindestens** 1-2 Aufgaben drankommen, die in der Kategorie der grafischen Aufgaben eingeordnet werden kann. Also Informationen aus einer Grafik auslesen, selber etwas in eine Grafik einzeichnen, anhand einer argumentieren oder oder oder ...
- Die letzte Aufgabe der Klausur ist (immer) eine **Programmieraufgabe**. Hierbei gilt, dass man in Java oder Java-ähnlichem Pseudocode geschrieben werden muss. Solange ihr nicht von irgendwelchen Packages ausgeht, die euch alle Funktionalitäten abnehmen und euer Code verständlich ist, alles super!
- Es gibt immer 2-3 Teilaufgaben, die sehr schwierig sind bzw. **nicht in den Tutorien** behandelt worden sind (und die ich vermutlich ohne Üben auch nicht schaffen würde). Macht euch darüber keine Sorge, ohne diese kann man immer noch eine 1,3 schreiben. ☺

- **Verständnis wichtiger als Rechenkünste:**

- Es gibt mehr Punktabzug, wenn ihr was falsch macht und wir sehen, dass es nicht verstanden wurde. Außerdem gibt es häufiger Verständnisfragen (meist nach einer kleinen Rechenaufgabe).
- Wir korrigieren vollständig mit **Folgefehler**. Aus diesem Grund ist Verständnis noch einmal wichtiger als Rechenkünste. Wobei ihr euch natürlich nicht nur auf dem Grundschulniveau der Rechenkünste befinden solltet. Im Allgemeinen gibt jeder mathematische Fehler, der nicht durch Folgefehler entstanden ist, **einen halben Punkt Abzug**.

- **Textantworten sind ok!**

Wenn ihr Schwierigkeiten habt, probiert eure Antwort wenigstens in Worten zu beschreiben, anstatt es leer zu lassen. Wenn wir anhand von den wenigen Worten erkennen können, dass ihr es eigentlich verstanden habt, **gibt es noch Punkte!**

- **Die Punktevergabe ist anders je Aufgabenteil.**

Es lohnt sich also, am Anfang der Klausur über alle Aufgaben drüberzuschauen und sich die Aufgaben auszusuchen, die man selber nicht nur schnell lösen kann, sondern die auch mehr Punkte geben.

- **Zu billige Aufgaben?**

Es kann durchaus seeehr billige Aufgaben in der Klausur geben. Also wenn ihr denkt, dass eine Aufgabe zu einfach um wahr zu sein ist, überprüft nochmal alles. Aber wenn ihr dann weiterhin sie für einfach haltet, ist sie das wahrscheinlich auch.

- **Hinweise auf dem Klausurblatt**

Bei einigen Aufgaben stehen Hinweise. Die stehen da meist nicht ohne Grund also nutzt sie! Falls ihr aber eine Aufgabe ohne den Hinweis lösen könnt, dann ist das auch super, nur eher unwahrscheinlich. 😊

- **Ruhe**

Das Wichtigste: immer Ruhe bewahren. Numprog ist cool, und so ist auch die Klausur. Macht euch keine Panik und liest euch alles gemütlich durch. Und hey, wenn ihr es nicht besteht, findet Numprog ja immerhin jedes Semester statt. 😊

18 Typische Fehler

Hier liste ich mal ein paar Fehler auf, die mir persönlich beim Korrigieren von Numprog Klausuren aufgefallen sind. Alles etwas querbeet und chaotisch, aber vielleicht trotzdem hilfreich. ☺

- Aufgabenstellung nicht sorgfältig genug durchgelesen und Teilaufgaben deswegen nicht beantwortet.
 - Gerade so Teile wie »allgemeine Formeln« angeben werden häufiger vernachlässigt. Darauf gibt es Punkte, sonst würde es nicht in der Aufgabe stehen!
 - Wenn nach einer »kleinsten Zahl größer Null« gefragt ist, darf die Antwort keine negative Zahl sein.
- Hinweise einer Aufgabe nicht durchgelesen. Zum Teil stehen da alle Formeln, die man für die Aufgabe benötigt.
- Wenn eine Zahlendarstellung x Bytes zur Verfügung hat, ist das nicht äquivalent zu x Bits, sondern $8 \cdot x$ Bits.
- Umwandlung zu IEEE:
 - Null steht beim Vorzeichenbit für eine positive Zahl!
 - Bei der Bestimmung des Exponenten vergessen, den konstanten Offset (die -127) einzuberechnen.
 - Die Anzahl an Mantissenbits vertan (es sind 23 Bits, ohne die erste Eins, da die nicht abgespeichert werden muss!) & vergessen, dass man runden muss (falls die Mantisse eigentlich länger als 23 Bits ist)
 - Klar spezifizieren, welche Teile Vorzeichen, Exponent und Mantisse ist.
- Die Ableitung von x^2 ist $2x \dots$
- Stabilität & Kondition verwechselt. Das ist nicht dasselbe!
- Für eine Limes Untersuchung kann häufig der »L'Hopital« weiterhelfen.
- Bei der LR-Zerlegung nach der eigentlichen Zerlegung und Vorwärtssubstitution aufgehört, obwohl noch die Rückwärtssubstitution durchgeführt werden muss.
- Beim Lösen eines überbestimmten Gleichungssystem einfach am Anfang die Matrix so ausschneiden, dass sie quadratisch wird, ist nicht erlaubt!
- Bei einem überbestimmten Gleichungssystem zum Lösen Gauß-Eliminationsverfahren anwenden ist Quatsch.
- Polynome sind stets beliebig häufig stetig differenzierbar.

- Wenn bei einer Quadratur Rechenaufgabe etwas negatives herauskommt und man bei der Zeichnung eine Fläche einzeichnet, die insgesamt positiv ist sollte man vielleicht noch einmal über seine Rechnung drüberschauen.
- Bei Lagrange-Basen den Aufwand machen, das LGS aufzustellen, obwohl die Matrix mit Lagrange eh immer einer Einheitsmatrix entspricht.
- Implizites Euler-Verfahren **ist NICHT** genauer als Explizites Euler-Verfahren. Beide haben Ordnung 1. Das implizite ist stabiler, aber nicht (allgemein) genauer!
- Runge Kutta ist mit 4. Ordnung genauer als Heun mit 2. Ordnung, beide sind genauer als Explizites- sowie Implizites Euler Verfahren, da diese nur Ordnung 1 haben.
- Bei einer Gauß-Quadratur Aufgabe die Gewichtungen ausrechnen, obwohl diese in der Aufgabe gegeben sind.
- Zahlen aus der Aufgabe falsch abgeschrieben...
- Wenn eine Iterationsformel ala $x_{k+1} = x_k + a(x_k)$ gegeben ist, ist die Iterationsvorschrift einfach nur $\Phi(x) = x + a(x)$.
- Einzeichnen einer Trapezregel: es heißt nicht umsonst Trapezregel, also eine Gerade oder Fünfeck einzeichnen ist irgendwo falsch. Es kann aber durchaus sein, dass ein Dreieck oder Rechteck entsteht.
- Bei grafischen Aufgaben sich die Koordinatenachsen nicht richtig angeschaut. Manchmal ist links unten nicht der Ursprung. Also schaut erst einmal an, wo $(0,0)$ überhaupt ist.
- Wenn eine Frage etwas ist wie »Was beobachten Sie? Warum?«, dann ist das eigentlich nie zum Spaß da. Falls es also nichts zu beobachten gibt hast du vielleicht etwas falsch ausgerechnet.
- Nicht in der Lage sein, die Winkelhalbierende einzuzichnen (das ist eine Gerade mit Steigung 1, die durch den Ursprung geht)

19 Abschlussworte

Dieses Projekt ist zwar zunächst als Witz und Experiment entstanden, doch hat es sich stetig weiterentwickelt. Das hätte ich aber niemals ohne Hilfe geschafft.

Herzlich bedanken möchte ich mich bei...

- **Lucas Wolf:**

Nicht nur war er mein Tutor in *Numerisches Programmieren* und verdanke ihm den Großteil meines Wissens, er hat mich auch erst dazu gebracht, Tutor in diesem Fach zu werden und hat mir für dieses Projekt auch seine ganzen Unterlagen bereitgestellt.

- **Michael Obersteiner und Michael Rippl:**

Zwei fantastische Übungsleiter, die uns Tutoren immer unterstützt haben und die mir schon häufig, schnell und präzise, Fragen beantwortet haben.

- **Meine Studenten:**

Für die stetig lustige (wenn auch nicht lehrreiche höhö) Zeit. Ihr seid meine Hauptmotivation, an diesem Dokument weiterzuarbeiten. Es ist immer schön, von seinen Studenten ausgelacht oder gedisst zu werden 😊.

- **Allen Lesern:**

Vielen lieben Dank! Danke an euch, die mir Mails geschickt haben, um Fehler zu melden. Vorschläge gemacht, was man besser machen könnte. Ihr, die mich immer auf Trab gehalten und wertvolle Tipps gegeben habt.

Ich wünsche euch viel Erfolg bei euren Klausuren, haut rein. Ihr besteht das alle! (außer vielleicht die Studenten, die in meine Tutorien gegangen sind 😊)

Liebe Grüße
Hendrik

Ich gehe jetzt damit angeben, dank euch Lesern eine riesige Anzahl an Klicks auf meiner Webseite zu haben! 😊