

Skript Numerik I

von Prof. Dr. Luise Blank im WS14/15

Gesina Schwalbe

7. Januar 2015

Inhaltsverzeichnis

1	Einführung	5
2	Lineare Gleichungssysteme: Direkte Methoden	13
2.1	Gaußsches Eliminationsverfahren	13
2.1.1	Vorwärtselimination	13
2.1.2	Rückwärtselimination	15
2.1.4	Weitere algorithmische Anmerkungen	16
2.1.7	Algorithmus: Gauß-Elimination zur Lösung von $Ax = b$	17
2.1.8	Rechenaufwand gezählt in „flops“	17
2.1.10	Allgemeines zur Aufwandsbetrachtung	18
2.1.11	Formalisieren des Gauß-Algorithmus: LR-Zerlegung	19
2.2	Gaußsches Eliminationsverfahren mit Pivotisierung	22
2.2.1	Spaltenpivotisierung	22
2.2.3	Algorithmus: Gauß-Elimination mit Spaltenpivotisierung	24
2.2.5	Lösen eines Gleichungssystems $Ax = b$	26
3	Fehleranalyse	29
3.1	Zahlendarstellung und Rundungsfehler	29
3.1.3	Bit-Darstellung zur Basis 2	30
3.1.4	Verteilung der Maschinenzahlen	31
3.1.6	Rundungsfehler	32
3.1.8	Auslöschung von signifikanten Stellen	33
3.2	Kondition eines Problems	34
3.3	Stabilität von Algorithmen	46
3.3.13	Allgemeine Faustregeln für die LR-Zerlegung	52
3.4	Beurteilung von Näherungslösungen linearer GLS	53
4	Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)	55
4.1	Gaußsches Eliminationsverfahren mit Äquilibration und Nachiteration	55
4.1.1	Äquilibration der Zeilen	55
4.1.2	Äquilibration der Spalten	55
4.1.4	Nachiteration	56
4.2	Cholesky-Verfahren	56
4.2.3	Cholesky-Zerlegung	59
4.2.4	Rechenaufwand in flops	59

4.3	Lineare Ausgleichsprobleme	59
4.3.2	Lineares Ausgleichsproblem	61
4.3.5	Lösung der Normalgleichung	64
4.4	Orthogonalisierungsverfahren	66
4.4.1	Givens-QR-Algorithmus	69
4.4.3	Aufwand des Givens-QR-Algorithmus	70
4.4.5	Speicherung	73
4.4.6	Householder QR-Algorithmus	74
4.4.7	Berechnung von $Q^T b$	74
4.4.8	Aufwand für den Householder-QR-Algorithmus	75
5	Numerische Lösung nichtlinearer Gleichungssysteme	77
5.1	Einführung	77
5.1.2	Das Bisektionsverfahren	78
5.2	Fixpunktiteration	79
5.3	Konvergenzordnung und Fehlerabschätzungen	82
5.4	Newton-Verfahren für skalare Gleichung	85
5.4.1	Iterationsschritt des Newton(-Kantorowitsch)-Verfahrens	85
5.4.5	Newton-Verfahren: Iterativer Linearisierungsprozess	87
5.4.7	Iterationsschritt des Sekantenverfahrens	87
5.5	Das Newton-Verfahren im Mehrdimensionalen	89
5.5.1	Iterationsschritt des Newton-Verfahrens	89
5.5.2	Newton-Verfahren	89
5.5.4	Aufwand pro Iteration	90
5.6	Abbruchkriterien beim Newton-Verfahren	92
5.6.1	Der Monotonietest	92
5.6.2	Kriterium für erreichte Konvergenz	93
5.7	Varianten des Newton-Verfahrens	93
5.7.1	Iterationsschritt des vereinfachten Newton-Verfahrens	93
5.7.2	Das Broyden-Verfahren	93
5.7.3	Das gedämpfte Newton-Verfahren	94
6	Interpolation	97
6.1	Polynom-Interpolation	98
6.1.3	Schema von Neville	100
6.1.4	Das Horner-Schema zur Auswertung $p(x)$	100
6.1.8	Das Schema der dividierten Differenzen	102
6.2	Stückweise polynomiale Approximation durch Splines	111
6.2.3	Nachteile der Splineraumbasis	112
6.2.6	Gestalt der B-Splines	114
6.2.8	Auswertung von s an der Stelle \bar{x}	117
6.2.13	Splineinterpolation allgemein	119
6.2.14	Lineare B-Splines	119
6.2.15	Kubische B-Spline-Interpolation	120

6.2.18	Berechnung der natürlichen Splines mittels B-Splines	122
7	Numerische Integration/Quadratur	127
7.1	Einführung und einfache Quadraturformeln	127
7.1.2	Mittelpunktregel	128
7.1.3	Trapezsumme	129
7.2	Interpolatorische Integrationsformeln	129
7.2.3	Newton-Cotes-Formeln	130
7.2.4	Tabelle (Newton-Cotes-Gewichte)	131
7.2.9	Tabelle: Fehler für Newton-Cotes-Formeln	133
7.2.10	Iterierte (wiederholte) Newton-Cotes-Formeln	134
7.2.12	Iterierte (oder summierte) Simpsonregel (Keplersche Fassregel) . .	135
7.3	Extrapolationsmethode und klassische Rombert-Quadratur	135
7.3.2	Idee der Extrapolation	136
	Literatur	139

Vorwort

Skriptfehler

An alle, die gedenken dieses Skript zur Numerikvorlesung im WS2014/15 zu nutzen:
Es wird keinerlei Anspruch auf Richtigkeit, Vollständigkeit und auch sicher nicht Schönheit (ich bin \LaTeX -Anfänger) dieses Dokuments erhoben.

Ihr würdet mir aber unglaublich weiterhelfen, wenn ihr jede Anmerkung – das kann alles, von groben inhaltlichen Fehlern über Rechtschreibkorrekturen bis hin zu Wünschen/Anregungen/Tipps zur Typografie, sein – an mich weiterleitet!

Jegliche Anmerkungen bitte gleich und jederzeit an:

`gesina.schwalbe@stud.uni-regensburg.de`

Bilder oder „IMAGE MISSING“

Ich selber bin leider nur wenig mit (ordentlicher) Grafikerstellung in \LaTeX vertraut, aber dank Josef Wimmer stehen inzwischen die inhaltsrelevanten Abbildungen zur Verfügung – an dieser Stelle nochmal herzlichen Dank! Die noch fehlenden Grafiken sind mit „IMAGE MISSING“ markiert und wir sind dankbar über jede Mithilfe beim Füllen der Lücken: Ihr könnt ihr jederzeit **die entsprechenden Bilder an mich schicken!** Dann werden sie an die entsprechenden Stellen eingebunden bzw. digital abgezeichnet. Und gerade bei den **Funktionsplots** (z.B. zum Newton-Verfahren) würde ich mich sehr über Plot-Bilder freuen :-)

Erscheinungsdatum

Ich werde mich bemühen, das Skript jeweils am Vorlesungstag zumindest in unverbesserter Form online zu stellen, so dass v.a. diejenigen, die die Vorlesung nicht besuchen können, einen Überblick über den Stoff bekommen.

Innerhalb einer Woche sollte das Skript aktuell und korrekturgelesen sein.

Copyright

Was das Rechtliche angeht bitte beachten:

Urheber dieses Skriptes ist Prof. Dr. Luise Blank.

Dies ist nur eine genehmigte Vorlesungsmitschrift und unterliegt dem deutschen Urheberrecht, jegliche nicht rein private Verwendung muss demnach vorher mit Frau Blank abgesprochen werden.

Danksagung

Vielen Dank an

Kerstin Blumenhofer für die fleißige und ordentliche Mitschrift

(und natürlich auch allen anderen, die mir Notizen zur Verfügung gestellt haben)

Josef Wimmer für die Unterstützung bei der Grafikerstellung

Oliver Rümpelein für den Großteil meiner L^AT_EX-Kenntnisse und die tatkräftige Unterstützung bei allen Fragen zu allen Zeiten

1 Einführung

Wozu?

Oft sind Probleme mit der gleichen Struktur zu lösen, z.B.

- $ax^2 + bx + c = 0 \Rightarrow x_{\pm} = -\frac{b}{2a} \pm \frac{1}{2a} \sqrt{b^2 - 4ac}$
- Bestimmung des größten gemeinsamen Teilers zweier Zahlen
 \leadsto euklidischer Algorithmus

Hierfür ist ein allgemeiner Algorithmus erwünscht.

Ein weiterer Fall ist, dass ein Problem zwar analytisch gelöst werden kann, es aber zu lange dauert bis das Ergebnis bestimmt ist, z.B. tauchen bei der numerischen Simulation von Strömungen bis zu 1 Million Unbekannte auf. Damit sind Systeme mit $\approx 10^6$ Gleichungen zu lösen, wofür effiziente Algorithmen notwendig sind. Näherungslösungen sind bei solchen Problemen häufig ausreichend.

Es gibt auch Probleme, die nicht analytisch gelöst werden können, z.B. bei Differentialgleichungen ist vielleicht Existenz- und Eindeutigkeit gewährleistet, aber keine konstruktive Methode zur Berechnung bekannt. Hierfür ist dann eine Näherungslösung gefragt.

Geschichte

Algorithmen gibt es schon lange bevor es Rechner gab, z.B. den euklidischen Algorithmus seit um 300 v.Chr.

Die Fragestellungen und der Blickwinkel verschieben sich jedoch in Abhängigkeit von den zu lösenden Problemen und der existierenden Computer-Hardware. (Strömungsprobleme modelliert mit partiellen Differentialgleichungen, es stehen Parallelrechner, Vektorrechner, größere Speicher zur Verfügung, etc.)

Anwendungen

- Herd, Waschmaschine, Heizungsanlage
- Handy (über welchen Satelliten wird übertragen), Digitalkamera, MP3-Player
- Navigationssysteme
- Erstellung des Zugfahrplans
- Robotersteuerung (bis hin zu Roboterfußball)
- Fahrzeugindustrie
 - Fahrzeugbau (Crash-Simulation, Strömungsmodellierung)
 - Fahrzeugsteuerung
- Finanzmarkt z.B. Risikoanalyse im Wertpapierhandel
- Klimaaanalyse z.B. Vorhersage von Erdbeben, Hurrikans, Überflutungen
- Medizinische Versorgung z.B. Bildverarbeitung, Prognose für Epidemieentwicklungen, Beschreibung des Blutkreislaufs
- Raffinerie-Industrie
- Kontrolle und Optimierung chemischer und biologischer Prozesse, z.B. Regelung von Wärmezufuhr
- u.s.w.

Fragestellungen der Numerik

Rechengeschwindigkeit & -aufwand

Interessant sind Rechengeschwindigkeit, Rechenaufwand (Anzahl der Rechenoperationen, Rechenzeit auf welchem Rechner...) sowie Komplexität des Algorithmus.

Beispiel

Berechnung der Lösung eines Gleichungssystems

$Ax = b$ mit einer $n \times n$ -Matrix A

1. Cramersche Regel

$x_j = \frac{\det(A)_j}{\det A}$ (ersetze die j -te Spalte von A durch b)

und $\det A = \sum_{\pi} \text{sign}(\pi) a_{1m_1} \cdot \dots \cdot a_{nm_n}$

Benötigt etwa **$n!$ Multiplikationen und Additionen**. Bei einer 20×20 -Matrix A (was heutzutage klein ist) wären dies

$\approx 2,5 \cdot 10^{18}$ Operationen. Falls jede arithmetische Operation 10^{-6} Sekunden (also eine Mikrosekunde) benötigt, ist eine Rechenzeit von mehr als **eine Millionen Jahre** nötig!

2. Gaußsches Eliminationsverfahren

Benötigt etwa n^3 Operationen, also bei einer Matrix wie oben ungefähr 8000 Operationen und weniger als **0,005 Sekunden** [siehe auch GO96].

3. Verhalten bei Störungen, Stabilität des Verfahrens (Eingabefehler, Rundungsfehler, Diskretisierungsfehler)

Beispiel

$\frac{1}{10^{-8}} = 10^8$ Störung des Nenners $\frac{1}{2 \cdot 10^{-8}} = 5 \cdot 10^7$

\leadsto kleine Störung im Nenner kann zu großen Störungen im Ergebnis führen.

Beispiel

$$x^2 + 314x - 2 = 0$$

Falls diese Gleichung mit der **p,q-Formel** (Mitternachtsformel) gelöst wird und immer auf 5 signifikante Stellen gerundet wird, ergibt sich ein

$$\text{relativer Fehler} = \frac{|\text{Fehler}|}{|\text{Lösung}|}$$

von $\approx 57\%$.

Eine geschickte Formatierung liefert ein Ergebnis mit einem relativen Fehler von $\approx 1,5 \cdot 10^{-5}$, d.h. die ersten **4 Stellen sind exakt**. Dabei werden für $ax^2 + bx + c = 0$ folgende Ausdrücke verwendet:

$$\begin{aligned} x_1 &= \frac{1}{2a} (-b - \text{sign}(b) \sqrt{b^2 - 4ac}) \\ x_2 &= \frac{2c}{-b - \text{sign}(b) \sqrt{b^2 - 4ac}} \end{aligned}$$

Genauigkeit des Verfahrens, Fehleranalyse, Konvergenzgeschwindigkeit, Konvergenzordnung

Beispiel

Numerische Approximation von Ableitungen

Für $f \in C^3(I)$ gilt die Taylor-Entwicklung:

$$f(x \pm h) = f(x) \pm h f'(x) + \frac{h^2}{2} f''(x) + R(x) \text{ mit } |R(x)| \leq ch^3$$

Vorwärtsgenommener Differenzenquotient

$$(D_h^+ f)(x) := \frac{f(x+h) - f(x)}{h} \approx f'(x),$$

$$\left| f'(x) - \frac{f(x+h) - f(x)}{h} \right| \leq ch,$$

konvergiert also mit linearer Abhängigkeit der Schrittweite h

Zentraler Differenzenquotient

$$(D_h^0 f)(x) := \frac{f(x+h) - f(x-h)}{2 \cdot h} \approx f'(x),$$

$$\left| f'(x) - \frac{f(x+h) - f(x-h)}{2 \cdot h} \right| \leq ch^2,$$

konvergiert mit quadratischer **Ordnung** bei **gleichem Aufwand**!

Some disasters attributable to bad numerical computing

(Last modified August 26, 1998 by Douglas N. Arnold, arnold@ima.umn.edu)

Have you been paying attention in your numerical analysis or scientific computation courses?

If not, it could be a costly mistake.

Here are some real life examples of what can happen when numerical algorithms are not correctly applied.

The Patriot Missile failure in Dharan, Saudi Arabia, on February 25, 1991 which resulted in 28 deaths, is ultimately attributable to poor handling of **rounding errors**.

The explosion of the Ariane 5 rocket just after lift-off on its maiden voyage off French Guiana, on June 4, 1996, was ultimately the consequence of a **simple overflow**.

The sinking of the Sleipner A offshore platform in Gandsfjorden near Stavanger, Norway, on August 23, 1991, resulted in a loss of nearly one billion dollars. It was found to be the result of **inaccurate finite element analysis**.

Weitere praxisrelevante Fragestellungen

- Nutze black box solver oder entwickle Lösungsmethode, welche auf das spezielle Problem angepaßt ist.
- Wie teuer ist die Implementierung?
(= wieviel Arbeitszeit)
- Ist der implementierte Algorithmus vielseitig einsetzbar?
(welche Problemklassen deckt er ab, welche Rechnerstruktur ist vorausgesetzt)

„Gute“ Programme

- zuverlässig (fehlerfrei)
- robust (z.B. behandeln Ausnahmesituationen und filtern ungeeignete Daten heraus)
- portierbar auf andere Rechenanlagen
- wartungsfreundlich (leicht zu ändern oder zu erweitern)
- gut dokumentiert
- ausgiebig getestet
soll in den Übungen trainiert werden

Eine Faustregel der numerischen Mathematik

Zu jedem noch so eleganten numerischen Verfahren gibt es ein Gegenbeispiel, für welches die Methode völlig versagt.

(Teubner Taschenbuch)

Welche Probleme werden hier behandelt?

1. Lineare Gleichungssysteme $Ax = b$

- „kleine“ bis „mittelgroße“ Matrizen
→ **direkte Methoden:** nach endlich vielen Schritten ist die **exakte Lösung** bis auf Rundungsfehler berechnet (z.B. Gauß-Elimination)
- strukturierte Matrizen

$$\begin{pmatrix} * & * & * & & 0 \\ * & * & * & * & \\ & \ddots & & \ddots & \\ & & * & * & * \\ 0 & & & * & * \end{pmatrix}$$

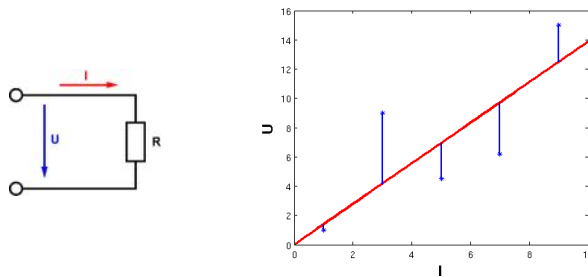
Symmetrie oder sogar Bandstruktur:

- große Matrizen (mit zusätzlichen Eigenschaften)
→ **iterative Methoden:** kenne Startwert x_0 , berechne neue Approximation x_i unter Ausnutzung der vorherigen bis die **Näherungslösung** x_i „gut genug ist“.

2. Lineare Ausgleichsprobleme

Beispiel:

Wir messen den Zusammenhang zwischen Spannung U und Stromstärke I



Ohmsches Gesetz: $U = R \cdot I$

Gesucht ist der Widerstand R .

(U_i, I_i) seien die Messdaten mit möglichen Messfehlern.

Finde nun R , sodass $f(R) = \min_r \sum_i (U_i - r I_i)^2$

3. Lösung nichtlinearer Gleichungen, z.B.

- Berechnung von Nullstellen $g(x) = 0$
- Berechnung von Fixpunkten $f(x) = x$

4. Eigenwertwertberechnung $Ax = \lambda x$, $\lambda \in \mathbb{C}$

5. Interpolation

Setze Meßdaten zu einer kontinuierlichen Funktion fort,
aber wie „glatt“?

z.B. stückweise konstant, stückweise linear, oder falls sie eine Schwingung repräsentieren, berechne die zugehörige Fourierreihe

6. Berechnung von Integralen (Quadraturformeln)

Approximation von $\int_a^b f(x) dx$

Bei allem spielt die **Fehleranalyse** eine große Rolle und ihre Grundbegriffe werden in einem extra Abschnitt behandelt.

2 Lineare Gleichungssysteme: Direkte Methoden

Sei $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$. Gesucht ist $x \in \mathbb{R}^n$ mit

$$A \cdot x = b$$

Weitere Voraussetzungen sind die Existenz und Eindeutigkeit einer Lösung.
Bemerkungen hierzu:

- Ein verlässlicher Lösungsalgorithmus überprüft dies und behandelt alle Fälle.
- Die Cramersche Regel ist ineffizient (s. Einführung).
- Das Inverse für $x = A^{-1} \cdot b$ aufzustellen ist ebenso ineffizient, denn es ist keine Lösung für alle $b \in \mathbb{R}^n$ verlangt und der Algorithmus wird evtl. instabil aufgrund vieler Operationen.

⇒ Invertieren von Matrizen vermeiden!!

⇒ Lösen des Linearen Gleichungssystems!!

2.1 Gaußsches Eliminationsverfahren

Das Verfahren wurde 1809 von Friedrich Gauß, 1759 von Joseph Louis Lagrange beschrieben und war seit dem 1. Jhd. v. Chr. in China bekannt.

2.1.1 Vorwärtselimination

Das Gaußverfahren gilt der Lösung eines linearen Gleichungssystems der Form

$$Ax = b$$

2 Lineare Gleichungssysteme: Direkte Methoden

mit $A = (a_{ij})_{i,j \leq n} \in K^{n \times n}$ Matrix und $b = (b_i)_{i \leq n} \in K^n$ Vektor.
Der zugehörige Algorithmus sieht folgendermaßen aus:

$$\begin{array}{cccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array}$$

\Downarrow

$$(i\text{-te Zeile}) - (1. \text{ Zeile}) \cdot \frac{a_{i1}}{a_{11}} \Rightarrow a_{i1} = 0$$

\Downarrow

$$\begin{array}{cccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ & & + & a_{22}^{(1)}x_2 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\ & & & \vdots & & & \vdots & & \vdots \\ & & & & & & a_{nn}^{(1)}x_n & = & b_n^{(1)} \end{array}$$

\Downarrow
 \vdots

mit

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij} - a_{1j} \cdot \frac{a_{i1}}{a_{11}} && \text{für } i, j = 2, \dots, n \\ b_i^{(1)} &= b_i - b_1 \cdot \frac{a_{i1}}{a_{11}} && \text{für } i = 2, \dots, n \end{aligned}$$

In jedem Schritt werden die Einträge der k -ten Spalte analog unterhalb der Diagonalen (also $k = 1, \dots, n-1$) eliminiert:

08.10.2014

$$(i\text{-te Zeile}) - (k\text{-te Zeile}) \cdot \frac{a_{ik}}{a_{kk}} \quad \text{für } i = k+1, \dots, n$$

Die Reihe

$$A \rightarrow A^{(1)} \rightarrow A^{(2)} \rightarrow \dots \rightarrow A^{(n-1)}$$

wird bis zum n -ten Schritt fortgeführt, d.h. bis eine obere Dreiecksgestalt eintritt:

$$\underbrace{\begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ & & \ddots & \vdots \\ 0 & & & a_{nn}^{(n-1)} \end{pmatrix}}_{:=R} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{R x = z} = \underbrace{\begin{pmatrix} b_1 \\ b_2^{(1)} \\ \vdots \\ b_n^{(n-1)} \end{pmatrix}}_{:=z} \quad (2.1.1)$$

wobei für $i = k + 1, \dots, n$ die Einträge wie folgt aussehen:

$$l_{ik} := \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \quad (2.1.2)$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{kj}^{(k-1)} \cdot l_{ik} \quad \text{für } j = k + 1, \dots, n \quad (2.1.3)$$

$$b_i^{(k)} = b_i^{(k-1)} - b_k^{(k-1)} \cdot l_{ik} \quad (2.1.4)$$

Dieser Prozess wird **Vorwärtselimination** genannt.

Der zugehörige Algorithmus ist:

```

for  $k = 1, \dots, n - 1$ 
|   for  $i = k + 1, \dots, n$ 
|   |    $l_{ik} = a_{ik} / a_{kk}$ 
|   |   for  $j = k + 1, \dots, n$ 
|   |   |    $a_{ij} = a_{ij} - l_{ik} a_{kj}$ 
|   |   end
|   |    $b_i = b_i - l_{ik} b_k$ 
|   end
end

```

2.1.2 Rückwärtselimination

Für die Lösung des Gleichungssystems ist dann noch die **Rückwärtssubstitution** nötig:

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}} \quad (2.1.5)$$

$$x_{n-1} = \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-1)} \cdot x_n}{a_{(n-1)(n-1)}^{(n-2)}} \quad (2.1.6)$$

$$x_k = \frac{b_k^{(k-1)} - \sum_{j=k+1}^n a_{kj}^{(k-1)} x_j}{a_{kk}^{(k-1)}} \quad (2.1.7)$$

Als Algorithmus:

```

for  $k = n, n-1, \dots, 1$ 
|    $x_k = b_k$ 
|   for  $j = k+1, \dots, n$ 
|   |    $x_k = x_k - a_{kj}x_j$ 
|   end
|    $x_k = x_k / a_{kk}$ 
end
    
```

Bemerkung 2.1.3. Algorithmen 2.1.1 und 2.1.2 sind nur ausführbar, falls für die sog. **Pivotelemente** $a_{kk}^{(k-1)}$ gilt:

$$a_{kk}^{(k-1)} \neq 0 \quad \text{für } k = 1, \dots, n$$

Dies ist auch für invertierbare Matrizen nicht immer gewährleistet.

2.1.4 Weitere algorithmische Anmerkungen

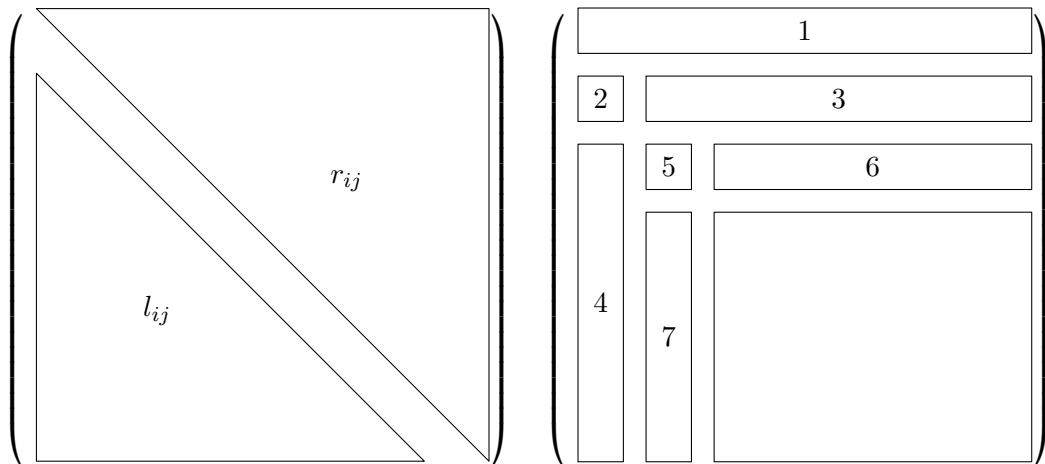
Matrix A und Vektor b sollten möglichst **nicht** überschrieben werden! (Stattdessen kann eine Kopie überschrieben werden.)

Das Aufstellen von A und b ist bei manchen Anwendungen das teuerste, sie gehen sonst verloren. In 2.1.1 wird das obere Dreieck von A überschrieben. Dies ist möglich, da in (2.1.3) nur die Zeilen $k+1, \dots, n$ mithilfe der k -ten bearbeitet werden. Am Ende steht R im oberen Dreieck von A und z in b .

Die l_{ik} werden spaltenweise berechnet und können daher anstelle der entsprechenden Nullen (in der Kopie) von A gespeichert werden, d.h.:

$$\tilde{L} := (l_{ik}) \tag{2.1.8}$$

und R werden sukzessive in A geschrieben.



Der Vektor z und anschließend der Lösungsvektor x kann in (eine Kopie von) b geschrieben werden. Wird eine neue rechte Seite b betrachtet, muss 2.1.1 nicht komplett neu ausgeführt werden, da sich \tilde{L} nicht ändert. Es reicht 2.1.4 zu wiederholen.

Definition 2.1.5. Die **Dreieckszerlegung** einer Matrix A entspricht dem Verfahren aus 2.1.1, nur ohne die Zeile (2.1.4).

Definition 2.1.6. Die **Vorwärtssubstitution** entspricht der in 2.1.4 bzw. dem Verfahren aus 2.1.1 ohne die Bestimmung von l_{ik} und R , also nur Schritt (2.1.4).

2.1.7 Algorithmus: Gauß-Elimination zur Lösung von $Ax = b$

1 Dreieckszerlegung

2 Vorwärtssubstitution $b_i^{(k)} = b_i^{(k-1)} - b_k^{(k-1)} \cdot l_{ik}$

3 Rückwärtssubstitution $x_k = \frac{b_k^{(k-1)} - \sum_{j=k+1}^n a_{kj}^{(k-1)} x_j}{a_{kk}^{(k-1)}}$

2.1.8 Rechenaufwand gezählt in „flops“

„flops“ = floating point operations

1. Dreieckszerlegung

für $j = k + 1, \dots, n$ 1 Addition, 1 Multiplikation für a_{ij}

für $i = k + 1, \dots, n$ 1 Division zusätzl. für l_{ik}

Dies ist je für $k = 1, \dots, n - 1$, also ist die Zahl an Additionen und Multiplikationen

$$\begin{aligned} \sum_{k=1}^{n-1} (n-k)^2 &= \sum_{k=1}^{n-1} k^2 \\ &= \frac{(n-1)n(2n-1)}{6} \\ &= \frac{2n^3 - 3n^2 + n}{6}. \end{aligned}$$

Für große n sind das etwa $\frac{n^3}{3}$ Additionen und Multiplikationen und

$$\sum_{k=1}^{n-1} (n-k) = \frac{n^2 - n}{2} \approx \frac{n^2}{n}$$

2 Lineare Gleichungssysteme: Direkte Methoden

Divisionen.

Damit ergibt sich eine Gesamtanzahl an flops von

$$2 \cdot \frac{2n^3 - 3n^2 + n}{6} + \frac{n^2 - n}{2} = \frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n \approx \frac{2}{3}n^2$$

für große n .

2. Vorwärts- bzw. Rückwärtssubstitution

Hier ergeben sich je

$$\sum_{k=1}^{n-1} (n-k) = \frac{n^2 - n}{2} \approx \frac{n^2}{2}$$

Multiplikationen und Additionen sowie n Divisionen für die Rückwärtssubstitution und damit insgesamt

$$n^2 + n$$

flops.

Zusammenfassung

Die Dreieckszerlegung benötigt $\mathcal{O}(n^3)$ flops und die Vorwärts- bzw. Rückwärtssubstitution $\mathcal{O}(n^2)$ flops.

Definition 2.1.9 (Landau-Symbole). Seien $f, g : D \rightarrow \mathbb{R}, D \subset \mathbb{R}, -\infty \leq a \leq \infty$ und $(a_n)_{n \in \mathbb{N}}, (b_n)_{n \in \mathbb{N}}$ Folgen in \mathbb{R} .

a) $f(x) = \mathcal{O}(g(x))$ für $x \rightarrow a$, falls

$$\exists U(a), c \in \mathbb{R} : \forall x \in U(a) : |f(x)| \leq c \cdot |g(x)|$$

$$(\text{bzw. falls } \lim_{x \rightarrow a} \frac{|f(x)|}{|g(x)|} \leq c)$$

b) $f(x) = o(g(x))$ für $x \rightarrow a$, falls $\lim_{x \rightarrow a} \frac{|f(x)|}{|g(x)|} = 0$

c) $a_n = \mathcal{O}(b_n)$ für $n \rightarrow \infty$, falls

$$\forall \varepsilon > 0 : \exists N \in \mathbb{N} : \forall n \geq N : |a_n| \leq \varepsilon |b_n|$$

2.1.10 Allgemeines zur Aufwandsbetrachtung

Die Anzahl der Rechenoperationen ist nicht immer ausschlaggebend für den Aufwand, da z.B.

Parallelrechner: In manchen Algorithmen sind Rechenschritte parallel ausführbar. Damit entspricht die Zeit nicht der Anzahl an Operationen und es wird zusätzlich „Kommunikationszeit“ benötigt.

Sortieralgorithmen: Die Indexverwaltung benötigt Zeit, aber keine/kaum Rechenoperationen

If-When-Abfragen: entsprechend

Rechenoperationen liefern jedoch oft eine gute Schätzung.

2.1.11 Formalisieren des Gauß-Algorithmus: LR-Zerlegung

13.10.2014

a) **Rückwärtssubstitution:** entspricht $Rx = b$

b) **Vorwärtssubstitution:**

$$b_i = b_i^{(k-1)} - l_{ik} b_k^{(k-1)} \quad i = k+1, \dots, n$$

$$b^{(k)} = b^{(k-1)} - l_k b_k^{(k-1)} \quad \text{mit } l_k := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ l_{k+1,k} \\ \vdots \\ l_{n,k} \end{pmatrix}$$

Sei $l_k \in \mathbb{R}^n$ der k-te Einheitsvektor und

$$L_k := I - l_k e_k^T = \begin{pmatrix} 1 & & & & \\ 0 & \ddots & & & 0 \\ & & 1 & & \\ \vdots & -l_{k+1,k} & & 1 & \\ & \vdots & & & \ddots \\ & -l_{n,k} & & & & 1 \end{pmatrix} \quad (2.1.9)$$

dann gilt $b^{(k)} = L_k b^{(k-1)}$, also

$$z = L_{n-1} \cdot L_{n-2} \cdots L_1 b$$

Sei

$$L := L_1^{-1} \cdots L_{n-1}^{-1} \quad (2.1.10)$$

Hiermit folgt dann, dass die Vorwärtssubstitution

$$Lz = b \quad (2.1.11)$$

entspricht.

2 Lineare Gleichungssysteme: Direkte Methoden

c) **Dreieckszerlegung:** Wie für die Vorwärtssubstitution ergibt sich

$$A^{(k)} = L_k A^{(k-1)}$$

und somit $R = L_{n-1} A^{(n-2)} = L_{n-1} \dots L_1 A$ bzw.

$$A = L \cdot R \quad (2.1.12)$$

Lemma 2.1.12.

1. L_k ist eine Frobeniusmatrix, d.h. sie unterscheidet sich höchstens in einer Spalte von der Einheitsmatrix I .
2. $L_k^{-1} = I + l_k e_k^T$
3. Es gilt:

$$\begin{aligned} L &= L_1^{-1} \cdot \dots \cdot L_{n-1}^{-1} \\ &= I + \sum_{i=1}^{n-1} l_i e_i^T \\ &= \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ l_{ij} & & 1 \end{pmatrix} \\ &= I + \tilde{L} \end{aligned} \quad (2.1.13)$$

Hiermit ergibt sich der folgende Satz:

Satz 2.1.13 (LR- oder LU-Zerlegung). *Das obige Verfahren ((2.1.2) und (2.1.3)) erzeugt unter der Voraussetzung von nicht-nullwertigen Pivotelementen eine Faktorisierung*

$$A = L \cdot R$$

wobei R eine obere Dreiecksmatrix und L eine untere, normierte Dreiecksmatrix ist, d.h. für $i = 1, \dots, n$ gilt $l_{ii} = 1$.

Weiterhin existiert zu jeder regulären Matrix höchstens eine solche Zerlegung.

Beweis. Zur Eindeutigkeit:

$$A = LR \Leftrightarrow a_{ij} = \sum_{k=1}^{\min j, k} l_{ik} r_{kj}$$

Da $l_{ii} = 1$ gilt, folgt

$$i \leq j: r_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} r_{kj} \text{ und}$$

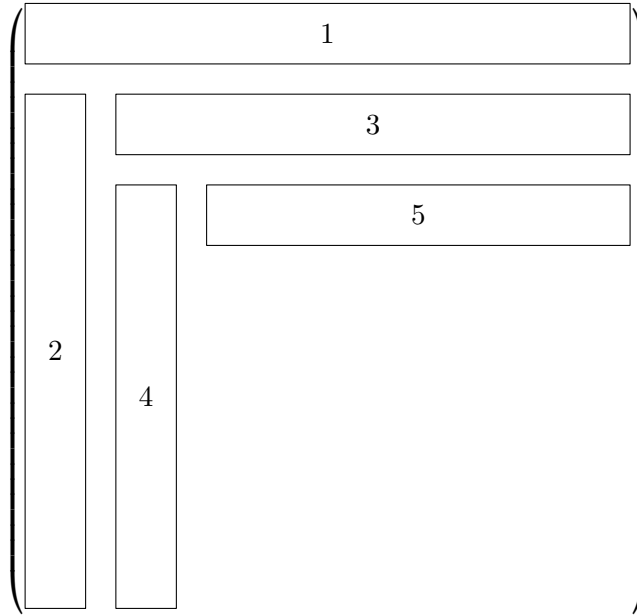
$$i > j: l_{ij} = \frac{1}{r_{ij}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} r_{kj} \right),$$

2.1 Gaußsches Eliminationsverfahren

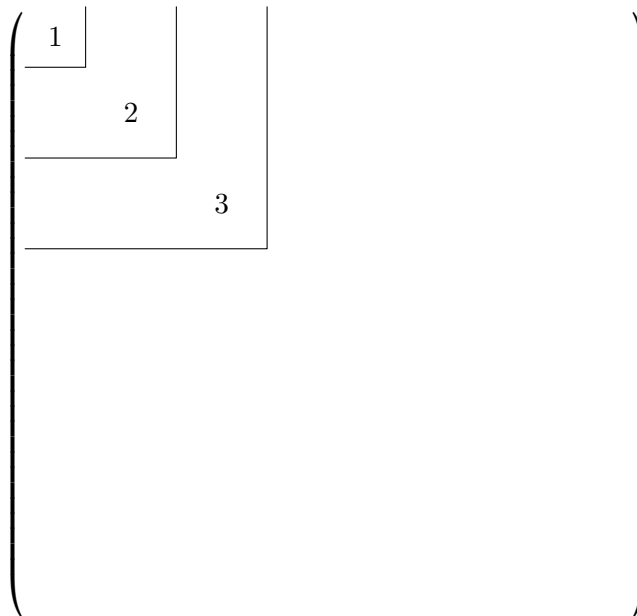
da mit A auch R regulär (=invertierbar) ist, und somit $\det(A) = \prod_{j=1}^n r_{jj} \neq 0$, also $r_{jj} \neq 0$ gilt.

Diese können rekursiv, also eindeutig, berechnet werden, wenn auch die Reihenfolge der Berechnung nicht eindeutig ist.

Mögliche Verfahren sind z.B. das **Verfahren von Crout**

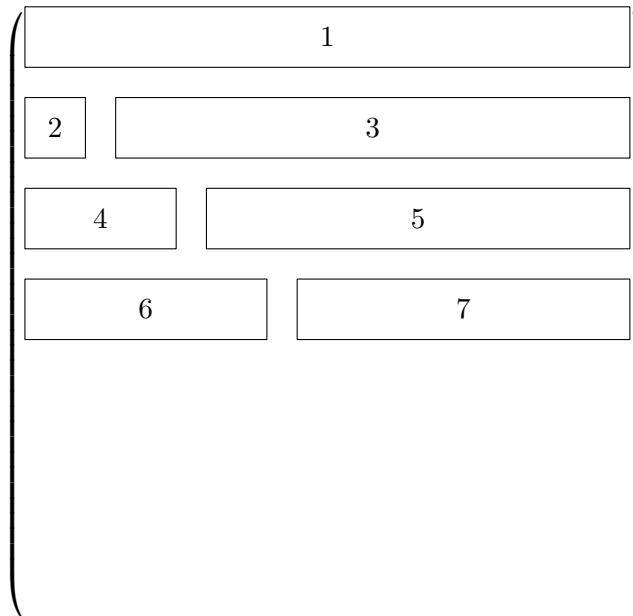


oder eckweise



2 Lineare Gleichungssysteme: Direkte Methoden

oder zeilenweise



oder wie in 2.1.1.

□

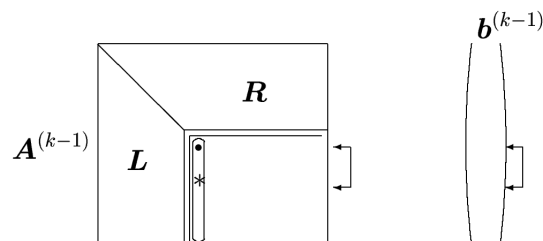
2.2 Gaußsches Eliminationsverfahren mit Pivotisierung

Beispiel Die Matrix $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ ist invertierbar, aber die Gauß-Elimination versagt. Permutiere also die erste mit der zweiten Zeile und der Algorithmus wird anwendbar.

Allgemein Vermeide die Division durch betragsmäßig kleine Zahlen!

2.2.1 Spaltenpivotisierung

Eine Spaltenpivotisierung, auch partielle oder halbmaximale Pivotisierung, erfolgt, indem im k -ten Eliminationsschritt durch Zeilenvertauschen das größte Spaltenelement das Pivotelement stellt:



- Bestimme das Pivotelement $a_{pk}^{(k-1)}$ als betragsmäßig größtes der „Rest-Spalte“, d.h.

$$|a_{pk}^{(k-1)}| \geq |a_{jk}^{(k-1)}| \quad \text{für } j = k, \dots, n$$

- Vertausche in $A^{(k-1)}$ die k -te mit der p -ten Zeile
- Führe einen Gauß-Eliminationsschritt aus.

Bemerkung 2.2.2.

- Hiermit gilt $|l_{jk}| \ll 1$.
- Anstelle von Spaltenpivotisierung kann eine **Zeilenpivotisierung** durchgeführt werden. Welche günstiger (in cpu-time) ist, hängt von der Rechnerarchitektur und der damit zusammenhängenden Umsetzung des Gauß-Algorithmus ab.
(Beispielsweise greifen Vektorrechner entweder auf die gesamte Spalte oder auf die gesamte Zeile einer Matrix zu und bevorzugen dementsprechend Operationen spalten- bzw. zeilenweise.)
- Der Aufwand enthält (bis auf $|\cdot|$) keine Rechenoperationen (flops), aber $\mathcal{O}(n^2)$ Vergleiche und Vertauschungen.
- Eine **vollständige Pivotsuche** sucht das betragsmäßig größte Element der gesamten Restmatrix und benötigt $\mathcal{O}(n^3)$ Vergleiche (sie wird so gut wie nie angewendet).

Damit die LR-Zerlegung unabhängig von der rechten Seite erstellt werden kann, müssen die Permutationen gespeichert werden. Hierfür verwendet man einen sog. **Permutationsvektor** Π , wobei

$$\Pi^{(k-1)}(r) = s$$

bedeutet, dass nach dem $(k-1)$ -ten Eliminationsschritt in der r -ten Zeile von $A^{(k-1)}$ die s -te bearbeitete Zeile von A steht, also

$$\Pi^{(k)}(k) = \Pi^{(k-1)}(p)$$

$$\Pi^{(k)}(p) = \Pi^{(k-1)}(k) \quad \text{und entsprechend}$$

$$\Pi^{(k)}(i) = \Pi^{(k)}(i) \quad \text{für } i \neq k, p$$

Für die **Permutationsmatrix**

$$P_{\Pi} = (e_{\Pi(1)}, \dots, e_{\Pi(n)}) \quad e_j := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow j\text{-te Stelle}$$

$$\text{mit } PA = LR$$

2 Lineare Gleichungssysteme: Direkte Methoden

gilt

$$P^{-1} = P^T$$

und

$$\det P_{\Pi} = \text{sign}(\Pi) = \begin{cases} +1 & \text{falls } \Pi \text{ von gerader} \\ -1 & \text{falls } \Pi \text{ von ungerader} \end{cases} \quad \text{Anzahl an Transpositionen erzeugt wird}$$

2.2.3 Algorithmus: Gauß-Elimination mit Spaltenpivotisierung

Der zugehörige Algorithmus zur Spaltenpivotisierung ist:

```

 $\pi(1:n) = [1:n]$ 
for  $k = 1, \dots, n-1$ 
|   bestimme Pivotzeile  $p$ , so dass
|        $|a_{pk}| = \max\{|a_{jk}|, j = k, \dots, n\}$ 
|    $\pi(k) \stackrel{1}{\leftrightarrow} \pi(p)$ 
|    $A(k, 1:n) \stackrel{1}{\leftrightarrow} A(p, 1:n)$ 
|   if  $a_{kk} \neq 0$ 
|       |    $zeile = [k+1:n]$ 
|       |    $A(zeile, k) = A(zeile, k)/a_{kk}$ 
|       |    $A(zeile, zeile) = A(zeile, zeile) - A(zeile, k)A(k, zeile)$ 
|   else
|       |   „ $A$  ist singulär“
|   end
end
```

Satz 2.2.4 (Dreieckszerlegung mit Permutationsmatrix). *Für jede invertierbare Matrix A existiert eine Permutationsmatrix P , so dass eine Dreieckszerlegung*

$$PA = LR$$

existiert. P kann so gewählt werden, dass alle Elemente von L betragsmäßig kleiner oder gleich 1 sind, d.h.

$$|l_{ij}| \leq 1 \quad \forall i, j$$

Beweis. Da $\det A \neq 0$ ist, existiert eine Transposition τ_1 , s.d.

$$a_{11}^{(1)} = a_{\tau_1, 1} \neq 0$$

¹ $\stackrel{1}{\leftrightarrow}$ bedeutet „vertausche mit“

und

$$|a_{\tau_1,1}| \geq |a_{i1}| \quad \forall i = 1, \dots, n.$$

Wir erhalten damit

$$L_1 P_{\tau_1} \cdot A = A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & \cdots \\ 0 & \\ \vdots & B^{(1)} \\ 0 & \end{pmatrix}$$

und alle Elemente von L_1 sind betragsmäßig kleiner oder gleich 1 sowie $\det L_1 = 1$.
Daraus folgt

$$\begin{aligned} \det B^{(1)} &= \underbrace{\frac{1}{a_{11}^{(1)}}}_{\neq 0} \cdot \det A^{(1)} \\ &= \frac{1}{a_{\tau_1,1}^{(1)}} \cdot \det(L_1) \cdot \det(A) \\ &\neq 0. \end{aligned}$$

Also ist $B^{(1)}$ invertierbar.

Induktiv erhalten wir dann

$$R = A^{(n-1)} = L_{n-1} P_{\tau_{n-1}} \cdot \dots \cdot L_1 P_{\tau_1} \cdot A$$

Da τ_i nur zwei Zahlen $\geq i$ vertauscht, ist

15.10.2014

$$\Pi_i := \tau_{n-1} \circ \dots \circ \tau_i \quad \text{für } i = 1, \dots, (n-1)$$

eine Permutation der Zahlen $\{i, \dots, n\}$, d.h. insbesondere gilt:

$$\begin{aligned} \Pi_i(j) &= j && \text{für } j = 1, \dots, (i-1) \\ \Pi_i(j) &\in \{i, \dots, n\} && \text{für } j = i, \dots, n. \end{aligned}$$

$$P_{\Pi_{i+1}} = (e_1, \dots, e_i, e_{\Pi_{i+1}(i+1)}, \dots, e_{\Pi_{i+1}(n)}) = \begin{pmatrix} I_i & 0 \\ 0 & P_\sigma \end{pmatrix}$$

2 Lineare Gleichungssysteme: Direkte Methoden

Damit folgt:

$$\begin{aligned}
 P_{\Pi_{i+1}} \cdot L_i \cdot P_{\Pi_{i+1}}^{-1} &= P_{\Pi_{i+1}} \cdot \left(\begin{array}{ccc|c} I_i & & & 0 \\ & -l_{i+1,i} & & \\ 0 & \vdots & & I_{n-i} \\ & -l_{n,i} & & \end{array} \right) \cdot \begin{pmatrix} I_i & 0 \\ 0 & P_\sigma^{-1} \end{pmatrix} \\
 &= \begin{pmatrix} I_i & 0 \\ 0 & P_\sigma \end{pmatrix} \cdot \frac{1}{\dots} \cdot \left(\begin{array}{ccc|c} I_i & & & 0 \\ & -l_{i+1,i} & & \\ & \vdots & & P_\sigma^{-1} \\ & -l_{n,i} & & \end{array} \right) \\
 &= \left(\begin{array}{ccc|c} I_i & & & 0 \\ & -l_{\Pi_{i+1}(i+1),i} & & \\ 0 & \vdots & & I_{n-i} \\ & -l_{\Pi_{i+1}(n),i} & & \end{array} \right) \\
 &= I - (P_{\Pi_{i+1}} l_i) e_i^T \\
 &=: \widehat{L}_i
 \end{aligned}$$

und

$$\begin{aligned}
 R &= L_{n-1} \\
 &\quad \cdot (P_{\tau_{n-1}} L_{n-2} P_{\tau_{n-1}}^{-1}) \\
 &\quad \cdot (P_{\tau_{n-1}} P_{\tau_{n-2}} L_{n-2} P_{\tau_{n-2}}^{-1} P_{\tau_{n-1}}^{-1}) \\
 &\quad \vdots \\
 &\quad \cdot (P_{\tau_{n-1}} \cdots P_{\tau_1} L_1 P_{\tau_1} \cdots P_{\tau_{n-1}}) \cdot A \\
 &= L_{n-1} \widehat{L}_{n-2} \cdots \widehat{L}_1 P_{\Pi_1} \cdot A
 \end{aligned}$$

Nach Lemma 2.1.12 gilt daher, es existiert eine Permutation Π_1 mit

$$P_{\Pi_1} \cdot A = LR,$$

wobei R obere Dreiecksgestalt hat und

$$L = \begin{pmatrix} 1 & & & 0 \\ l_{\Pi_2(2),1} & \ddots & & \\ \vdots & \ddots & 1 & \\ l_{\Pi_n(n),1} & \cdots & l_{\Pi_n(n),n-1} & 1 \end{pmatrix} \quad \text{mit } |l_{ij}| \leq 1$$

gilt. □

2.2.5 Lösen eines Gleichungssystems $Ax = b$

Das Lösen eines linearen Gleichungssystems der Form $Ax = b$ wird mittels Elimination durch folgende drei Schritte durchgeführt:

- 1) Zerlege A durch $PA = LR$
- 2) Löse durch Vorwärtssubstitution $Lz = Pb$
- 3) Löse durch Rückwärtssubstitution $Rx = z$

Bemerkung 2.2.6.

- a) $P_{\Pi}A = LR$ kann zur Berechnung von $\det(A)$ genutzt werden (allgemeine Formel: $\det(A) = \text{sign}(\Pi) \cdot r_{11} \cdot \dots \cdot r_{nn}$).
- b) Algorithmus 2.2.3 testet, ob die Matrix singulär ist, bis auf den Fall $r_{nn} = a_{nn}^{(n-1)} = 0$.
- c) $\det(A) = 0$ sollte nicht als (numerischer) Nachweis für die Singularität von A genutzt werden.
Z.B. ist $10^{-8}I$ regulär, aber $\det(10^{-8}I) = 10^{-8n}$ ist fast 0 für große n , also ist A numerisch singulär.

Beispiel 2.2.7. Wir betrachten die Pivotisierung mit betragsmäßig größtem Spaltenelement und Rundungsfehlern zu

$$A = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Die der Gauß-Elimination mit Rundung auf 3 Dezimalstellen ergibt $l_{21} = 10^4$, denn kleines Pivotelement bedeutet großer Multiplikator.

$$\begin{aligned} r_{22} &= a_{22} - l_{21}a_{12} = 1 - 10^4 \cdot 1 = -999 \approx -10^4 =: \tilde{r}_{22} \\ b_2^{(1)} &= b_2 - l_{21}b_1 = 2 - 10^4 = -9998 \approx -10^4 =: \tilde{b}_2^{(1)} \end{aligned}$$

Die Rückwärtssubstitution ergibt

$$\begin{aligned} x_2 &= \frac{-b_2^{(1)}}{r_{22}} = \frac{9998}{-999} \approx 1 \\ \tilde{x}_2 &= \frac{\tilde{b}_2}{\tilde{r}_{22}} = \frac{-10^4}{-10^4} = 1 \end{aligned}$$

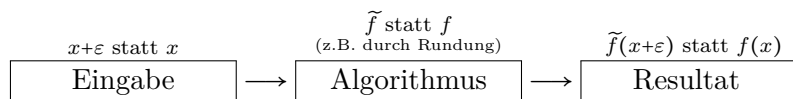
$$\begin{aligned} x_1 &= \frac{1}{r_{11}}(b_1 - r_{12}x_2) = 10^4(1 - 1x_2) = \frac{10^4}{9999} \approx 1 \\ \tilde{x}_1 &= \frac{1}{\tilde{r}_{11}}(1 - 1\tilde{x}_2) = 10^4(1 - 1 \cdot 1) = 0 \end{aligned}$$

Dies führt zu einem starken Fehler.

Mit Spaltenpivotisierung ist $l_{21} = 10^{-4} < 1$ und $\tilde{r}_{22} = 1$, $b_2^{(2)} = 1$, $\tilde{x}_2 = 1$, $\tilde{x}_1 = 1$. Diese Werte führen auch bei Rundungsfehlern zu besseren Ergebnissen.

3 Fehleranalyse

15.10.2014



Bei der Fehleranalyse liegt das Hauptaugenmerk auf

Eingabefehler

z.B. Rundungsfehler, Fehler in Messdaten, Fehler im Modell (falsche Parameter)

Fehler im Algorithmus

z.B. Rundungsfehler durch Rechenoperationen, Approximationen

(z.B. Ableitung durch Differenzenquotient oder die Berechnung von Sinus durch abgebrochene Reihenentwicklung)

1. *Frage* Wie wirken sich Eingabefehler auf das Resultat unabhängig vom gewählten Algorithmus aus?
2. *Frage* Wie wirken sich (Rundungs-)Fehler des Algorithmus aus?
Und wie verstärkt der Algorithmus Eingabefehler?

3.1 Zahlendarstellung und Rundungsfehler

Auf (Digital-)Rechnern können nur endlich viele Zahlen realisiert werden.

Die wichtigsten Typen sind:

- **ganze Zahlen** (integer):

$$z = \pm \sum_{i=0}^m z_i \beta_i \quad \text{mit} \quad \begin{array}{l} \beta = \text{Basis des Zahlensystems (oft } \beta = 2) \\ z_i \in \{0, \dots, \beta - 1\} \end{array}$$

- **Gleitpunktzahlen** (floating point)

3 Fehleranalyse

Definition 3.1.1. Eine Zahl $x \in \mathbb{Q}$ mit einer Darstellung

$$x = \sigma \cdot (a_1.a_2 \dots a_t)_\beta \cdot \beta^e = \sigma \beta^e \cdot \sum_{\nu=1}^t a_\nu \beta^{-\nu+1}$$

$\beta \in \mathbb{N}$	Basis des Zahlensystems
$\sigma \in \{\pm 1\}$	Vorzeichen
$m = (a_1.a_2 \dots a_t)_\beta$ $= \sum_{\nu=1}^t a_\nu \beta^{-\nu+1}$	Mantisse
$a_i \in \{0, \dots, \beta - 1\}$	Ziffern der Mantisse
$t \in \mathbb{N}$	Mantissenlänge
$e \in \mathbb{Z}$	mit $e_{\min} \leq e \leq e_{\max}$ Exponent

heißt **Gleitkommazahl** mit t Stellen und Exponent e zur Basis b .
Ist $a_1 \neq 0$, so heißt x **normalisierte Gleitkommazahl**.

Bemerkung 3.1.2.

- a) 0 ist keine normalisierte Gleitkommazahl, da $a_1 = 0$ ist.
- b) $a_1 \neq 0$ stellt sicher, dass die Gleitkommadarstellung eindeutig ist.
- c) In der Praxis werden auch nicht-normalisierte Darstellungen verwendet.
- d) Heutige Rechner verwenden meist $\beta = 2$, aber auch $\beta = 8, \beta = 16$.

3.1.3 Bit-Darstellung zur Basis 2

Bit-Darstellung nach IEEE-Standard 754 von floating point numbers
Sei die Basis $\beta = 2$.

	Speicherplatz	t	e_{\min}	e_{\max}
einfache Genauigkeit (float)	32bits = 4Bytes	24	-126	127
doppelte Genauigkeit (double)	64bits = 8Bytes	52	-1022	1023

Darstellung im Rechner (Bitmuster) für float:

s	$b_0 \dots b_7$	$a_2 \dots a_{24}$
-----	-----------------	--------------------

(Da $a_1 \neq 0$, also $a_1 = 1$ gilt, wird a_1 nicht gespeichert)

Interpretation ($s, b, a_i \in \{0, 1\} \forall i$)

- s Vorzeichenbit: $\sigma = (-1)^s \Rightarrow \begin{matrix} \sigma(0) = 1 \\ \sigma(1) = -1 \end{matrix}$

- $b = \sum_{i=0}^7 b_i \cdot 2^i \in \{1, \dots, 254\}$ speichert den Exponenten mit
 $e = b - \underbrace{127}_{\text{Basiswert}}$ (kein Vorzeichen nötig)

Beachte: $b_0 = \dots = b_7 = 1$ sowie $b_0 = \dots = b_7 = 0$ sind bis auf Ausnahmen keine gültigen Exponenten

- $m = (a_1.a_2 \dots a_{24}) = 1 + \sum_{\nu=2}^{24} a_\nu 2^{1-\nu}$ stellt die Mantisse dar, $a_1 = 1$ wird nicht abgespeichert.
- Besondere Zahlen per Konvention:

$$x = 0: s \text{ bel.}, b = 0, m = 1 \quad \boxed{s \mid 0 \dots 0 \mid 0 \dots 0}$$

$$x = \pm\infty: s \text{ bel.}, b = 255, m = 1 \quad \boxed{s \mid 1 \dots 1 \mid 0 \dots 0}$$

$$x = \text{NaN} \quad s \text{ bel.}, b = 255, m \neq 1$$

$$x = (-1)^s \quad s \text{ bel.}, b = 0, m \neq 1 \text{ und } x \text{ hat die Form } x = (0 + \sum_{\nu=2}^{24} a_\nu \cdot 2^{1-\nu}) \cdot 2^{126} \text{ („denormalized“ number)}$$

20.10.2014

Betragsmäßig **größte Zahl**:

$$\boxed{0 \mid 01 \dots 1 \mid 1 \dots 1}$$

$$x_{\max} = (2 - 2^{-23}) \cdot 2^{127} \approx 3,4 \cdot 10^{38}$$

Betragsmäßig **kleinste Zahl**:

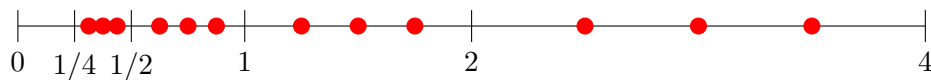
$$\boxed{0 \mid 0 \dots 0 \mid 0 \dots 01}$$

$$x_{\min} = (2 - 2^{-23}) \cdot 2^{-126} = 2^{-149} \approx 1,4 \cdot 10^{-45}$$

3.1.4 Verteilung der Maschinenzahlen

Die Maschinenzahlen sind ungleichmäßig im Dezimalsystem verteilt, z. B.

$$x = \pm a_1.a_2a_3 \cdot 2^e \quad \text{mit } -2 \leq e \leq 1 \text{ und } a_i \in \{0, 1\}$$



ist im Dualsystem gleichmäßig, jedoch im Dezimalsystem sehr ungleichmäßig verteilt.

Definition 3.1.5.

overflow es ergibt sich eine Zahl, die betragsmäßig größer ist als die größte maschinendargestellbare Zahl

underflow entsprechend, betragsmäßig kleiner als die kleinste positive Zahl

3 Fehleranalyse

Bsp.: overflow beim integer $b = e + 127$

$$\begin{array}{rcl} b & = & 254 \quad 11111110 \\ & + & 3 \quad 00000011 \\ b + 3 = 257 \bmod 2^8 & = & 1 \quad \cancel{1}00000001 \end{array}$$

3.1.6 Rundungsfehler

Habe $x \in \mathbb{R}$ die normalisierte Darstellung

$$\begin{aligned} x &= \sigma \cdot \beta^e \left(\sum_{\nu=1}^t a_{\nu} \beta^{1-\nu} + \sum_{\nu=t+1}^{\infty} a_{\nu} \beta^{1-\nu} \right) \\ &= \sigma \cdot \beta^e \left(\sum_{\nu=1}^t a_{\nu} \beta^{1-\nu} + \beta^{1-t} \sum_{l=1}^{\infty} a_{t+l} \beta^{-l} \right) \end{aligned}$$

mit $e_{\min} \leq e \leq e_{\max}$, dann wird mit $fl(x)$ die gerundete Zahl bezeichnet, wobei $fl(x)$ eindeutig gegeben ist durch die Schranke an den **absoluten Rundungsfehler**

$$|fl(x) - x| \leq \begin{cases} \frac{1}{2} \beta^{e+1+t} & \text{bei symmetrischem Runden} \\ \beta^{e+1+t} & \text{bei Abschneiden} \end{cases}.$$

Für die **relative Rechengenauigkeit** folgt somit

$$\frac{|fl(x) - x|}{|x|} \leq \begin{cases} \frac{1}{2} \beta^{1-t} & \text{bei symmetrischem Runden} \\ \beta^{1-t} & \text{bei Abschneiden} \end{cases}.$$

Die **Maschinengenauigkeit** des Rechners ist daher durch

$$eps = \beta^{1-t} \quad (\text{für float} \approx 10^{-7}, \text{ für double} \approx 10^{-16})$$

gegeben.

Die Mantissenlänge bestimmt also die Maschinengenauigkeit. Bei einfacher Genauigkeit ist $fl(x)$ bis auf ungefähr 7 signifikante Stellen genau.

Im Folgenden betrachten wir symmetrisches Runden und definieren daher

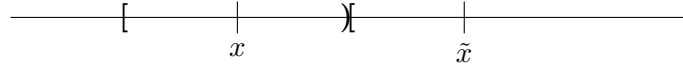
$$\tau := \frac{1}{2} eps$$

Weiterhin gilt:

a) Die kleinste Zahl am Rechner, welche größer als 1 ist, ist

$$1 + eps$$

b) Eine Maschinenzahl x repräsentiert eine Eingabemenge

$$E(x) = \{\tilde{x} \in \mathbb{R} : |\tilde{x} - x| \leq \tau|x|\}$$


Bemerkung 3.1.7. Gesetze der arithmetischen Operationen gelten i.A. nicht, z.B.

- x Maschinenzahl $\Rightarrow fl(x + \nu) = x$ für $|\nu| < \tau|x|$
- Assoziativ- und Distributivgesetze gelten nicht, z.B. für $\beta = 10$, $t = 3$, $a = 0,1$, $b = 105$, $c = -104$ gilt:

$$\begin{aligned} fl(a + fl(b + c)) &= 1,1 \\ fl(fl(a + b) + c) &= fl(fl(105,1) + (-104)) \\ &= fl(105 - 104) \\ &= 1 \quad \text{!} \end{aligned}$$

\Rightarrow Für einen Algorithmus ist die Reihenfolge der Operationen wesentlich! Mathematisch äquivalente Formulierungen können zu verschiedenen Ergebnissen führen.

3.1.8 Auslöschung von signifikanten Stellen

Sei $x = 9,995 \cdot 10^{-1}$, $y = 9,984 \cdot 10^{-1}$. Runde auf drei signifikante Stellen und berechne $x - y$:

$$\begin{aligned} \tilde{f}(x, y) &:= fl(fl(x) - fl(y)) = fl(1,00 \cdot 10^0 - 9,98 \cdot 10^{-1}) \\ &= fl(0,02 \cdot 10^{-1}) \\ &= fl(2,00 \cdot 10^{-3}) \\ f(x, y) &:= x - y \\ &:= 0,0011 = 1,1 \cdot 10^{-3} \end{aligned}$$

Daraus ergibt sich der relative Fehler

$$\frac{|\tilde{f}(x, y) - f(x, y)|}{|f(x, y)|} = \frac{|2 \cdot 10^{-3} - 1,1 \cdot 10^{-3}|}{|1,1 \cdot 10^{-3}|} = 82\%$$

Der Grund hierfür ist, dass das Problem der Subtraktion zweier annähernd gleich großer Zahlen schlecht konditioniert ist.

Zwei Regeln:

3 Fehleranalyse

- 1) Umgehbare Subtraktion annähernd gleich großer Zahlen vermeiden!
- 2) Unumgängliche Subtraktion möglichst an den Anfang des Algorithmus stellen! (siehe später)

3.2 Kondition eines Problems

Es wird das Verhältnis

$$\frac{\text{Ausgabefehler}}{\text{Eingabefehler}}$$

untersucht.

Definition 3.2.1. Sei $f : U \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m$ mit U offen und sei $x \in U$. Dann bezeichne (f, x) das Problem, zu einem gegebenen x die Lösung $f(x)$ zu finden.

Definition 3.2.2. Sei $x \in \mathbb{R}^n$ und $\tilde{x} \in \mathbb{R}^n$ eine Näherung an x . Weiterhin sei $\|\cdot\|$ eine Norm auf \mathbb{R}^n .

a) $\|\tilde{x} - x\|$ heißt **absoluter Fehler**

b) $\frac{\|\tilde{x} - x\|}{\|x\|}$ heißt **relativer Fehler**

Da der relative Fehler skalierungsinvariant ist, d.h. unabhängig von der Wahl von x ist, ist dieser i.d.R. von größerem Interesse. Beide Fehler hängen von der Wahl der Norm ab! Häufig werden Fehler auch komponentenweise gemessen:

$$\begin{array}{lll} \text{Für } i = 1, \dots, n : & |\tilde{x}_i - x_i| \leq \delta & (\text{absolut}) \\ & |\tilde{x}_i - x_i| \leq \delta |x_i| & (\text{relativ}) \end{array}$$

Wiederholung 3.2.3 (Normen).

$$\text{Summennorm } (l_1\text{-Norm}): \quad \|x\|_1 := \sum_{i=1}^n |x_i|$$

$$\text{Euklidische Norm } (l_2\text{-Norm}): \quad \|x\|_2 := \sqrt{\sum_{i=1}^n |x_i|^2}$$

$$\text{Maximumsnorm } (l_\infty\text{-Norm}): \quad \|x\|_\infty := \max\{|x_i| : i = 1, \dots, n\}$$

$$\text{Hölder-Norm } (l_p\text{-Norm}): \quad \|x\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

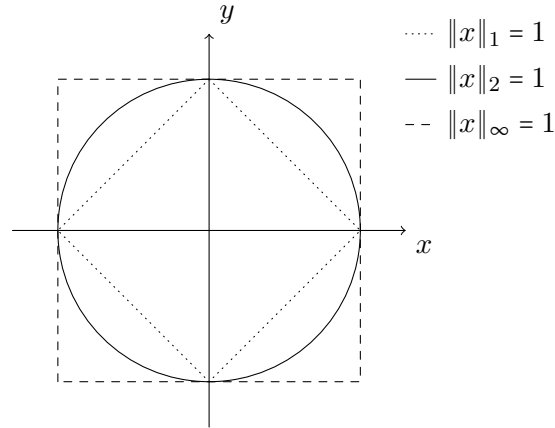


Abbildung 3.1: Sphären mit gleichem Normbetrag

Definition 3.2.4. Auf dem \mathbb{R}^n sei die Norm $\|\cdot\|_a$ und auf dem \mathbb{R}^m die Norm $\|\cdot\|_b$ gegeben. Dann ist die zugehörige **Matrixnorm** gegeben durch:

$$\begin{aligned} \|A\|_{a,b} &:= \sup_{x \neq 0} \frac{\|Ax\|_b}{\|x\|_a} \\ &= \sup_{\|x\|_a=1} \|Ax\|_b \end{aligned} \quad (3.2.1)$$

Also ist $\|A\|_{a,b}$ die kleinste Zahl $c > 0$ mit

$$\|Ax\|_b \leq c \|x\|_a \quad \forall x \in \mathbb{R}^n$$

Definition 3.2.5. Sei $A \in \mathbb{R}^{m \times n}$.

a) **Frobeniusnorm** (Schurnorm): $\|A\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}^2|}$

b) **p-Norm**: $\|A\|_p := \|A\|_{p,p}$

c) Eine Matrixnorm heißt **verträglich** mit den Vektornormen $\|\cdot\|_a, \|\cdot\|_b$, falls gilt ¹:

$$\|Ax\|_b \leq \|A\| \cdot \|x\|_a \quad \forall x \in \mathbb{R}^n$$

Bemerkung 3.2.6.

a) Die Normen $\|\cdot\|_F$ und $\|\cdot\|_p$ sind **submultiplikativ**, d.h.

$$\|A \cdot B\| \leq \|A\| \cdot \|B\|$$

¹ Beachte: $\|A\|_{a,b}$ ist die kleinste Norm im Gegensatz zu $\|A\|$, welche hier beliebig ist.

3 Fehleranalyse

b) Die Norm $\|\cdot\|_{1,1}$ wird auch **Spaltensummennorm** genannt:

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|$$

Sie ist das Maximum der Spaltensummen².

c) Die Norm $\|\cdot\|_{\infty,\infty}$ wird auch **Zeilensummennorm** genannt³:

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|$$

d) Die Frobeniusnorm $\|\cdot\|_F$ ist verträglich mit der euklidischen Norm $\|\cdot\|_2$

e) Die Wurzeln aus den Eigenwerten von $A^T A$ heißen **Singulärwerte** σ_i von A. Mit ihnen kann die $\|\cdot\|_{2,2}$ Norm dargestellt werden⁴:

$$\begin{aligned}\|A\|_2 &= \max \{ \sqrt{\mu} : A^T A \cdot x = \mu x \text{ für ein } x \neq 0 \} \\ &= \sigma_{\max}\end{aligned}$$

3.2a) Normweise Konditionsanalyse

22.10.2014

Definition 3.2.7. Sei (f, x) ein Problem mit $f : U \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ und $\|\cdot\|_a$ auf \mathbb{R}^n und $\|\cdot\|_b$ auf \mathbb{R}^m eine Norm.

a) Die **absolute normweise Kondition** eines Problems (f, x) ist die kleinste Zahl $\kappa_{abs} > 0$ mit

$$\begin{aligned}\|f(\tilde{x}) - f(x)\|_b &\leq \kappa_{abs}(f, x) \|\tilde{x} - x\|_a + o(\|\tilde{x} - x\|_a) \\ \left(f(\tilde{x}) - f(x) = \underbrace{f'(x)(\tilde{x} - x)}_{\text{Taylorentwicklung}} \pm o(\|\tilde{x} - x\|) \right) &\quad \text{für } \tilde{x} \rightarrow x\end{aligned}\tag{3.2.2}$$

b) Die **relative normweise Kondition** eines Problems (f, x) mit $x \neq 0, f(x) \neq 0$ ist die kleinste Zahl $\kappa_{rel} > 0$ mit

$$\frac{\|f(\tilde{x}) - f(x)\|_b}{\|f(x)\|_b} \leq \kappa_{rel}(f, x) \frac{\|\tilde{x} - x\|_a}{\|x\|_a} + o\left(\frac{\|\tilde{x} - x\|_a}{\|x\|_a}\right) \quad \text{für } \tilde{x} \rightarrow x\tag{3.2.3}$$

c) Sprechweise:

²Beweis: siehe Übungsblatt 3

³Beweis: siehe Übungsblatt 3

⁴Beweis: siehe Übungsblatt 3

- falls κ „klein“ ist, ist das Problem „gut konditioniert“
- falls κ „groß“ ist, ist das Problem „schlecht konditioniert“

Lemma 3.2.8. Falls f differenzierbar ist, gilt

$$\kappa_{abs}(f, x) = \|Df(x)\|_{a,b} \quad (3.2.4)$$

und für $f(x) \neq 0$

$$\kappa_{rel}(f, x) = \frac{\|x\|_a}{\|f(x)\|_b} \cdot \|Df(x)\|_{a,b} \quad (3.2.5)$$

wobei $Df(x)$ die Jakobi-Matrix bezeichnet.

Beispiel 3.2.9 (Kondition der Addition). $f(x_1, x_2) := x_1 + x_2$, $f: \mathbb{R}^2 \rightarrow \mathbb{R}$.
Wähle l_1 -Norm auf \mathbb{R}^2 (und \mathbb{R})

$$\begin{aligned} Df(x_1, x_2) &= (\nabla f^T) = \left(\frac{\partial}{\partial x_1} f, \frac{\partial}{\partial x_2} f \right) \\ &= (1, 1) \end{aligned} \quad (\text{Matrix!})$$

damit

$$\begin{aligned} \kappa_{abs}(f, x) &= \|Df(x)\|_{1,1} && (\text{Matrix-Norm!!}) \\ &= \|Df(x)\|_1 \\ &= 1 \\ \kappa_{rel}(f, x) &= \frac{\|x\|_1}{\|f(x)\|_1} \cdot \|Df(x)\|_1 \\ &= \frac{|x_1| + |x_2|}{|x_1 + x_2|} \end{aligned}$$

Daraus folgt: Die Addition zweier Zahlen mit gleichem Vorzeichen ergibt

$$\kappa_{rel} = 1$$

Die Subtraktion zweier annähernd gleich großer Zahlen ergibt eine sehr schlechte relative Konditionierung:

$$\kappa_{rel} \gg 1$$

Zum Beispiel in 3.1.8: Es ist

$$\begin{aligned} x &= \begin{pmatrix} 9,995 \\ -9,984 \end{pmatrix} \cdot 10^{-1} \\ \tilde{x} = fl(x) &= \begin{pmatrix} 1 \\ -9,98 \cdot 10^{-1} \end{pmatrix} \end{aligned}$$

3 Fehleranalyse

also

$$\begin{aligned}\frac{|f(\tilde{x}) - f(x)|}{|f(x)|} &= \frac{0,9}{1,1} = 0,81 \\ &\leq \kappa_{rel}(f, x) \cdot \frac{\|\tilde{x} - x\|_1}{\|x\|_1} \\ &= \kappa_{rel}(f, x) \cdot 4,6 \cdot 10^{-4}\end{aligned}$$

Beispiel 3.2.10 (Lösen eines lin. Gleichungssystems). Sei $A \in \mathbb{R}^{n \times n}$ invertierbar und $b \in \mathbb{R}^n$. Es soll

$$Ax = b$$

gelöst werden. Die möglichen Lösungen in A und in b lassen sich folgendermaßen ermitteln:

a) Betrachte die Störungen in b :

Sei hierzu

$$f : b \mapsto x = A^{-1}b$$

Berechne dann $\kappa(f, b)$ und löse

$$\begin{aligned}A(x + \Delta x) &= b + \Delta b \\ f(b + \Delta b) - f(b) &= \Delta x \\ &= A^{-1} \cdot \Delta b && \text{da } x = A^{-1}b \\ \Rightarrow \|\Delta x\|_b &= \|A^{-1} \Delta b\|_b \\ &\leq \|A^{-1}\|_{a,b} \cdot \|\Delta b\|_b && \forall b, \Delta b\end{aligned}$$

wobei $\|\cdot\|$ auf $\mathbb{R}^{n \times n}$ die dem \mathbb{R}^n zugeordnete Matrix-Norm sei.

Die Abschätzung ist **scharf**, d.h. es gibt ein $\Delta b \in \mathbb{R}^n$, so dass „=“ gilt, nach Definition 3.2.4.

Also gilt⁵:

$$\kappa_{abs}(f, b) = \|A^{-1}\|_{a,b} \tag{3.2.6}$$

unabhängig von b . ($x \mapsto Ax \quad \kappa_{abs}$)

Ebenso folgt die scharfe Abschätzung

$$\begin{aligned}\frac{\|f(b + \Delta b) - f(b)\|}{\|f(b)\|} &= \frac{\|\Delta x\|}{\|x\|} \\ &= \frac{\|A^{-1} \Delta b\|}{\|x\|} \\ &\leq \frac{\|A^{-1}\| \cdot \|b\|}{\|x\|} \cdot \frac{\|\Delta b\|}{\|b\|}\end{aligned}$$

⁵vgl. auch Lemma 3.2.8: $\kappa_{abs}(f, b) = \|Df(b)\|_{a,b} = \|A^{-1}\|_{a,b}$

Damit

$$\kappa_{rel}(f, b) = \|A^{-1}\| \cdot \frac{\|b\|}{\|A^{-1} \cdot b\|} \quad (3.2.7)$$

Da $\|b\| \leq \|A\| \cdot \|x\| = \|A\| \cdot \|A^{-1}b\|$ folgt:

$$\kappa_{rel}(f, b) \leq \|A\| \cdot \|A^{-1}\| \quad (3.2.8)$$

für alle (möglichen rechten Seiten) b .

3.2.8 ist scharf in dem Sinne, dass es ein $\widehat{b} \in \mathbb{R}^n$ gibt mit

$$\|\widehat{b}\| = \|A\| \cdot \|\widehat{x}\|$$

und somit

$$\kappa_{rel}(f, \widehat{b}) = \|A\| \cdot \|A^{-1}\|$$

b) Betrachte die Störungen in A :

Löse also

$$(A + \Delta A)(x + \Delta x) = b$$

Sei hierzu

$$\begin{aligned} f : A &\mapsto x = A^{-1}b \\ \mathbb{R}^{n \times n} &\rightarrow \mathbb{R}^n \end{aligned}$$

und berechne $\kappa(f, A)$ mittels Ableitung $Df(A) : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^n$:

$$\begin{aligned} C \mapsto Df(A)C &= \frac{d}{dt} \left((A + tC)^{-1} \cdot b \right) \Big|_{t=0} \\ &= \frac{d}{dt} \left((A + tC)^{-1} \right) \Big|_{t=0} \cdot b \end{aligned}$$

Weiterhin gilt

$$\frac{d}{dt} \left((A + tC)^{-1} \right) \Big|_{t=0} = -A^{-1}CA^{-1}, \quad (3.2.9)$$

da

$$\begin{aligned} 0 &= \frac{d}{dt} I \\ &= \frac{d}{dt} \left((A + tC)(A + tC)^{-1} \right) \\ &= C(A + tC)^{-1} + (A + tC) \cdot \frac{d}{dt} (A + tC)^{-1} \\ \Leftrightarrow \frac{d}{dt} (A + tC)^{-1} &= -(A + tC)^{-1} \cdot C(A + tC)^{-1}, \end{aligned}$$

3 Fehleranalyse

falls $(A + tC)$ invertierbar ist. Für ein genügend kleines t ist das gewährleistet, da A invertierbar ist (s. Lemma 3.2.12).

$$\Rightarrow Df(A)C = -A^{-1}CA^{-1}b$$

Somit folgt

$$\begin{aligned}\kappa_{abs}(f, A) &= \|Df(A)\| \\ &= \sup_{\substack{C \neq 0 \\ C \in \mathbb{R}^{n \times n}}} \frac{\|A^{-1}CA^{-1}b\|}{\|C\|} \\ &\leq \sup_{\substack{C \neq 0 \\ C \in \mathbb{R}^{n \times n}}} \frac{\|A^{-1}\| \cdot \|C\| \cdot \|A^{-1}b\|}{\|C\|} \\ &= \|A^{-1}\| \cdot \|x\| \\ &\leq \|A^{-1}\|^2 \cdot \|b\| \\ \kappa_{rel}(f, A) &= \frac{\|A\|}{\|f(A)\|} \cdot \|Df(A)\| \\ &\leq \|A\| \cdot \|A^{-1}\|\end{aligned}\tag{3.2.10}$$

c) betrachte Störungen in A und b :

$$(A + \Delta A)(x + \Delta x) = (b + \Delta b)$$

Für κ müsste $\|(A, b)\|$ festgelegt werden. Dies wird jedoch nicht betrachtet. Es gilt aber folgende Abschätzung für invertierbare Matrizen $A \in \mathbb{R}^{n \times n}$ und Störungen $\Delta A \in \mathbb{R}^{n \times n}$ mit $\|A^{-1}\| \cdot \|\Delta A\| < 1$:

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot (1 - \|A^{-1}\| \cdot \|\Delta A\|) \cdot \underbrace{\left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right)}_{\neq \frac{\|(\Delta A, \Delta b)\|}{\|(A, b)\|}}\tag{3.2.11}$$

Beweis. s. Übungsblatt

□

Definition 3.2.11. Sei $\|\cdot\|$ eine Norm auf $\mathbb{R}^{n \times n}$ und $A \in \mathbb{R}^{n \times n}$ eine reguläre Matrix. Die Größe

$$\kappa_{\|\cdot\|}(A) = \text{cond}_{\|\cdot\|}(A) := \|A\| \cdot \|A^{-1}\|$$

heißt **Kondition der Matrix** bzgl. der Norm $\|\cdot\|$.

Ist $\|\cdot\|$ von einer Vektor-Norm $\|\cdot\|_p$ induziert, bezeichnet $\text{cond}_p(A)$ die $\text{cond}_{\|\cdot\|_p}(A)$. Wir schreiben $\text{cond}(A)$ für $\text{cond}_2(A)$.

$\text{cond}_{\|\cdot\|}(A)$ schätzt die relative Kondition eines linearen GLS $Ax = b$ für alle möglichen

Störungen in b oder in A ab und diese Abschätzung ist scharf.

Es stellt sich nun die Frage:

Wann existiert die Inverse der gestörten invertierbaren Matrix A ?

Hierzu werden wir die Relationen benötigen:

$$A + \Delta A = A(I + A^{-1}\Delta A)$$

und mit $C \in \mathbb{R}^{n \times n}$, $\|C\| < 1$

$$(I - C)^{-1} = \sum_{k=0}^{\infty} C^k$$

$$\|(I - C)^{-1}\| \leq \frac{1}{1 - \|C\|}$$

Lemma 3.2.12. *Neumannsche Reihe* Sei $C \in \mathbb{R}^{n \times n}$ mit $\|C\| < 1$ und mit einer submultiplikativen Norm $\|\cdot\|$, so ist $(I - C)$ invertierbar und es gilt: 27.10.2014

$$(I - C)^{-1} = \sum_{k=0}^{\infty} C^k$$

Weiterhin gilt:

$$\|(I - C)^{-1}\| \leq \frac{1}{1 - \|C\|}$$

Beweis. Es gilt zu zeigen, dass $\sum_{k=1}^{\infty} C^k$ existiert:

Sei $q := \|C\| < 1$, dann gilt:

$$\begin{aligned} \left\| \sum_{k=0}^m C^k \right\| &\leq \sum_{k=0}^m \|C^k\| && \text{Dreiecksungleichung} \\ &\leq \sum_{k=0}^m \|C\|^k && \text{da } \|\cdot\| \text{ submultiplikativ} \\ &= \sum_{k=0}^m q^k \\ &= \frac{1 - q^{m+1}}{1 - q} \\ &\leq \frac{1}{1 - \|C\|} && \forall m \in \mathbb{N}, \text{ da } q < 1 \text{ (geometr. Reihe)} \end{aligned}$$

Daraus folgt bereits, dass $\sum_{k=1}^{\infty} C^k$ existiert (nach Majorantenkriterium).

Weiter gilt dann:

$$\begin{aligned} (I - C) \sum_{k=1}^{\infty} C^k &= \lim_{m \rightarrow \infty} (I - C) \sum_{k=1}^m C^k \\ &= \lim_{m \rightarrow \infty} (C^0 - C^{m+1}) \\ &= I \end{aligned}$$

□

Bemerkung 3.2.13.

a) Für symmetrische, positiv definite Matrix $A \in \mathbb{R}^{n \times n}$ gilt⁶:

$$\kappa_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \quad (3.2.13)$$

b) Eine andere Darstellung von $\kappa(A)$ ist

$$\kappa(A) := \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|} \in [0, \infty] \quad (3.2.14)$$

Diese ist auch für nicht invertierbare und rechteckige Matrizen wohldefiniert.
Dann gilt offensichtlich:

c) $\kappa(A) \geq 1$

d) $\kappa(\alpha A) = \kappa(A)$ für $0 \neq \alpha \in \mathbb{R}$ (skalierungsinvariant)

e) $A \neq 0$ und $A \in \mathbb{R}^{n \times n}$ ist genau dann singulär, wenn $\kappa(A) = \infty$.

Wegen der Skalierungsinvarianz ist die Kondition zur Überprüfung der Regularität von A besser geeignet als die Determinante.

Beispiel 3.2.14 (Kondition eines nichtlin. Gleichungssystems). Sei $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ stetig differenzierbar und $y \in \mathbb{R}^n$ gegeben.

Löse

$$f(x) = y$$

Gesucht:

$$\kappa(f^{-1}, y)$$

mit f^{-1} Ausgabe und y Eingabe.

Sei $Df(x)$ invertierbar, dann existiert aufgrund des Satzes für implizite Funktionen die inverse Funktion f^{-1} lokal in einer Umgebung von y mit $f^{-1}(y) = x$, sowie

$$D(f^{-1})(y) = (Df(x))^{-1}$$

Hiermit folgt:

$$\begin{aligned} \kappa_{abs}(f^{-1}, y) &= \|(Df(x))^{-1}\| \\ \kappa_{rel}(f^{-1}, y) &= \frac{\|f(x)\|}{\|x\|} \cdot \|(Df(x))^{-1}\| \end{aligned} \quad (3.2.15)$$

⁶Beweis: siehe Übungsblatt 3

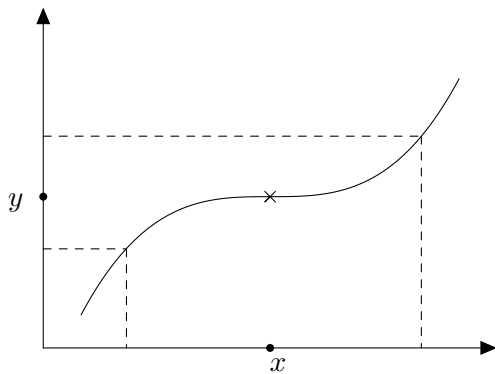
Für skalare Funktionen $f: \mathbb{R} \rightarrow \mathbb{R}$ folgt somit:

$$\kappa_{rel}(f^{-1}, y) = \frac{|f(x)|}{|x|} \cdot \frac{1}{|f'(x)|}$$

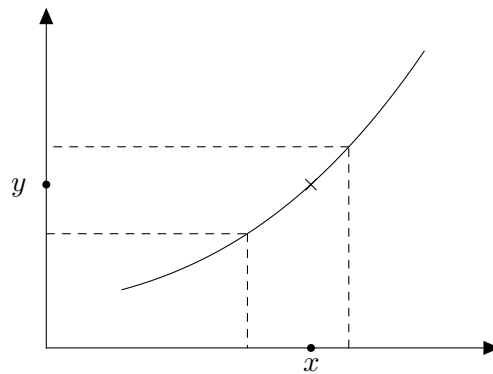
Falls $|f'(x)| \rightarrow 0$ ist es eine schlechte absolute Kondition.

Für $|f'(x)| \gg 0$ ist es eine gute absolute Kondition.

Damit bedeutet eine kleine Störung in y eine große Störung in x .



Schlechte Kondition



Gute Kondition

3.2b) Komponentenweise Konditionsanalyse

Beispiel 3.2.15. Falls A Diagonalgestalt hat, sind die Gleichungen unabhängig voneinander (entkoppelt). Die erwartete relative Kondition wäre dann – wie bei skalaren Gleichungen – stets gleich 1. Ebenso sind Störungen nur in der Diagonale zu erwarten. Jedoch:

$$\begin{aligned} A &= \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix} \\ \Rightarrow A^{-1} &= \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon^{-1} \end{pmatrix} \\ \Rightarrow \kappa_{\infty} = \kappa_2 &= \frac{1}{\varepsilon} \quad \text{für } 0 < \varepsilon \leq 1 \end{aligned}$$

Definition 3.2.16. Sei (f, x) ein Problem mit $f(x) \neq 0$ und $x = (x_i)_{i=1, \dots, n}$ mit $x_i \neq 0$ für alle $i = 1, \dots, n$. Die **komponentenweise Kondition** von (f, x) ist die kleinste Zahl $\kappa_{rel} \geq 0$, so dass:

$$\frac{\|f(\tilde{x}) - f(x)\|_{\infty}}{\|f(x)\|_{\infty}} \leq \kappa_{rel} \cdot \max_i \frac{|\tilde{x}_i - x_i|}{|x_i|} + o\left(\max_i \frac{|\tilde{x}_i - x_i|}{|x_i|}\right) \quad \text{für } \tilde{x} \rightarrow x$$

3 Fehleranalyse

Vorsicht:

$$\frac{\|\tilde{x} - x\|_\infty}{\|x\|_\infty} \neq \max_i \frac{|\tilde{x}_i - x_i|}{|x_i|}$$

Lemma 3.2.17. Sei f differenzierbar und fasse $|\cdot|$ komponentenweise auf, d.h. $|x| = \begin{pmatrix} |x_1| \\ \vdots \\ |x_n| \end{pmatrix}$.

Dann gilt:

$$\kappa_{rel} = \frac{\| |Df(x)| \cdot |x| \|_\infty}{\|f(x)\|_\infty} \quad (3.2.16)$$

Beweis. Vergleiche seien ebenfalls komponentenweise zu verstehen.
Nach dem Satz von Taylor gilt:

$$\begin{aligned} f_i(\tilde{x}) - f_i(x) &= \left(\frac{\partial f_i}{\partial x_1}(x), \dots, \frac{\partial f_i}{\partial x_n}(x) \right) \cdot \begin{pmatrix} \tilde{x}_1 - x_1 \\ \vdots \\ \tilde{x}_n - x_n \end{pmatrix} + o(\|\tilde{x} - x\|) \\ \Rightarrow |f_i(\tilde{x}) - f_i(x)| &\leq |Df(x)| \cdot \begin{pmatrix} |x_1| \cdot \frac{\tilde{x}_1 - x_1}{|x_1|} \\ \vdots \\ |x_n| \cdot \frac{\tilde{x}_n - x_n}{|x_n|} \end{pmatrix} + o\left(\max_i \frac{|\tilde{x}_i - x_i|}{|x_i|}\right) \quad \text{da } x_i \text{ fest und } \tilde{x}_i \rightarrow x_i \\ &\leq |Df(x)| \cdot |x| \cdot \max_i \frac{|\tilde{x}_i - x_i|}{|x_i|} + o\left(\max_i \frac{|\tilde{x}_i - x_i|}{|x_i|}\right) \\ \Rightarrow \frac{\|f(\tilde{x}) - f(x)\|_\infty}{\|f(x)\|_\infty} &\leq \frac{\| |Df(x)| \cdot |x| \|_\infty}{\|f(x)\|_\infty} \cdot \max_i \frac{|\tilde{x}_i - x_i|}{|x_i|} + o\left(\max_i \frac{|\tilde{x}_i - x_i|}{|x_i|}\right) \end{aligned}$$

Wähle $\tilde{x}_i = x_j + h \cdot \text{sign} \frac{\partial f_i}{\partial x_j}(x)$ mit $h > 0$, dann gilt:

$$|Df_i(x)(\tilde{x} - x)| = Df_i(x)(\tilde{x} - x)$$

und in obiger Rechnung gilt Gleichheit.

Also folgt, dass

$$\frac{\| |Df(x)| \cdot |x| \|_\infty}{\|f(x)\|_\infty} = \kappa_{rel}$$

□

Beispiel 3.2.18.

a) Komponentenweise Kondition der Multiplikation

$$\begin{aligned}
 f: \mathbb{R}^2 &\rightarrow \mathbb{R}, f(x, y) := x \cdot y \\
 \Rightarrow Df(x, y) &= (y, x) \\
 \Rightarrow \kappa_{rel}(x, y) &= \frac{\left\| (|y|, |x|) \cdot \begin{pmatrix} |x| \\ |y| \end{pmatrix} \right\|_{\infty}}{|x \cdot y|} \\
 &= \frac{2 \cdot |x| \cdot |y|}{|x \cdot y|} \\
 &= 2
 \end{aligned}$$

b) Komponentenweise Kondition eines linearen Gleichungssystems:

Löse $Ax = b$ mit möglichen Störungen in b , also zu

$$\begin{aligned}
 f: b &\mapsto A^{-1}b \\
 \kappa_{rel} &= \frac{\|A^{-1} \cdot |b|\|_{\infty}}{\|A^{-1}b\|_{\infty}}
 \end{aligned}$$

Falls A eine Diagonalmatrix ist, folgt:

$$\kappa_{rel} = 1$$

c) Komponentenweise Kondition des Skalarproduktes:

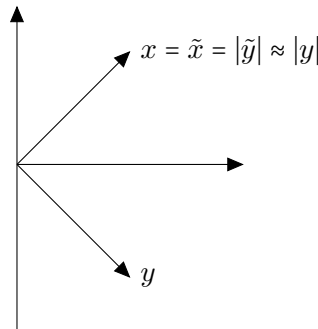
$$\begin{aligned}
 \langle x, y \rangle &:= \sum_{i=1}^n x_i y_i = x^T y \\
 f: \mathbb{R}^2 &\rightarrow \mathbb{R}, f(x, y) = \langle x, y \rangle \\
 \Rightarrow Df(x, y) &= (y^T, x^T) \\
 \kappa_{rel} &= \frac{\left\| |(y^T, x^T)| \cdot \begin{pmatrix} |x| \\ |y| \end{pmatrix} \right\|_{\infty}}{\|\langle x, y \rangle\|_{\infty}} \\
 &= \frac{2 \cdot |y^T| \cdot |x|}{|\langle x, y \rangle|} \\
 &= 2 \cdot \frac{\langle |x|, |y| \rangle}{|\langle x, y \rangle|} \\
 &= 2 \cdot \frac{\cos(|x|, |y|)}{\cos(x, y)}
 \end{aligned}$$

$$\text{da } \cos(x, y) = \frac{\langle y, x \rangle}{\|x\|_2 \cdot \|y\|_2}.$$

Falls x und y nahezu senkrecht aufeinander stehen, kann das Skalarprodukt sehr schlecht konditioniert sein.

3 Fehleranalyse

Zum Beispiel für $x = \tilde{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ und $y = \begin{pmatrix} 1 + 10^{-10} \\ -1 \end{pmatrix}$, $\tilde{y} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

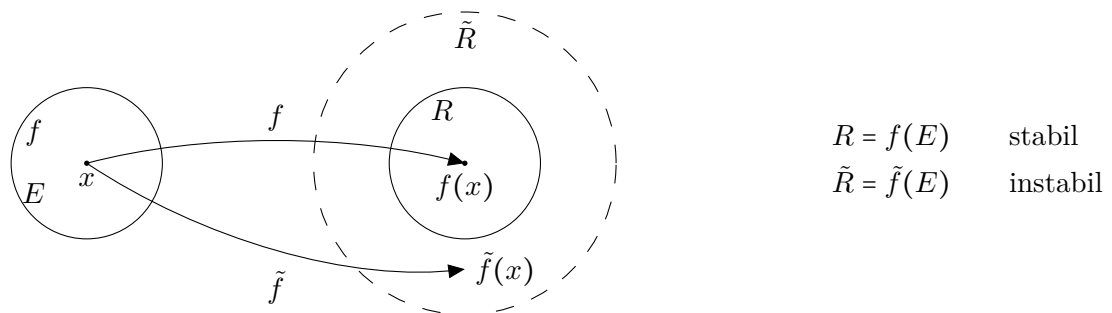


3.3 Stabilität von Algorithmen

Bislang: Kondition eines gegebenen Problems (f, x) .

Nun stellt sich die Frage: *Was passiert durch das Implementieren am Rechner?*

Ein „stabiler“ Algorithmus sollte ein gut konditioniertes Problem nicht „kaputt machen“.



3.3a) Vorwärtsanalyse

Die Fehlerfortpflanzung durch die einzelnen Rechenschritte, aus denen die Implementierung aufgebaut ist, wird abgeschätzt.

Bemerkung 3.3.1. Für die Rechenoperationen $+$, $-$, \cdot , $/$, kurz ∇ , gilt:

$$\begin{aligned} fl(a \nabla b) &= (a \nabla b) \cdot (1 + \varepsilon) \\ &= (a \nabla b) \cdot \frac{1}{1 + \mu} \end{aligned} \tag{3.3.1}$$

mit $|\varepsilon|, |\mu| \leq eps$.

Beispiel 3.3.2. Sei $f(x_1, x_2, x_3) := \frac{x_1 x_2}{x_3}$ mit Maschinenzahlen x_i und $x_3 \neq 0$ und sei der Algorithmus durch

$$f(x_1, x_2, x_3) = (f^{(2)} \circ f^{(1)})(x_1, x_2, x_3)$$

gegeben mit

$$\begin{aligned} f^{(1)}(x_1, x_2, x_3) &= (x_1 \cdot x_2, x_3) & \text{und} \\ f^{(2)}(y, z) &= \frac{y}{z} \end{aligned}$$

Die Implementierung \tilde{f} von f beinhaltet Rundungsfehler.

Sei $x = (x_1, x_2, x_3)$. Daraus folgt:

$$\begin{aligned} \tilde{f}^{(1)}(x) &= (fl(x_1 \cdot x_2), x_3) \\ &= (x_1 x_2 (1 + \varepsilon_1), x_3) \end{aligned}$$

mit $|\varepsilon_1| \leq eps$:

$$\begin{aligned} \tilde{f}(x) &= \tilde{f}^{(2)}(\tilde{f}^{(1)}(x)) \\ &= fl(f^{(2)}(x_1 x_2 (1 + \varepsilon_1), x_3)) \\ &= \frac{x_1 x_2 (1 + \varepsilon_1)}{x_3} \cdot (1 + \varepsilon_2) \\ &= f(x) \cdot (1 + \varepsilon_1)(1 + \varepsilon_2) \end{aligned}$$

mit $|\varepsilon_2| \leq eps$:

$$\begin{aligned} \frac{|\tilde{f}(x) - f(x)|}{|f(x)|} &= |\varepsilon_1 + \varepsilon_2 + \varepsilon_1 \cdot \varepsilon_2| \\ &\leq 2eps + eps^2 \end{aligned}$$

Dies ist eine „worst case“ Analyse, da immer der maximale Fehler angenommen wird, und gibt i.d.R. eine starke Überschätzung des Fehlers an. Für qualitative Aussagen sind sie jedoch unnützlich.

In Computersystemen stehen mehr Operationen wie ∇ zur Verfügung, die mit einer relativen Genauigkeit eps realisiert werden können.

Daher:

Definition 3.3.3. Eine Abbildung $\phi : U \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m$ heißt **elementar ausführbar**, falls es eine elementare Operation $\tilde{\phi} : \mathbb{F}^n \rightarrow \mathbb{F}^m$ gibt, wobei \mathbb{F} die Menge der Maschinenzahlen bezeichne mit

$$|\tilde{\phi}_i(x) - \phi_i(x)| \leq eps \cdot |\phi_i(x)| \quad \forall x \in \mathbb{F}^n \text{ und } i = 1, \dots, m. \quad (3.3.2)$$

$\tilde{\phi}$ heißt dann **Realisierung** von ϕ .

3 Fehleranalyse

Bemerkung

aus (3.3.2) folgt für $1 \leq p \leq \infty$:

$$\|\tilde{\phi}(x) - \phi(x)\|_p \leq \text{eps} \cdot \|\phi(x)\|_n \quad \forall x \in \mathbb{F}^n \quad (3.3.3)$$

Definition 3.3.4. Sei $f : E \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m$ gegeben.

Ein Tupel $(f^{(1)}, \dots, f^{(l)})$ mit $l \in \mathbb{N}$ von elementar ausführbaren Abbildungen

$$f^{(i)} : U_1 \subseteq \mathbb{R}^{k_i} \rightarrow U_{i+1} \subseteq \mathbb{R}^{k_{i+1}}$$

mit $k_1 = n$ und $k_{l+1} = m$ heißt **Algorithmus** von f , falls

$$f = f^{(l)} \circ \dots \circ f^{(1)}$$

Das Tupel $(\tilde{f}^{(1)}, \dots, \tilde{f}^{(l)})$ mit Abbildungen $\tilde{f}^{(i)}$, welche Realisierungen der $f^{(i)}$ sind, heißt **Implementation** von $(f^{(1)}, \dots, f^{(l)})$. Die Komposition

$$\tilde{f} = \tilde{f}^{(l)} \circ \dots \circ \tilde{f}^{(1)}$$

heißt Implementation von f .

Im Allgemeinen gibt es verschiedene Implementierungen einer Abbildung f .

Lemma 3.3.5 (Fehlerfortpflanzung). Sei $x \in \mathbb{R}^n$ und $\tilde{x} \in \mathbb{F}^n$ mit $|\tilde{x}_i - x_i| \leq \text{eps}|x_i|$ für alle $i = 1, \dots, n$. Sei $(f^{(1)}, \dots, f^{(l)})$ ein Algorithmus für f und $(\tilde{f}^{(1)}, \dots, \tilde{f}^{(l)})$ eine zugehörige Implementation.

Mit den Abkürzungen

$$\begin{aligned} x^{(j+1)} &:= f^{(j)} \circ \dots \circ f^{(1)}(x) \\ x^{(1)} &:= x \end{aligned}$$

und entsprechend mit $\tilde{x}^{(j+1)}$ gilt, falls $x^{(j+1)} \neq 0$ für alle $j = 0, \dots, (l-1)$ und $\|\cdot\|$ eine beliebige p -Norm ist:

$$\begin{aligned} \frac{\|\tilde{x}^{(j+1)} - x^{(j+1)}\|}{\|x^{(j+1)}\|} &\leq \text{eps} \cdot \mathcal{K} + o(\text{eps}) \\ \mathcal{K}^{(j)} &= (1 + \kappa^{(j)} + \kappa^{(j)} \cdot \kappa^{(j-1)} + \dots + \kappa^{(j)} \cdot \dots \cdot \kappa^{(1)}) \end{aligned} \quad (3.3.4)$$

wobei $\kappa^{(j)} := \kappa_{\text{rel}}(f^{(j)}, x^{(j)})$ die Kondition der elementar ausführbaren Operationen $f^{(j)}$ ist.

Beweis.

$$\begin{aligned}
 \frac{\|\tilde{x}^{(j+1)} - x^{(j+1)}\|}{\|x^{(j+1)}\|} &= \frac{\|\tilde{f}^{(j)}(\tilde{x}^{(j)}) - f^{(j)}(x^{(j)})\|}{\|f^{(j)}(x^{(j)})\|} \\
 &\leq \frac{\|\tilde{f}(\tilde{x}) - f(\tilde{x})\|}{\|f(\tilde{x})\|} \cdot \frac{\|f(\tilde{x})\|}{\|f(x)\|} + \frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} \quad (\text{Index } j \text{ vernachlässigt}) \\
 &\leq \text{eps} \left(1 + \frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} \right) + \frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} \\
 &\stackrel{\text{nach 3.3.3}}{=} \text{eps} + (\text{eps} + 1) \cdot \left(\kappa(j) \cdot \frac{\|\tilde{x}^{(j)} - x^{(j)}\|}{\|x^{(j)}\|} \right) + o \left(\frac{\|\tilde{x}^{(j)} - x^{(j)}\|}{\|x^{(j)}\|} \right)
 \end{aligned}$$

Nach Voraussetzung gilt Gleichung (3.3.4) mit $\mathcal{K}^{(0)} = 1$ für $j = 0$.

Für $j = 1$ folgt nach Voraussetzung mit Gleichung (3.3.3)

$$\begin{aligned}
 \frac{\|\tilde{x}^{(2)} - x^{(2)}\|}{\|x^{(2)}\|} &\leq \text{eps} + (\text{eps} + 1) \cdot \left(\kappa^{(1)} \text{eps} + o(\text{eps}) \right) \\
 &= \text{eps}(1 + \kappa^{(1)}) + o(\text{eps}) \\
 &= \text{eps}\mathcal{K}^{(1)} + o(\text{eps})
 \end{aligned}$$

Womit der Induktionsanfang gezeigt ist.

Für den Induktionsschritt von $j - 1$ zu j :

$$\begin{aligned}
 \frac{\|\tilde{x}^{(j+1)} - x^{(j+1)}\|}{\|x^{(j+1)}\|} &\leq \text{eps} + (1 + \text{eps})\kappa^{(j)} \left[\text{eps}\mathcal{K}^{(j-1)} + o(\text{eps}) \right] \\
 &\quad + (1 + \text{eps}) \cdot o \left(\text{eps} \cdot \mathcal{K}^{(j-1)} + o(\text{eps}) \right) \\
 &= \text{eps} \left(1 + \kappa^{(j)} \cdot \mathcal{K}^{(j-1)} \right) + o(\text{eps})
 \end{aligned}$$

Mit $\mathcal{K}^{(j)} = 1 + \kappa^{(j)} \cdot \mathcal{K}^{(j-1)}$ folgt die Behauptung. □

Hiermit folgt:

Korollar 3.3.6. *Unter der Voraussetzung von Lemma 3.3.5 gilt:*

$$\frac{\|\tilde{f}(\tilde{x}) - f(x)\|}{\|f(x)\|} \leq \text{eps} \cdot \left(1 + \kappa^{(l)} + \kappa^{(l)} \cdot \kappa^{(l-1)} + \dots + \kappa^{(l)} \cdot \dots \cdot \kappa^{(1)} \right) + o(\text{eps}) \quad (3.3.5)$$

Bemerkung 3.3.7. Mit Korollar 3.3.6 ist offensichtlich, dass schlecht konditionierte Probleme zu elementar ausführbaren Abbildungen so früh wie möglich ausgeführt werden sollten.

Nach Beispiel 3.2.9 ist die Subtraktion zweier annähernd gleicher Zahlen schlecht konditioniert. Deshalb sollte man unvermeidbare Subtraktionen möglichst früh durchführen. Allerdings hängt $\kappa^{(j)}$ nicht nur von $f^{(j)}$, sondern auch vom Zwischenergebnis $x^{(j)}$ ab, welches a priori unbekannt ist.

Bemerkung 3.3.8 (Sprechweise). Der Quotient

$$\frac{\overbrace{\|\tilde{f}(\tilde{x}) - f(x)\|}^{\text{Gesamtfehler}}}{\|f(x)\|} = \underbrace{\frac{\|\tilde{f}(\tilde{x})\|}{\|f(x)\|}}_{\substack{\text{Fehler} \\ \text{durch} \\ \text{Problem}}} \cdot \underbrace{\frac{\|\tilde{x} - x\|}{\|x\|}}_{\substack{\text{Eingabe-} \\ \text{fehler}}} \quad (3.3.6)$$

gibt die **Güte des Algorithmus** an. Als Stabilitätsindikator kann also

$$\sigma(f, \tilde{f}, x) := \frac{\mathcal{K}}{\kappa_{rel}(f, x)} \quad (3.3.7)$$

verwendet werden und es gilt

$$\frac{\|\tilde{f}(\tilde{x}) - f(x)\|}{\|f(x)\|} < \underbrace{\sigma(f, \tilde{f}, x)}_{\substack{\text{Beitrag} \\ \text{des} \\ \text{Algorithmus}}} \cdot \underbrace{\kappa_{rel}(f, x)}_{\substack{\text{Beitrag} \\ \text{des} \\ \text{Problems}}} \cdot \underbrace{eps}_{\substack{\text{Rundungs-} \\ \text{fehler}}} + o(eps)$$

Falls $\sigma(f, \tilde{f}, x) < 1$, dämpft der Algorithmus die Fehlerfortpflanzung der Eingabe- und Rundungsfehler und heißt **stabil**.

Für $\sigma(f, \tilde{f}, x) \gg 1$ heißt der Algorithmus **instabil**.

Beispiel 3.3.9. Nach Gleichung (3.3.3) gilt für die Elementaroperationen $\mathcal{K} \leq 1$. Da für die Subtraktion zweier annähernd gleich großer Zahlen $\kappa_{rel} \gg 1$ gilt, ist der Stabilitätsfaktor zweier annähernd gleich großer Zahlen sehr klein und der Algorithmus also stabil. Falls es sich jedoch bei einer zusammengesetzten Abbildung $f = h \circ g$ bei der zweiten Abbildung h um eine Subtraktion handelt, gilt

$$\mathcal{K} = (1 + \kappa(sub) + \kappa(sub) \cdot \kappa(g))$$

und die Stabilität ist gefährdet. Genauere Abschätzungen und damit genauere Indikatoren können durch komponentenweise Betrachtungen erhalten werden.

3.3b) Rückwärtsanalyse

Die Fragestellung ist nun:

Kann $\tilde{f}(\tilde{x})$ als exaktes Ergebnis von einer gestörten Eingabe \tilde{x} unter der exakten Abbildung f aufgefasst werden?

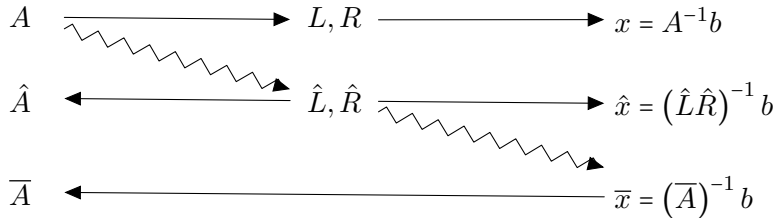
Das würde heißen

$$\exists \hat{x} \in \mathbb{R}^n : f(\hat{x}) = \tilde{f}(\tilde{x}).$$

Dann schätze den Fehler $\|\hat{x} - x\|$ bzw. für nicht injektive f

$$\min_{\hat{x} \in \mathbb{R}^n} \{ \|\hat{x} - x\| \mid f(\hat{x}) = \tilde{f}(\hat{x}) \}$$

ab.



Ein Anwendungsbeispiel:

Die Eingangsdaten seien Messdaten \tilde{x} mit 1 % relativer Genauigkeit. Liefert die Rückwärtsanalyse, dass $\tilde{f}(\tilde{x})$ als exaktes Ergebnis $f(\hat{x})$ mit Eingangsdaten \hat{x} , die höchstens um 0,5 % schwanken, aufgefasst werden kann, so ist das Verfahren „geeignet“.

Die Rückwärtsanalyse ist

- in der Regel leichter durchführbar als die Vorwärtsanalyse und
- ebenfalls nur eine qualitative Schätzung der Genauigkeit der numerisch berechneten Werte.

Bemerkung 3.3.10.

Vorwärtsfehler \leq Kondition des Problems \cdot Rückwärtsfehler.

$$\|\tilde{f}(\tilde{x}) - f(x)\| \leq \kappa(f, x) \|\hat{x} - x\|$$

Beispiel: Rückwärtsanalyse der Gauß-Elimination (geht auf Wilkinson zurück)

Satz 3.3.11. $A \in \mathbb{R}^{n \times n}$ besitze eine LR-Zerlegung. Dann berechnet die Gauß-Elimination Matrizen \hat{L} und \hat{R} , so dass

$$\hat{L} \hat{R} = \hat{A}$$

und

$$\begin{aligned} \|\hat{A} - A\| &\leq \frac{\epsilon ps}{1 - n \cdot \epsilon ps} \left(\|\hat{L}\| \left\| \begin{pmatrix} 1 & & 0 \\ & 2 & \\ 0 & \ddots & n \end{pmatrix} \right\| \|\hat{R}\| - \|\hat{R}\| \right) \\ &\leq \frac{n \cdot \epsilon ps}{1 - n \cdot \epsilon ps} \|\hat{L}\| \|\hat{R}\| \\ &= n \cdot \epsilon ps \|\hat{L}\| \cdot \|\hat{R}\| + \mathcal{O}(n^2 \epsilon ps^2) \end{aligned}$$

falls $n \cdot \epsilon ps \leq \frac{1}{2}$.

3 Fehleranalyse

Beweis. [siehe SB90]. □

Satz 3.3.12 (Sautter 1971). $A \in \mathbb{R}^{n \times n}$ besitze eine LR-Zerlegung. Dann berechnet das Gaußsche Eliminationsverfahren für das Gleichungssystem $Ax = b$ eine Lösung \bar{x} mit

$$\bar{A}\bar{x} = b$$

mit

$$|\bar{A} - A| \leq 2n \epsilon_{ps} |\hat{L}| |\hat{R}| + \mathcal{O}(n^2 \epsilon_{ps}^2).$$

Beweis. [siehe DH08]. □

Weitere Abschätzungen existieren für Gauß-Elimination mit Pivotisierung und für spezielle Klassen von Matrizen.

3.3.13 Allgemeine Faustregeln für die LR-Zerlegung

- Falls die Matrix $n|\hat{L}||\hat{R}|$ die selbe Größenordnung wie $|A|$ besitzt, ist der Algorithmus „gutartig“;
- Für tridiagonale Matrizen ist der Algorithmus mit Spaltenpivotisierung stabil.
- Falls A oder A^T **strikt diagonal dominant** ist, d.h.

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}| \text{ für alle } i = 1, \dots, n,$$

ist Spaltenpivotisierung überflüssig. Der Algorithmus ist stabil.

- Für symmetrische, positiv definite Matrizen sollte keine Pivotisierung durchgeführt werden, um die Symmetrie zu erhalten. Der Algorithmus ist stabil.

Vorsicht

Selbst wenn die LR-Zerlegung stabil ist, in dem Sinne dass $|\bar{A} - A|$ klein ist für $\bar{A}\bar{x} = b$, kann die numerische Lösung \bar{x} sehr ungenau sein, da der Vorwärtsfehler $|\bar{x} - x|$ auch von der Kondition abhängt.

Ein Beispiel hierzu ist die Hilbertmatrix

$$H = \left(\frac{1}{i+j-1} \right)_{i,j=1,\dots,n},$$

für die $\text{cond}(H)$ exponentiell mit der Dimension n wächst.

3.4 Beurteilung von Näherungslösungen linearer GLS

Zu $Ax = b$ liege eine Näherungslösung \tilde{x} vor.

3.4a) Im Sinne der Vorwärtsanalyse

Im Sinne der Vorwärtsanalyse und der Fehlerentwicklung durch das Problem gilt:

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \text{cond}(A) \cdot \frac{\|\Delta b\|}{\|b\|}$$

nach Beispiel 3.2.10, mit dem Residuum

$$\begin{aligned} r(\tilde{x}) &:= A\tilde{x} - b \\ &= \tilde{b} - b \\ &= \Delta b \end{aligned} \tag{3.4.1}$$

Wie der absolute Fehler ist das Residuum skalierungsabhängig. Daher ist $\|r(\tilde{x})\|$ „klein“ ungeeignet, um Genauigkeitsaussagen zu treffen.

Um den Fehler in x abzuschätzen, ist die Betrachtung von

$$\frac{\|r(\tilde{x})\|}{\|b\|} \tag{3.4.2}$$

geeigneter.

Für große $\text{cond}(A)$ ist dieser Quotient jedoch weiterhin ungeeignet.

3.4b) Im Sinne der Rückwärtsanalyse

Satz 3.4.1 (Prager und Oettli, 1964). *Sei \tilde{x} eine Näherungslösung für $Ax = b$. Falls*

$$|r(\tilde{x})| \leq \varepsilon(|A||\tilde{x}| + |b|). \tag{3.4.3}$$

dann existiert eine Matrix \tilde{A} und ein Vektor \tilde{b} , so dass

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

und

$$|\tilde{A} - A| \leq \varepsilon|A| \quad \text{und} \quad |\tilde{b} - b| \leq \varepsilon|b|. \tag{3.4.4}$$

Aufgrund von (3.4.3) wird der komponentenweise relative Rückwärtsfehler durch

$$\max_i \frac{|A\tilde{x} - b|_i}{(|A||\tilde{x}| + |b|)_i}$$

3 Fehleranalyse

abgeschätzt.

Für den normweisen relativen Rückwärtsfehler gilt entsprechend (Rigal und Gaches 1967)

$$\frac{\|A\tilde{x} - b\|}{\|A\|\|\tilde{x}\| + \|b\|}.$$

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

4.1 Gaußsches Eliminationsverfahren mit Äquilibration und Nachiteration

Mit Skalierung $D_z A$ (**Zeilen**skalierung) oder $D_s A$ (**Spalten**skalierung) mittels Diagonalmatrizen D_z, D_s lässt sich eine Pivotstrategie beliebig abändern. Jetzt ist die Frage: *Was ist eine „gute“ Skalierung?*

Skalierung ändert die Länge der Basisvektoren des Bild- bzw. des Urbildvektorraumes. Durch Normierung der Länge auf 1 wird die Pivotstrategie unabhängig von der gewählten Einheit.

Sei $A \in \mathbb{R}^{n \times m}$ und $\|\cdot\|$ eine Vektornorm.

4.1.1 Äquilibration der Zeilen

Alle Zeilen von $D_z A$ haben die gleiche Norm, z.B. $\|\cdot\| = 1$, wofür

$$D_z = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \end{pmatrix} \quad \text{mit } \sigma_i := \frac{1}{\|(a_{i1}, \dots, a_{im})\|} \quad (4.1.1)$$

gesetzt wird.

4.1.2 Äquilibration der Spalten

Alle Spalten von $A D_s$ haben die gleiche Norm, z.B. $\|\cdot\| = 1$, wofür

$$D_s = \begin{pmatrix} \tau_1 & & 0 \\ & \ddots & \\ 0 & & \tau_m \end{pmatrix} \quad \text{mit } \tau_j := \left\| \begin{pmatrix} a_{1j} \\ \vdots \\ a_{nj} \end{pmatrix} \right\|^{-1} \quad (4.1.2)$$

gesetzt wird.

Äquilibration von Zeilen **und** Spalten führt zu einem nichtlinearen Gleichungssystem und ist i.d.R. aufwendig.

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

Lemma 4.1.3. Sei A zeilenäquibriert bzgl. der l_1 -Norm, dann gilt:

$$\text{cond}_\infty(A) \leq \text{cond}_\infty(DA) \quad (4.1.3)$$

für alle regulären Diagonalmatrizen D .

Beweis. siehe Übungsaufgabe □

Wie in Kapitel 3 gesehen, kann die Näherungslösung \tilde{x} trotz Pivotisierung und Äquibrierung noch sehr ungenau sein.

4.1.4 Nachiteration

Die Näherung \tilde{x} kann durch Nachiteration verbessert werden.
Falls \tilde{x} exakt ist, gilt:

$$r(\tilde{x}) := b - A\tilde{x} = 0 \quad (4.1.4)$$

ansonsten ist $A(x - \tilde{x}) = r(\tilde{x})$. Also löse die Korrekturgleichung

$$A\Delta x = r(\tilde{x}) \quad (4.1.5)$$

und setze

$$x^{(1)} := \tilde{x} + \Delta x$$

Wiederhole dies sooft, bis $x^{(i)}$ „genau genug“ ist. Die Lösung \tilde{x} wird durch Nachiteration meist mit sehr gutem Erfolg verbessert [genaueres in DR08]

(4.1.5) wird mit der bereits vorhandenen LR-Zerlegung nur mit der neuen rechten Seite $r(\tilde{x})$ gelöst, d.h. eine vorwärts und eine Rückwärtssubstitution mit $\mathcal{O}(n^2)$ flops.

Bemerkung 4.1.5 (nach Skeel 1980). Die Gauß-Elimination mit Spaltenpivotsuche und einer Nachiteration ist komponentenweise stabil.

4.2 Cholesky-Verfahren

Im Folgenden sei A eine symmetrische, positiv definite Matrix in $\mathbb{R}^{n \times n}$, d.h. $A = A^T$ und $\langle x, Ax \rangle = x^T Ax > 0$ für alle $x \neq 0$.

(kurs: **spd Matrix**)

Satz 4.2.1. Für jede spd Matrix $A \in \mathbb{R}^{n \times n}$ gilt:

- i) A ist invertierbar
- ii) $a_{ii} > 0$ für $i = 1, \dots, n$

$$iii) \max_{ij} |a_{ij}| = \max_i a_{ii}$$

iv) Bei der Gauß-Elimination ohne Pivotsuche ist jede Restmatrix wieder eine spd Matrix.

Beweis.

i) Es gilt $x^T A x \neq 0 \Rightarrow A x \neq 0$. Nach ii) ist $A x \neq 0 \quad \forall x \in \mathbb{R}^n \setminus \{0\}$ also $\ker(A) = 0$.

ii) Sei e_i der i-te Einheitsvektor, so folgt $a_{ii} = e_i^T A e_i > 0$.

iii) siehe Übungsaufgabe

iv) Es gilt:

$$\begin{aligned} A^{(1)} &:= A = \begin{pmatrix} a_{11} & z^T \\ z & B^{(1)} \end{pmatrix} \\ A^{(2)} &:= L_1 A^{(1)} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -\frac{z}{a_{11}} & & & I \end{pmatrix} = \begin{pmatrix} a_{11} & z^T \\ 0 & & & B^{(2)} \\ \vdots & & & \\ 0 & & & \end{pmatrix} \\ \Rightarrow L_1 A^{(1)} L_1^T &= \begin{pmatrix} a_{11} & z^T \\ 0 & B^{(2)} \\ \vdots & \\ 0 & \end{pmatrix} \cdot \begin{pmatrix} 1 & -\frac{z}{a_{11}} \\ 0 & I \\ \vdots & \\ 0 & \end{pmatrix} \\ &= \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & & B^{(2)} \\ 0 & & & \end{pmatrix} \end{aligned}$$

Weiterhin gilt:

$$x \neq 0 \Leftrightarrow L_1 x \neq 0$$

da L_1 invertierbar. Also gilt insgesamt:

$$\begin{aligned} \tilde{x}^T B^{(2)} \tilde{x} &= x^T L_1 A^{(1)} L_1^T x && \text{für } x := \begin{pmatrix} 0 \\ \tilde{x} \end{pmatrix} \\ &= (L_1^T x)^T A (L_1^T x) > 0 && \forall \tilde{x} \neq 0 \end{aligned}$$

und damit ist auch $B^{(2)}$ spd.

Induktiv folgt hiermit iv).

□

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

Insbesondere ergibt sich:

$$(L_{n-1} \cdots L_1)A^{(1)}(L_1^T \cdots L_{n-1}^T) = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix},$$

wobei d_i das i -te Diagonalelement von $A^{(i)}$ ist und somit $d_i > 0$ für $i = 1, \dots, n$ gilt.

Sei $L := (L_1^{-1} \cdots L_{n-1}^{-1})$ wie in (2.1.8), so ergibt sich:

Folgerung 4.2.2 (Cholesky-Zerlegung). *Für jede spd Matrix A existiert eine eindeutige Zerlegung der Form*

$$A = LDL^T$$

wobei L eine reelle unipotente (d.h. $l_{ii} = 1$) (, normierte) untere Dreiecksmatrix und D eine positive Diagonalmatrix ist. Diese Zerlegung heißt **rationale Cholesky-Zerlegung**. Die Zerlegung

$$A = \bar{L}\bar{L}^T \tag{4.2.1}$$

mit der reellen unteren Dreiecksmatrix

$$\bar{L} = L \begin{pmatrix} \sqrt{d_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{d_n} \end{pmatrix} = LD^{\frac{1}{2}}$$

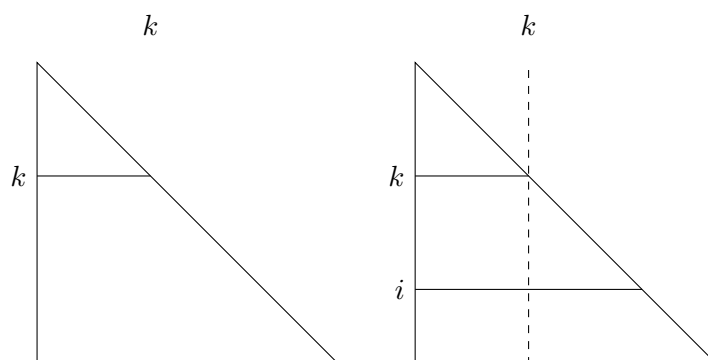
heißt **Cholesky-Zerlegung**.

Wegen (4.2.1) gilt:

$$a_{kk} = \bar{l}_{k1}^2 + \cdots + \bar{l}_{kk}^2 \tag{4.2.2}$$

$$a_{ik} = \bar{l}_{i1}\bar{l}_{k1} + \cdots + \bar{l}_{ik}\bar{l}_{kk} \tag{4.2.3}$$

Demnach funktioniert spaltenweises und zeilenweises Berechnen.



Es ergibt sich folgender Algorithmus:

4.2.3 Cholesky-Zerlegung

Der Algorithmus der Cholesky-Zerlegung ist wie folgt:

```

for  $k = 1, \dots, n$ 
|    $l_{kk} = (a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2)^{\frac{1}{2}}$ 
|   for  $i = k + 1, \dots, n$ 
|   |    $l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj}) / l_{kk}$ 
|   end
end

```

4.2.4 Rechenaufwand in flops

Es sind je

$\frac{1}{6}(n^2 - n)$ Additionen sowie Multiplikationen und

$\frac{1}{6}(3n^2 - 3n)$ Divisionen

also ca. $\frac{2}{3}n^2$ flops für große n notwendig.

Im Vergleich zur LR-Zerlegung halbiert sich in etwa der Aufwand.

Bemerkung 4.2.5.

- a) Wegen (4.2.2) gilt $|\bar{l}_{kj}| \leq \sqrt{a_{kk}}$, d.h. die Matrizeneinträge können nicht zu groß werden.
- b) Für spd Matrizen ist der Cholesky-Algorithmus stabil nach (3.3.13)
- c) Da A symmetrisch ist, muss nur die untere Dreiecksmatrix gespeichert werden. In Algorithmen kann \bar{L} in eine Kopie dieser Dreiecksmatrix geschrieben werden.
- d) Fast singuläre Matrizen können durch die Diagonale erkannt werden.

4.3 Lineare Ausgleichsprobleme

10.11.2014

Beispiel 4.3.1. (s. Einführung)

Seien m Messungen (I_i, U_i) für die Stromstärke I und die Spannung U gegeben.

Das Ohmsche Gesetz liefert hierfür:

$$U = R \cdot I$$

Gesucht ist der zugehörige Widerstand R .

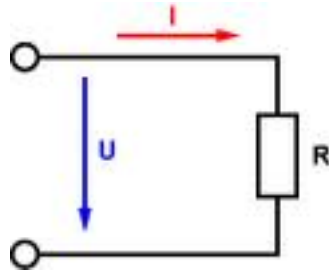


Abbildung 4.1: Schaltplan einer einfachen U-I-Messung

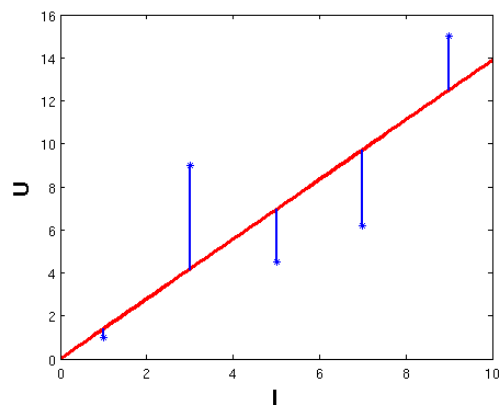


Abbildung 4.2: Linearausgleich einer U-I-Messung mit Ursprungsgerade als Modellfunktion

Wird jetzt davon ausgegangen dass die I_i exakt sind, wird das R gesucht, für das RI_i im Mittel den minimalen Abstand zu U_i hat. Genauer gesagt berechne

$$\min_{r \in \mathbb{R}} \sum_{i=1}^m (U_i - rI_i)^2$$

Vorsicht: Es wird **nicht** die Gerade (bzw. der lineare Untervektorraum) mit minimalem euklidischem Abstand zu (I_i, U_i) gesucht!

Dieses Problem ist nichtlinear und aufwendig zu lösen.

4.3.2 Lineares Ausgleichsproblem

Gegeben seien Messdaten (t_i, b_i) mit $t_i, b_i \in \mathbb{R}$ für $i = 1, \dots, m$ und die Abhängigkeit $b(t)$ werde beschrieben durch eine Modellfunktion, welche linear von den unbekannten Parametern x_1, \dots, x_n des Modells abhängt, d.h.

$$b(t) = a_1(t)x_1 + \dots + a_n(t)x_n$$

Für exakte Messdaten b_i würde

$$b(t_i) = b_i \quad \forall i \in \{1, \dots, m\}$$

gelten.

Im Allgemeinen werden jedoch $m \geq n$ Messwerte b_i bestimmt, und hiermit die n Parameter x_i so gewählt, dass die kleinsten **Fehlerquadrate auftreten**:

$$\min_{x_1, \dots, x_n} \sum_{i=1}^m (b_i - b(t_i))^2 \quad (4.3.1)$$

(Nach Gauß kann (4.3.1) auch aus der Maximum-Likelihood-Methode für einen stochastischen Ansatz hergeleitet werden.)

Definiere:

$$\begin{aligned} b &= (b_i)_{i=1, \dots, m} \in \mathbb{R}^m \\ x &= (x_j)_{j=1, \dots, n} \in \mathbb{R}^n \\ A &= (a_j(t_i))_{\substack{i=1, \dots, m \\ j=1, \dots, n}} \in \mathbb{R}^{m \times n} \end{aligned}$$

Damit ist (4.3.1) äquivalent zum **linearen Ausgleichsproblem**:

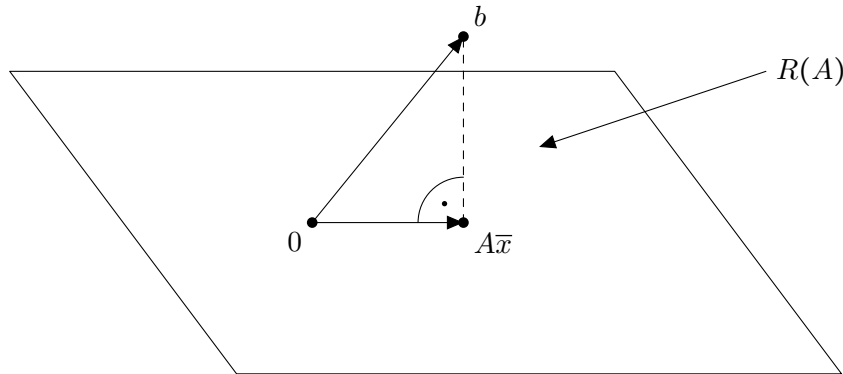
Zu gegebenem $b \in \mathbb{R}^m$ und $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ ist das $\bar{x} \in \mathbb{R}^n$ gesucht mit

$$\|b - A\bar{x}\|_2 = \min_{x \in \mathbb{R}^n} \|b - Ax\|_2 \quad (4.3.2)$$

Das entspricht der „Lösung“ eines überbestimmten, i.A. nicht erfüllbaren GLS $Ax = b$. Aufgrund der l_2 -Norm ist \bar{x} gegeben durch die orthogonale Projektion von b auf den

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

Bildraum $R(A)$, wie gleich gezeigt wird.



Satz 4.3.3 (Projektionssatz). Sei V ein reeller Vektorraum mit einem Skalarprodukt $\langle \cdot, \cdot \rangle$ und der induzierten Norm $\|v\| := \sqrt{\langle v, v \rangle}$. Sei $U \subset V$ ein endlich dimensionaler Untervektorraum und sei

$$U^\perp := \{v \in V \mid \langle v, u \rangle = 0 \quad \forall u \in U\}$$

Dann gilt:

1) Zu jedem $v \in V$ existiert genau ein $\bar{u} \in U$, so dass $v - \bar{u} \in U^\perp$, d.h.

$$\langle v - \bar{u}, u \rangle = 0 \quad \forall u \in U$$

Dies definiert die **orthogonale Projektion**

$$P : V \rightarrow U, \quad v \mapsto \bar{u} = Pv$$

2) Zu jedem $v \in V$ bestimmt $P \cdot v$ die eindeutige Lösung

$$\|v - Pv\| = \min_{u \in U} \|v - u\|$$

Also gilt mit einem eindeutigen $\bar{u} = Pv$, dass

$$\|v - \bar{u}\| = \min_{u \in U} \|v - u\| \iff \langle v - \bar{u}, u \rangle = 0 \quad \forall u \in U \quad (4.3.3)$$

Beweis. 1) Sei $\{u_1, \dots, u_n\}$ eine Orthonormalbasis von U und $\bar{u} \in U$.

Daraus folgt:

$$\exists! (\alpha_i)_{i=1, \dots, n} \in \mathbb{R} : \bar{u} = \sum_{i=1}^n \alpha_i u_i$$

Damit gilt:

$$\begin{aligned}
 0 &= \langle v - \bar{u}, u \rangle & \forall u \in U \\
 \iff 0 &= \left\langle v - \sum_{i=1}^n \alpha_i u_i, u_j \right\rangle & \forall j = 1, \dots, n \\
 \iff \langle v, u_j \rangle &= \sum_{i=1}^n \alpha_i \langle u_i, u_j \rangle = \alpha_j
 \end{aligned}$$

Setze also

$$\begin{aligned}
 P \cdot v &= \bar{u} \\
 &= \sum_{i=1}^n \langle v, u_i \rangle u_i \in U
 \end{aligned} \tag{4.3.4}$$

dann ist \bar{u} die eindeutig bestimmte Lösung für $v - \bar{u} \in U^\perp$



Sei $u \in U$. Dann gilt:

$$\begin{aligned}
 \|v - u\|^2 &= \|v - \bar{u} + \bar{u} - u\|^2 \\
 &= \|v - \bar{u}\|^2 + \underbrace{\langle v - \bar{u}, \overbrace{\bar{u} - u}^{\in U} \rangle}_{=0} + \|u - \bar{u}\|^2 \\
 &= \|v - \bar{u}\|^2 + \|u - \bar{u}\|^2
 \end{aligned} \tag{4.3.5}$$

(Dies ist anschaulich der Satz des Pythagoras.)

□

Satz 4.3.4. Der Vektor $\bar{x} \in \mathbb{R}^n$ ist genau dann Lösung des linearen Ausgleichsproblems

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

falls er die Normalengleichung

$$A^T A \bar{x} = A^T b \tag{4.3.6}$$

erfüllt.

Insbesondere ist \bar{x} eindeutig, falls $A \in \mathbb{R}^{m \times n}$ maximalen Rang $n \leq m$ hat.

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

Beweis. Bezeichne $V = \mathbb{R}^m, U = R(A) = \{Ax \mid x \in \mathbb{R}^n\}, b \in \mathbb{R}^m$.
Nach (4.3.3) gilt:

$$\begin{aligned} \|b - A\bar{x}\|_2 &= \min_{x \in \mathbb{R}^n} \|b - Ax\|_2 \\ \Leftrightarrow \quad \langle b - A\bar{x}, Ax \rangle &= 0 \quad \forall x \in \mathbb{R}^n \\ \Leftrightarrow \quad \langle A^T(b - A\bar{x}), x \rangle &= 0 \quad \forall x \in \mathbb{R}^n \\ \Leftrightarrow \quad A^T(b - A\bar{x}) &= 0 \\ \Leftrightarrow \quad A^T A\bar{x} &= A^T b \end{aligned}$$

Nach dem Projektionssatz 4.3.3 existiert mindestens ein eindeutiges $\bar{y} = Pb$. Für dieses \bar{y} ist $\bar{x} \in \mathbb{R}^n$ mit $\bar{y} = A\bar{x}$ eindeutig bestimmt, falls A injektiv ist, d.h. falls $\text{rang}(A) = n$. \square

Ähnlich zum Skalarprodukt ist die relative Kondition von (P, b) schlecht, falls b fast senkrecht zu U steht. Die relative Kondition des linearen Ausgleichsproblems hängt zusätzlich von $\text{cond}(A)$ ab.

4.3.5 Lösung der Normalgleichung

Falls $\text{rang}(A) = n$, ist $A^T A$ spd und das Cholesky-Verfahren ist anwendbar.
Dafür ist

1. $A^T A$ zu berechnen:

Aufwand ca. $\frac{1}{2}n^2m$ Multiplikationen

Kondition häufig schlecht, da $\frac{1}{2}n^2$ Skalarprodukte berechnet werden

2. die Cholesky-Zerlegung von $A^T A$ durchzuführen:

Aufwand ca. $\frac{1}{6}n^3$ Multiplikationen

Kondition Für $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ und $\text{rang}(A) = n$ gilt:

$$\text{cond}_2(A^T A) = \text{cond}_2(A)^2 \quad (4.3.7)$$

(siehe Übungsaufgabe 19)

Also überwiegt für $m \gg n$ der Aufwand $A^T A$ zu berechnen. Die auftretenden Konditionen entsprechen i.d.R. nicht dem des Ausgangsproblems.

Damit ist die **Cholesky-Zerlegung für Normalgleichungen ungeeignet**.

Satz 4.3.6. Sei $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ und $\text{rang}(A) = n$, sei $b \in \mathbb{R}^m$ und besitze A eine Zerlegung

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

4.3 Lineare Ausgleichsprobleme

mit einer orthogonalen Matrix $Q \in \mathbb{R}^{m \times m}$ und einer oberen Dreiecksmatrix $R \in \mathbb{R}^{n \times n}$.
Dann ist R invertierbar.

Bezeichne

$$\begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} := Q^T \cdot b \quad (4.3.8)$$

dann ist

$$\bar{x} = R^{-1} \bar{b}_1 \quad (4.3.9)$$

die Lösung des linearen Ausgleichsproblems und

$$\begin{aligned} \|\bar{b}_2\| &= \|b - A\bar{x}\| \\ &= \min_{x \in \mathbb{R}^n} \|b - Ax\| \end{aligned}$$

Zur Erinnerung:

$$\begin{aligned} Q \text{ orthogonal} &: \Leftrightarrow QQ^T = I \\ &\Leftrightarrow Q^{-1} = Q^T \end{aligned}$$

Weiterhin ist Q längenerhaltend, d.h. $\|Qv\|_2 = \|v\|_2$ und somit folgt

$$\begin{aligned} \|Q\|_2 &= \|Q^{-1}\|_2 = 1 \quad \text{und} \\ \text{cond}_2(Q) &= 1 \end{aligned} \quad (4.3.10)$$

Beweis. R ist invertierbar, da

$$\begin{aligned} \text{rang}(R) &= \text{rang}(Q^{-1} \cdot A) \\ &= \text{rang}(A) \\ &= n \end{aligned}$$

Außerdem gilt:

$$\begin{aligned} \|b - Ax\|_2^2 &= \left\| Q \left(Q^T b - \begin{pmatrix} R \\ 0 \end{pmatrix} x \right) \right\|_2^2 \\ &= \left\| Q^T b - \begin{pmatrix} Rx \\ 0 \end{pmatrix} \right\|_2^2 \\ &\stackrel{Q \text{ längenerhaltend}}{=} \|\bar{b}_1 - Rx\|_2^2 + \|\bar{b}_2\|_2^2 \end{aligned}$$

wird minimal für $R\bar{x} = \bar{b}_1$

□

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

Da Q längenerhaltend ist, folgt mit 3.2.13 b) ($\text{cond}_A := \frac{\max\|Ax\|}{\min\|Ax\|}$) sofort:

$$\text{cond}_2(A) = \text{cond}_2(R)$$

Die auftretende Kondition entspricht also der des Ausgleichsproblems.

Bemerkung 4.3.7. Sei $A \in \mathbb{R}^{n \times n}$ invertierbar und habe eine QR-Zerlegung, d.h. es existiert eine orthogonale Matrix Q und eine obere Dreiecksmatrix R , so dass:

$$A = Q \cdot R$$

Dann kann das Gleichungssystem $Ax = b$ wie folgt gelöst werden:

1. Setze $z = Q^T b$, was Kondition 1 hat.
2. Löse durch Rückwärtssubstitution $Rx = z$.

4.4 Orthogonalisierungsverfahren

12.11.2014

Konstruiere eine QR-Zerlegung

$$A = Q \cdot \begin{pmatrix} R \\ 0 \end{pmatrix} \quad (4.4.1)$$

durch einen Eliminationsprozess:

$$A \rightarrow Q^{(1)} A \rightarrow Q^{(2)} Q^{(1)} A \rightarrow Q^{(p)} \dots Q^{(1)} A = \begin{pmatrix} R \\ 0 \end{pmatrix}. \quad (4.4.2)$$

mit orthogonalen Matrizen $Q^{(i)}$. Dann gilt

$$Q = Q^{(1)T} \dots Q^{(p)T} \quad (4.4.3)$$

Dies ist im Gegensatz zur LR -Zerlegung aufgrund von $\text{cond}(Q^{(i)}) = 1$ immer stabil.

Für $Q \in \mathbb{R}^{2 \times 2}$ gibt es zwei mögliche Anschauungen, nämlich:

- a) Drehung IMAGE MISSING
- b) Spiegelung IMAGE MISSING

4.4a) Givens-Rotation

Es wird eine Drehung auf den 1. Einheitsvektor durchgeführt:



Drehung auf den Einheitsvektor

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \rightarrow \begin{pmatrix} \alpha \\ 0 \end{pmatrix} = \alpha e_1$$

d.h. Elimination von a_2 mit

$$\|\alpha e_1\|_2 = \|a\|_2$$

Also gilt für α

$$\alpha = \pm \|a\|_2$$

Drehungen werden beschrieben durch

$$Q = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} =: \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \quad \theta \in [0, 2\pi)$$

und es muss gelten

$$Qa = \begin{pmatrix} \alpha \\ 0 \end{pmatrix}$$

Hiermit folgt für $\|a\| \neq 0$

$$\begin{aligned} c &= 1, s = 0 \\ c &= \frac{a_1}{\alpha}, s = \frac{a_2}{\alpha} \quad \text{mit} \\ \alpha &= \pm \sqrt{a_1^2 + a_2^2} \end{aligned} \tag{4.4.4}$$

für $\|a\| \neq 0$.

Im Folgenden wird dies kurz mit

$$[c, s] = \text{givens}(a_1, a_2)$$

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

bezeichnet.

Als **Givens-Rotation** wird eine Matrix der Form

$$\Omega_{k,l} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \mathbf{c} & & \mathbf{s} & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ & & & -\mathbf{s} & & \mathbf{c} & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \leftarrow k\text{-te Zeile} \\ \\ \\ \\ \leftarrow l\text{-te Zeile} \end{matrix} \quad (4.4.5)$$

mit $c^2 + s^2 = 1$ und $k < l$ bezeichnet.

Es folgt:

$$\begin{aligned} \Omega_{kl}A &= \tilde{A} \quad \text{mit} \\ \tilde{a}_{ij} &= a_{ij} \quad \text{für } i \neq k, l \\ \tilde{a}_{kj} &= ca_{kj} + sa_{lj} \\ \tilde{a}_{lj} &= -sa_{kj} + ca_{lj} \end{aligned}$$

Demnach werden nur die k -te und l -te Zeile verändert.

Falls nun $[c, s] = \text{givens}(x_k, x_l)$ gilt

$$\Omega_{k,l} \cdot x = \begin{pmatrix} x_1 \\ \vdots \\ x_{k-1} \\ \alpha \\ x_{k+1} \\ \vdots \\ x_{l-1} \\ 0 \\ x_{l+1} \\ \vdots \\ x_n \end{pmatrix} \quad \text{mit } \alpha = \pm \left\| \begin{pmatrix} x_k \\ x_l \end{pmatrix} \right\|_2 \quad (4.4.6)$$

d.h. eine Givens-Rotation erzeugt eine Null.

Da nun

$$\mathbb{R}^{m \times n} \ni \begin{pmatrix} * & * & * & \dots \\ 0 & * & & \\ 0 & 0 & \ddots & \\ \vdots & & & \\ 0 & 0 & 0 & \dots & * \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

gilt, sind

$$p = \sum_{j=1}^n (m - j)$$

Givens-Rotationen nötig, um eine QR-Zerlegung nach (4.4.1) zu erzeugen. Und eine Rotation, welche a_{ij} auf 0 setzt, ist durch zugehörige (c_{ij}, s_{ij}) gegeben.

Für eine 3x4-Matrix sieht das Verfahren folgendermaßen aus:

$$\begin{aligned}
 A = \begin{pmatrix} * & & & \\ * & & * & \\ * \swarrow & & & \\ * \swarrow & & & \end{pmatrix} &\xrightarrow{\Omega_{3,4}} \begin{pmatrix} * & & & \\ * \swarrow & & * & \\ * \swarrow & & & \\ 0 & & & \end{pmatrix} \xrightarrow{\Omega_{1,2}} \begin{pmatrix} * \swarrow & & & \\ * \swarrow & & & \\ 0 & & & \\ 0 & & & \end{pmatrix} \\
 \xrightarrow{\Omega_{2,3}} \begin{pmatrix} * & & & \\ 0 & & * & \\ 0 & & & \\ 0 & & & \end{pmatrix} &\xrightarrow{\Omega_{3,4}} \begin{pmatrix} * & * & & \\ 0 & * \swarrow & & * \\ 0 & * \swarrow & & \\ 0 & 0 & & \end{pmatrix} \xrightarrow{\Omega_{2,3}} \begin{pmatrix} * & * & * & \\ 0 & * & * & \\ 0 & 0 & * \swarrow & \\ 0 & 0 & * \swarrow & \end{pmatrix} \\
 \xrightarrow{\Omega_{3,4}} \begin{pmatrix} * & * & * & \\ 0 & * & * & \\ 0 & 0 & * & \\ 0 & 0 & 0 & \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}
 \end{aligned}$$

Es ergibt sich:

4.4.1 Givens-QR-Algorithmus

```

for  $j = 1, \dots, n$ 
|   for  $i = m, m - 1, \dots, j + 1$ 
|   |   % setze  $a_{ij}$  auf 0
|   |    $[c, s] = \text{givens}(a_{i-1,j}, \dots, a_{ij})$ 
|   |   speichere  $c$  und  $s$  für  $a_{ij}$ 
|   |    $A(i-1:j, j:h) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} * A(i-1:j, j:n)$ 
|   end
end

```

Bemerkung 4.4.2.

- a) $A(i-1:i, 1:j-1) = 0$ und ist daher nicht zu berechnen oder zu speichern. Der Speicherplatz kann für die Speicherung der Givensrotationen benutzt werden.

4 Lineare Gleichungssysteme: Direkte Methoden (Fortsetzung)

b) R steht anschließend in A .

c) Die Bestimmung der Länge $|\alpha|$ wird so ausgeführt, dass *overflow* oder *underflow* vermieden wird. Weiterhin wird das Vorzeichen von c oder s festgelegt, so dass aufgrund von $c^2 + s^2 = 1$ nur ein Wert ρ gespeichert werden muss. Hiermit wird auch das Vorzeichen von α festgelegt:

$a_2 = 0$: Setze $c = 1, s = 0, \alpha = a_1$, merke $\rho = 1$.

$|a_2| > |a_1|$: Setze $\tau = \frac{a_1}{a_2}, s = \frac{1}{\sqrt{1+\tau^2}}, c = s \cdot \tau, \alpha = a_2 \sqrt{1+\tau^2}$, merke $\rho = \frac{c}{s}$.

$|a_1| \geq |a_2|$: Setze $\tau = \frac{a_2}{a_1}, c = \frac{1}{\sqrt{1+\tau^2}}, s = c \cdot \tau, \alpha = a_1 \sqrt{1+\tau^2}$, merke $\rho = \frac{2}{s}$.

d) Aufgrund von c) muss nur ρ gespeichert werden:

$\rho = 1$: Setze $c = 1, s = 0$.

$|\rho| < 1$: Setze $c = 2\rho, s = \sqrt{1-c^2}$.

$|\rho| > 1$: Setze $s = \frac{2}{\rho}, c = \sqrt{1-s^2}$.

Hiermit können alle notwendigen Givens-Rotationen als untere Dreiecksmatrix zusammen mit R in A gespeichert werden.

4.4.3 Aufwand des Givens-QR-Algorithmus

a) $m \approx n$

→ ca. $\frac{4}{3}n^3$ Multiplikationen und $\frac{1}{2}n^2$ Quadratwurzeln nötig

Die Givens-QR-Zerlegung ist somit ungefähr viermal so aufwändig wie die Gauß-Elimination, dafür jedoch stabil.

b) $m \gg n$

→ ca. $2n^2m$ Multiplikationen und mn Quadratwurzeln nötig

Das Verfahren ist daher zwei- bis viermal so aufwändig wie das Cholesky-Verfahren für die Normalgleichungen, aber stabil.

c) Bei Hessenberg-Matrizen, d.h. Matrizen mit der Gestalt

$$A = \begin{pmatrix} * & \dots & * \\ * & * & \\ & \ddots & \ddots \\ 0 & & * \end{pmatrix}, \quad (4.4.7)$$

also $a_{ik} = 0 \forall i < k + 1$, sind nur $(n - 1)$ Givens-Rotationen auszuführen.

Diese Matrizen tauchen z.B. bei Eigenwertberechnungen auf und sind dort ein wichtiger Bestandteil der Verfahren.

4.4b) Householder-Reflexion

Es sei H eine Hyperebene im \mathbb{R}^m und zusätzlich ein Vektor $a \in \mathbb{R}^m$ gegeben.



Gesucht ist nun die Reflexion Q , so dass

$$Qa = \alpha e_1 = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{mit } \alpha = \pm \|a\|$$

Mit

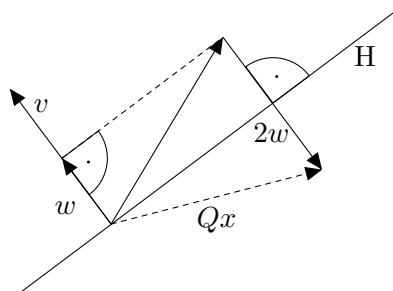
$$v = a - \alpha e_1 \quad (4.4.8)$$

gegeben, welches senkrecht zu H steht.

Damit Stellenauslöschungen in v , d.h. in v_1 , vermieden werden, wähle ein entsprechendes Vorzeichen für α , also

$$\alpha = -\text{sign}(a_1) \|a\|_2 \quad (4.4.9)$$

Die zugehörige Reflexion Q ist gegeben durch



wobei

$$w = \left\langle \frac{v}{\|v\|}, x \right\rangle \cdot \frac{v}{\|v\|}$$

$$Qx = x - 2w = x - 2 \frac{v^T x}{v^T v} v = \left(I - 2 \frac{vv^T}{v^T v} \right) x \quad (4.4.10)$$

$$Q = I - 2 \frac{vv^T}{v^T v} \quad vv^T \in \mathbb{R}^{n \times n}, \quad v^T v \in \mathbb{R} \quad (4.4.11)$$

und heißt **Householder Reflexion** (wurde 1958 von Householder eingeführt).
Für die spezielle Wahl (4.4.8) mit (4.4.9) vom Vektor v folgt

$$\begin{aligned} vv^T &= \|v\|^2 = \|a\|^2 - 2\alpha\langle a, e_1 \rangle + \alpha^2 \\ &= -2\alpha(a_1 - \alpha) \\ &= -2\alpha v \end{aligned} \quad (4.4.12)$$

Bemerkung 4.4.4.

- a) Q ist symmetrisch
- b) Q ist orthogonal
- c) Q ist involutorisch, d.h. $Q^2 = I$ (bzw. gilt $Q^{-1} = Q^T = Q$)

Die Householder Reflexion setzt nicht nur eine Null, sondern im Vektor gleich alle gewünschten Nullen

$$\begin{aligned} \textbf{Rotation:} \quad & \begin{pmatrix} * \\ * \\ * \end{pmatrix} \rightarrow \begin{pmatrix} * \\ * \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} * \\ 0 \\ 0 \end{pmatrix} \\ \textbf{Reflexion:} \quad & \begin{pmatrix} * \\ * \\ * \end{pmatrix} \rightarrow \begin{pmatrix} * \\ 0 \\ 0 \end{pmatrix} \end{aligned}$$

Um die erste Spalte in A auf die gewünschte Gestalt zu bringen, bestimme $Q^{(1)}$ wie oben, indem die erste Spalte als a gewählt wird:

$$A \rightarrow A^{(1)} = Q^{(1)} A = \begin{pmatrix} \alpha^{(1)} & & \\ 0 & * & \\ \vdots & & \\ 0 & & \end{pmatrix}$$

In der k -ten Spalte sollen nun die $(k-1)$ -ten Zeilen und die $(k-1)$ -ten Spalten bleiben und die Restmatrix verändert werden.

$$A^{(k-1)} = \begin{pmatrix} * & & & & \\ & \ddots & & & \\ & & * & & \\ & & 0 & | & - & - & - \\ 0 & \vdots & \vdots & | & T^{(k-1)} & & \\ & \vdots & \vdots & | & & & \\ & 0 & | & & & & \end{pmatrix} \begin{matrix} \leftarrow (k-1)\text{-te Zeile} \\ \\ \\ \uparrow (k-1)\text{-te Spalte} \end{matrix}$$

Setze also

$$Q^{(k)} = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \overline{Q}^{(k)} \end{pmatrix} \quad (4.4.13)$$

wobei $\overline{Q}^{(k)}$ durch die erste Spalte von $T^{(k)}$, d.h.

$$a = (a_{i,k}^{k-1})_{i=k,\dots,m} \in \mathbb{R}^{m+1-k} \quad (4.4.14)$$

bestimmt wird. Dann gilt

$$Q^{(k)} A^{(k-1)} = \begin{pmatrix} * & & & & & \\ & \ddots & & & * & \\ & & * & & - & - & - & - \\ & & 0 & | & * & & & \\ & 0 & \vdots & | & 0 & * & & \\ & & \vdots & | & \vdots & & \ddots & \\ & & 0 & | & 0 & & & * \end{pmatrix}$$

Nach insgesamt

$$p = \min(m-1, n) \quad (4.4.15)$$

Schritten erhalten wir für $A \in \mathbb{R}^{m \times n}$

$$Q^T A = Q^{(p)} \cdot \dots \cdot Q^{(1)} A = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad (4.4.16)$$

wobei Bemerkung 4.4.4 auch für $Q^T = Q^{(p)} \cdot \dots \cdot Q^{(1)}$ und somit auch für

$$Q = Q^{(1)} \cdot \dots \cdot Q^{(p)} \quad (4.4.17)$$

gilt.

4.4.5 Speicherung

17.11.2014

Gespeichert werden müssen die obere Dreiecksmatrix R und die **Householdervektoren** $v^{(i)} \in \mathbb{R}^{m+1-i}$. Die Diagonalelemente von R sind $r_{ii} = \alpha^{(i)}$.

Folgende Speicheraufteilung ist möglich:

$$A \rightarrow \left(\begin{array}{c|c|c|c|c} & & & & R \\ & & & & \\ v^{(1)} & | & & & \\ & v^{(2)} & | & & \\ & & & \ddots & \\ & & & & v^{(p)} \end{array} \right) \quad \text{und} \quad \begin{pmatrix} \alpha^{(1)} \\ \vdots \\ \alpha^{(p)} \end{pmatrix}$$

Wohlgedenkt kann so auch $A \in \mathbb{R}^{m \times n}$ mit $m < n$ bearbeitet werden, dann wird A zu:



Zerlegung einer $m \times n$ -Matrix für $m < n$

Falls zusätzlich $v^{(i)}$ so normiert ist, dass $v_1^{(i)} = 1$ ist, braucht diese Komponente nicht gespeichert werden und R kann komplett in A gespeichert werden.

4.4.6 Householder QR-Algorithmus

```

for  $j = 1, \dots, \min(m-1, n)$ 
| // berechne  $\alpha$  für  $a = A(j:m, j)$ 
|  $\alpha(j) = -\text{sign}(A(j, j))\sqrt{A(j:m, j)^T A(j:m, j)}$            siehe (4.4.9)
| // berechne  $A(j:m, j) = v = a - \alpha e_1$ 
|  $A(j, j) = A(j, j) - \alpha(j)$                                    siehe (4.4.8)
| // berechne  $-\langle v, v \rangle = \alpha(a_1 - \alpha) = \alpha v_1$ 
|  $\beta = \alpha(j)A(j, j)$                                        siehe (4.4.12)
| // berechne  $\overline{Q}^{(j)} T^{(j)}$  aber nicht mehr die erste Spalte, welche  $\alpha(j)e_1$  ist
| for  $l = j+1 : n$ 
| | // setze  $v_x = -2(v^T x)(v^T v)^{-1}$ 
| |  $v_x = A(j:m, j)^T A(j:m, l) \cdot \frac{1}{\beta}$                    siehe (4.4.10)
| | // berechne  $\overline{Q}^{(j)} x = x + v_x \cdot v$ 
| |  $A(j:m, l) = A(j:m, l) + v_x A(j:m, l)$                  siehe (4.4.10)
| end
end
    
```

4.4.7 Berechnung von $Q^T b$

Zur Lösung eines Gleichungssystems oder eines linearen Ausgleichsproblems muss noch $Q^T b = Q^{(p)} \dots Q^{(1)}$ berechnet werden.


```

for  $j = 1, \dots, \min\{m-1, n\}$ 
|   // beachte (4.4.13), also  $b(1:j-1)$  bleibt gleich
|   // setze  $v_x = -2(v^T b)(v^T v)^{-1}$ 
|    $v_x = (A(j:m, j)^T b(j:m)) \cdot (\alpha(j)A(j, j))^{-1}$ 
|   // berechne  $\overline{Q}^{(j)} b = b + v_x v$ 
|    $b(j:m) = b(j:m) + v_x A(j:m, j)$ 
end

```

4.4.8 Aufwand für den Householder-QR-Algorithmus

- a) Falls $m \approx n$ sind ungefähr $\frac{2}{3}n^3$ Multiplikationen notwendig und ist somit ungefähr doppelt so teuer wie die LR-Zerlegung, ist aber stabil.
- b) Falls $m \gg n$ sind ungefähr $2mn^2$ Multiplikationen notwendig. Der Aufwand ist daher ungefähr so hoch wie beim Cholesky-Verfahren für Normalgleichungen, aber stabil.

5 Numerische Lösung nichtlinearer Gleichungssysteme

Beispiel:

linear: $Ax = b$

nichtlinear: $f(x) = \sin(x) + x^3 - 4 = 0$

5.1 Einführung

Beispiel 5.1.1.

1) $f(x) = x^2 - c = 0 \Leftrightarrow x = \pm\sqrt{c}$:
Berechnung der Wurzel

2) Sei p ein Polynom:
Nullstellenbestimmung

3) Löse das nichtlineare Randwertproblem

$$-\Delta u = f(u)$$

in $\Omega = (0, 1)^2$ mit $u = 0$ auf $\partial\Omega$.

Mit dem Differenzenverfahren¹ ergibt sich

$$A\vec{u} = h^2 \vec{f}(\vec{u})$$

ein System nichtlinearer Gleichungen.

Nullstellenbestimmung

Gegeben $D \subseteq \mathbb{R}^n, f : D \rightarrow \mathbb{R}^m$ stetig

Gesucht $x^* \in D$ mit $f(x^*) = 0$

¹s. Übungsaufgabe 2)

Fixpunktgleichung

Gegeben $D \subseteq \mathbb{R}^n, g: D \rightarrow \mathbb{R}^n$ stetig

Gesucht $x^* \in D$ mit $g(x^*) = x^*$

Falls $m=n$ ist dies äquivalent zur Nullstellenbestimmung.

5.1.2 Das Bisektionsverfahren

Sei $f: [a, b] \rightarrow \mathbb{R}$ stetig und $f(a) \cdot f(b) < 0$.

Dann folgt aus dem Zwischenwertsatz die Existenz mindestens einer Nullstelle $x^* \in (a, b)$.



Beispiel zur Nullstellensexistenz

Generiere eine Folge von Intervallen $[a^{(i)}, b^{(i)}] \in [a^{(i-1)}, b^{(i-1)}]$, die eine Nullstelle enthalten und mit $b^{(i)} - a^{(i)} \rightarrow 0$. Definiere

$$x^{(i+1)} = \frac{1}{2}(b^{(i)} + a^{(i)}) \quad (5.1.1)$$

und

$$[a^{(i+1)}, b^{(i+1)}] := \begin{cases} [x^{(i+1)}, b^{(i)}] & \text{für } f(a^{(i)}) \cdot f(x^{(i+1)}) > 0 \\ [a^{(i)}, x^{(i+1)}] & \text{für } f(a^{(i)}) \cdot f(x^{(i+1)}) < 0 \end{cases} \quad (5.1.2)$$

Für jedes $i \geq 1$ gilt somit

$$b^{(i)} - a^{(i)} = \frac{1}{2^i}(b - a)$$

und es existiert eine Nullstelle x^* in $[a^{(i)}, b^{(i)}] \in [a^{(i-1)}, b^{(i-1)}]$ für alle i .
Damit folgt

$$\begin{aligned} |x^{(i-1)} - x^*| &\leq \frac{1}{2}(b^{(i)} - a^{(i)}) \\ &= 2^{-(i+1)}(b - a) \rightarrow 0 \end{aligned}$$

Also $\lim_{i \rightarrow \infty} x^{(i)} = x^*$.

Korollar 5.1.3. Das oben angegebene Bisektionsverfahren konvergiert, falls $f : [a, b] \rightarrow \mathbb{R}$ stetig ist und $f(a) \cdot f(b) < 0$ gilt.

Bemerkung 5.1.4.

- a) $x^{(i)}$ wird als Intervallmitte, also unabhängig von $f(x^{(i)})$ gewählt. die Konvergenzgeschwindigkeit hängt von der Länge des Intervalls $[a, b]$ ab und der Lage von x^* bezüglich der Intervallhalbierung ab.
Die Konvergenz kann demnach sehr langsam sein.
- b) Ein Vorteil ist, dass keine Differenzierbarkeitsvoraussetzungen nötig sind.
- c) Das Verfahren ist nicht für $f : D \rightarrow \mathbb{R}^n$ anwendbar.

5.2 Fixpunktiteration

Gesucht sei ein Fixpunkt $x^* \in D \subseteq \mathbb{R}^n$ der stetigen Funktion $g : D \rightarrow \mathbb{R}^n$, d.h.

$$x^* = g(x^*) \quad (5.2.1)$$

Idee: Nutze (5.2.1) zur Iteration, d.h. wähle $x^{(0)} \in D$, setze

$$x^{(k+1)} = g(x^{(k)}) \quad \text{für } k \in 0, 1, \dots \quad (5.2.2)$$

Es bedarf noch der Voraussetzung, dass $x^{(k)} \in D \quad \forall k$

Falls $x^{(k)}$ konvergiert, ist der Grenzwert x^* ein Fixpunkt, denn für stetiges g gilt:

$$\begin{aligned} x^* &= \lim_{k \rightarrow \infty} x^{(k+1)} = \lim_{k \rightarrow \infty} g(x^{(k)}) \\ &\stackrel{g \text{ stetig}}{=} g\left(\lim_{k \rightarrow \infty} x^{(k)}\right) = g(x^*) \end{aligned} \quad (5.2.3)$$

Beispiel 5.2.1. Löse $x - e^{-x} - 1 = 0$.

- a) $x = 1 + e^{-x} =: g_1(x)$



Konvergenz der Fixpunktiteration für $x = 1 + e^{-x}$

→ Konvergenz

- b) $e^{-x} = x - 1 \Leftrightarrow x = -\ln(x - 1) =: g_2(x)$



Versagen der Fixpunktiteration für $x = -\ln(x-1)$

$\longrightarrow g(x^{(2)})$ nicht definiert!

Definition 5.2.2. Sei $D \subseteq \mathbb{R}^n$ abgeschlossen und $\|\cdot\|$ eine Norm auf dem \mathbb{R}^n . Eine Abbildung $g : D \rightarrow \mathbb{R}^n$ heißt **Kontraktion** bezüglich $\|\cdot\|$, falls es ein $\kappa \in [0, 1)$ gibt mit

$$\|g(u) - g(v)\| \leq \kappa \|u - v\| \quad \forall u, v \in D$$

Die kleinste solche Zahl κ heißt Kontraktionszahl von g .



Grafische Veranschaulichung einer Kontraktion

Offensichtlich ist jede auf D kontrahierende Abbildung stetig.

Lemma 5.2.3. Sei $D = \overline{\Omega}$ mit $\Omega \subseteq \mathbb{R}^n$ offen und konvex und $\|\cdot\|$ eine Norm auf dem \mathbb{R}^n . Falls $g : D \rightarrow \mathbb{R}^n$ eine stetig differenzierbare Funktion ist und bezüglich der zugeordneten Matrixnorm $\sup_{x \in \Omega} \|Dg(x)\| < 1$ gelte, so ist g kontrahierend bezüglich $\|\cdot\|$.

Beweis. Mit $u, v \in D$ gilt $u + t(v - u) \in D$, da D konvex ist.

Somit ist $h : [0, 1] \rightarrow \mathbb{R}^n$ mit $h(t) := g(u + t(v - u))$ wohldefiniert und stetig differenzierbar. Mit dem Hauptsatz der Differenzial- und Integralrechnung folgt:

$$\begin{aligned} \|g(u) - g(v)\| &= \|h(1) - h(0)\| \\ &= \left\| \int_0^1 h'(t) dt \right\| \\ &= \left\| \int_0^1 Dg(u + t(v - u)) \cdot (v - u) dt \right\| \\ &\leq \int_0^1 \|Dg(u + t(v - u))\| dt \cdot \|v - u\| \\ &\leq \underbrace{\sup_{x \in \Omega} \|Dg(x)\|}_{=: \kappa} \cdot \|v - u\| \end{aligned} \tag{5.2.4}$$

□

Satz 5.2.4 (Banachscher Fixpunktsatz). Sei $D \subset \mathbb{R}^n$ abgeschlossen und die Abbildung $g : D \rightarrow \mathbb{R}^n$ eine Kontraktion.

Dann gilt:

1) Es existiert genau ein Fixpunkt x^* von g .

2) Für jeden Startwert $x^{(0)} \in D$ konvergiert die Folge der Fixpunktiterierten

$$x^{(k+1)} := g(x^{(k)}) \xrightarrow{k \rightarrow \infty} x^* \quad (5.2.5)$$

3) Es gelte die a posteriori Fehlerabschätzung

$$\|x^{(k)} - x^*\| \leq \frac{\kappa}{1 - \kappa} \|x^{(k)} - x^{(k-1)}\| \quad (5.2.6)$$

und die a priori Fehlerabschätzung

$$\|x^{(k)} - x^*\| \leq \frac{\kappa^k}{1 - \kappa} \|x^{(1)} - x^{(0)}\| \quad (5.2.7)$$

19.11.2014

Beweis. zu 2) Sei $x_0 \in D$ beliebig. (5.2.5) ist wohldefiniert, da $g(D) \subset D$. $(x^{(k)})_{k \in \mathbb{N}}$ bilden eine Cauchyfolge, da

$$\begin{aligned} \|x^{(k+1)} - x^{(k)}\| &= \|g(x^{(k)}) - g(x^{(k-1)})\| \\ &\leq \kappa \|x^{(k)} - x^{(k-1)}\| \\ &\leq \kappa^k \|x^{(1)} - x^{(0)}\| \\ \implies \|x^{(k+l)} - x^{(k)}\| &\leq \sum_{i=0}^l \|x^{(k+i+1)} - x^{(k+i)}\| \\ &\leq \sum_{i=0}^l \kappa^{k+1} \|x^{(1)} - x^{(0)}\| \\ &\leq \frac{\kappa^k}{1 - \kappa} \|x^{(1)} - x^{(0)}\| \quad \forall l \in \mathbb{N} \end{aligned}$$

Daraus folgt, dass $\lim_{k \rightarrow \infty} x^{(k)} = x^*$ existiert und $x^* \in D$, da D abgeschlossen ist und somit vollständig.

zu 1) Da g stetig ist, ist $g(x^*) = x^*$ (siehe hierzu (5.2.3)).

x^* ist eindeutiger Fixpunkt, da für einen weiteren Fixpunkt y^* gilt

$$0 \leq \|x^* - y^*\| = \|g(x^*) - g(y^*)\| \leq \kappa \|x^* - y^*\|$$

Da $\kappa < 1$, muss $\|x^* - y^*\| = 0$ sein und damit $x^* = y^*$.

zu 3) Betrachte

$$\begin{aligned} \|x^* - x^{(k)}\| &= \lim_{l \rightarrow \infty} \|x^{(k+l)} - x^{(k)}\| \\ &\leq \frac{\kappa^k}{1 - \kappa} \|x^{(1)} - x^{(0)}\| \end{aligned}$$

bzw.

$$\begin{aligned} \lim_{l \rightarrow \infty} \|x^{(k+l)} - x^{(k)}\| &\leq \lim_{l \rightarrow \infty} \sum_{i=0}^{l-1} \|x^{(k+i+1)} - x^{(k+i)}\| \\ &\leq \lim_{l \rightarrow \infty} \sum_{i=0}^{l-1} \kappa^{i+1} \|x^{(k)} - x^{(k-1)}\| \\ &\leq \frac{\kappa}{1 - \kappa} \|x^{(k)} - x^{(k-1)}\| \end{aligned}$$

□

Bemerkung 5.2.5.

- 1) Als Voraussetzung wäre bereits ausreichend:
 D ist vollständiger metrischer Raum mit Metrik d .
 Dann ersetze die Norm durch die Metrik d .
- 2) Im Allgemeinen ist der Nachweis $g(D) \subset D$ schwierig.

Folgerung 5.2.6. Sei $x^* \in \mathbb{R}^n$, so dass $g(x^*) = x^*$ und sei g in einer Umgebung von $\overline{B_\varepsilon(x^*)} = \{x \in \mathbb{R}^n \mid \|x - x^*\| \leq \varepsilon\}$ stetig differenzierbar und es gelte $\|g'(x)\| < 1$ für $x \in B_\varepsilon(x^*)$, so ist Satz 5.2.4 mit $D = \overline{B_\varepsilon(x^*)}$ anwendbar.

Beweis. Nutze (5.2.4) und Lemma 5.2.3.

□

5.3 Konvergenzordnung und Fehlerabschätzungen

Definition 5.3.1. Eine Folge $(x^{(k)})_{k \in \mathbb{N}}$ mit $x^{(k)} \in \mathbb{R}^n$ **konvergiert** mit (mindestens) der **Ordnung** $p \geq 1$ gegen x^* , falls

$$\lim_{k \rightarrow \infty} x^{(k)} = x^*$$

und falls es ein $C > 0$ sowie $N \in \mathbb{N}$ gibt, so dass

$$\|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^p \quad \forall k \geq N$$

5.3 Konvergenzordnung und Fehlerabschätzungen

Im Fall $p = 1$ ist zusätzlich $C < 1$ und man spricht von **linearer Konvergenz**.

Für $p = 2$ heißt es **quadratische Konvergenz**.

Gilt

$$\lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0,$$

so konvergiert die Folge **superlinear**.

Bemerkung 5.3.2. Die Fixpunktiteration konvergiert unter der Voraussetzung in 5.2.4 mindestens linear.

Bemerkung 5.3.3.

- a) lineare Konvergenz hängt von der gewählten Norm ab.
- b) Hat die Folge bzgl. einer Vektornorm auf dem \mathbb{R}^n die Konvergenzordnung $p > 1$, hat sie diese bzgl. jeder Norm.

Definition 5.3.4. a) Ein iteratives Verfahren zur Bestimmung eines Wertes x^* hat die Konvergenzordnung p , falls es eine Umgebung U um x^* gibt, so dass für alle Startwerte aus $U \setminus \{x^*\}$ die erzeugte Folge mit Ordnung p konvergiert.

b) Das Verfahren heißt **lokal konvergent**, falls es für alle Startwerte in einer Umgebung von x^* konvergiert.

c) Das Verfahren heißt **global konvergent**, falls es im gesamten Definitionsbereich des zugehörigen Problems konvergiert.

Lemma 5.3.5. Sei $(x^{(k)})_{k \in \mathbb{N}}$ eine konvergente Folge in \mathbb{R} mit Grenzwert x^* .

a) Falls

$$\lim_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = A \in (-1, 1), \quad A \neq 0 \quad (5.3.1)$$

hat die Folge genau die Konvergenzordnung 1. Weiter gilt mit $A_k = \frac{x^{(k)} - x^{(k-1)}}{x^{(k-1)} - x^{(k-2)}}$

$$\lim_{k \rightarrow \infty} \frac{A_k}{1 - A_k} \cdot \frac{x^{(k)} - x^{(k-1)}}{x^* - x^{(k)}} = 1 \quad (5.3.2)$$

$$\lim_{k \rightarrow \infty} A_k = A$$

b) Falls die Folge Konvergenzordnung $p > 1$ hat, gilt

$$\lim_{k \rightarrow \infty} \frac{x^{(k)} - x^{(k-1)}}{x^* - x^{(k)}} = 1 \quad (5.3.3)$$

zu Def. 5.3.1: Im Fall $p = 1$ ist zusätzlich $C < 1$ verlangt.

5 Numerische Lösung nichtlinearer Gleichungssysteme

Beweis. (skizzenhaft, siehe Übungsaufgaben)

Sei $e^{(k)} := x^* - x^{(k)}$.

Nutze $x^{(k+1)} - x^{(k)} = e^{(k)} - e^{(k+1)}$:

a) Zeige

$$\lim_{k \rightarrow \infty} \frac{x^{(k)} - x^{(k-1)}}{e^{(k)}} = \frac{1-A}{A},$$

sowie

$$\lim_{k \rightarrow \infty} A_k = A,$$

so folgt die Behauptung.

b) Folgt aus $\lim_{k \rightarrow \infty} \frac{e^{(k+1)}}{e^{(k)}} = 0$.

□

Folgerung 5.3.6 (a posteriori Fehlerabschätzung).

a) Für $p = 1$ gilt

$$x^* - x^{(k)} \approx \frac{A_k}{1 - A_k} (x^{(k)} - x^{(k-1)}) \quad (5.3.4)$$

für große k und A_k in etwa konstant.

$|x^{(k)} - x^{(k-1)}|$ ist im Allgemeinen **keine** sinnvolle Schätzung des Fehlers $|x^* - x^{(k)}|$!

b) Für $p > 1$ gilt:

$$x^* - x^{(k)} \approx x^{(k+1)} - x^{(k)} \quad (5.3.5)$$

für große k .

Bemerkung 5.3.7. Für Folgen im \mathbb{R}^n gibt es für $p = 1$ kein Analogon zu (5.3.4). Falls $p > 1$, lässt sich (5.3.3) für die Normen der Differenzen zeigen, d.h.

$$\|x^* - x^{(k)}\| \approx \|x^{(k+1)} - x^{(k)}\| \quad (5.3.6)$$

Beweis. Nutze $\lim_{k \rightarrow \infty} \frac{\|e^{(k+1)}\|}{\|e^{(k)}\|} = 0$ und

$$\|e^{(k)}\| - \|e^{(k+1)}\| \leq \|x^{(k+1)} - x^{(k)}\| \leq \|e^{(k)}\| + \|e^{(k+1)}\|.$$

□

Folgerung 5.3.8. Falls $p > 1$ ist, kann p folgendermaßen approximiert werden:

$$p \approx \frac{\log(\|x^{(k+2)} - x^{(k+1)}\|)}{\log(\|x^{(k+1)} - x^{(k)}\|)}$$

Beweis. Siehe Übungsaufgabe.

□

5.4 Newton-Verfahren für skalare Gleichung

Sei $f : [a, b] \rightarrow \mathbb{R}$ differenzierbar. Dann gilt

$$f(x^*) = f(x) + f'(x)(x^* - x) + o(\|x - x^*\|),$$

d.h. f kann lokal gut durch die Tangente approximiert werden.

Betrachte die Nullstellengleichung $f(x^*) = 0$



Veranschaulichung des Newton-Verfahrens an einem Funktionsgraphen

und bestimme iterativ die Nullstelle der Tangentengleichung

$$0 = f(x) + f'(x)(\bar{x} - x) \Leftrightarrow \bar{x} = x - \frac{f(x)}{f'(x)}$$

Notwendig ist hier die Bedingung $f'(x) \neq 0$.

5.4.1 Iterationsschritt des Newton(-Kantorowitsch)-Verfahrens

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \quad (5.4.1)$$

wird auch Tangentenverfahren genannt und stammt von J. Raphson (1630). Newton hat eine ähnliche Technik früher angewendet.

Satz 5.4.2. Sei $f \in C^1(a, b)$ und $x^* \in (a, b)$ eine einfache Nullstelle von f , d.h. $f'(x^*) \neq 0$. Dann gibt es ein $\varepsilon > 0$, s.d. für jedes $x^{(0)} \in \overline{B}_\varepsilon(x^*)$ das Newton-Verfahren (5.4.1) superlinear gegen x^* konvergiert.

Falls $f \in C^2(a, b)$ tritt mindestens quadratische Konvergenz ein, d.h. das Verfahren konvergiert lokal quadratisch.

Beweis. Gleichung (5.4.1) definiert eine Fixpunktiteration mit $g(x) = x - \frac{f(x)}{f'(x)}$.

Für $f \in C^2(a, b)$ gilt

$$g'(x) = 1 - \frac{f'f' - ff''}{(f')^2}(x) = \frac{f(x)f''(x)}{(f'(x))^2}.$$

5 Numerische Lösung nichtlinearer Gleichungssysteme

Da $f(x^*) = 0$ und $f'(x^*) \neq 0$ gilt $g'(x^*) = 0$.

Weiterhin gibt es eine Umgebung U_0 von x^* , in der $f(x) \neq 0 \forall x \in U_0$, da f stetig ist.

In U_0 ist somit g' stetig. Da $g'(x^*) = 0$ ist, existiert ein $\varepsilon > 0$ mit

$$g'(x) \leq \kappa < 1 \quad \forall x \in \overline{B_\varepsilon(x^*)}.$$

Da $g(x^*) = x^*$ ist, ist die Folgerung 5.2.6 anwendbar, also ist g eine Kontraktion. und $g(\overline{B_\varepsilon(x^*)}) \subset \overline{B_\varepsilon(x^*)}$. Der Banachsche Fixpunktsatz liefert Konvergenz für alle $x^{(0)} \in \overline{B_\varepsilon(x^*)}$.

Die quadratische Konvergenz folgt aus

$$\begin{aligned} |x^{(k-1)} - x^*| &= \left| x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} - x^* + \frac{f(x^*)}{f'(x^*)} \right| \\ &= \frac{|f(x^{(k)}) - f(x^*) + f'(x^{(k)})(x^{(k)} - x^*)|}{|f'(x^{(k)})|} \\ &\leq \sup_{x \in \overline{B_\varepsilon(x^*)}} \frac{1}{|f'(x^*)|} \cdot \sup_{x \in \overline{B_\varepsilon(x^*)}} |f''(x)| \cdot \frac{1}{2} |x^{(k)} - x^*|^2 \end{aligned}$$

aufgrund der Taylorentwicklung und da $x^{(k-1)} \in \overline{B_\varepsilon(x^*)} \forall k \in \mathbb{N}$ (da g Kontraktion).

Für $f \in C^1$ siehe [HH94]. □

Bemerkung 5.4.3.

- Mehrfache Nullstellen könne im Allgemeinen nicht mit (5.4.1) bestimmt werden.
- Die Ableitung f' muss analytisch (als Funktion) gegeben sein.
- Die Lage und Größe des Konvergenzinterfalls ist a priori unbekannt. (Hierfür könnte z.B. das Bisektionsverfahren Anwendung finden.)

24.11.2014

Beispiel 5.4.4 (Newton-Verfahren ohne Konvergenz).

- $x^{(1)}$ nicht mehr im Definitionsbereich



Fehlschlagen des Newton-Verfahren: außerhalb des Definitionsbereichs

- $|x^* - x^{(1)}| \not\leq |x^* - x^{(0)}|$



Fehlschlagen des Newton-Verfahren: Konvergenz nicht gesichert

5.4.5 Newton-Verfahren: Iterativer Linearisierungsprozess

Die entscheidende Idee beim Newton-Verfahren ist der **iterative Linearisierungsprozess**, d.h. die Lösung einer nichtlinearen Gleichung wird durch eine Folge von Lösungen linearer Gleichungen ersetzt.

Beispiel 5.4.6. Es ist die Lösung von $x - e^{-\frac{1}{2}x} = 0$ mit $x^{(0)} = 0,8$ gesucht.

a) Mit der Banachschen Fixpunktiteration angewendet auf $x = e^{(-\frac{1}{2}x)}$ ergibt sich

$$x^{(10)} = \mathbf{0,70347017} \quad \text{auf 4 Stellen exakt}$$

b) Mit dem Newton-Verfahren

$$x^{(3)} = 0,70346742 \quad \text{bis auf 17 Stellen exakt}$$

$$x^{(4)} \quad \text{bis auf Maschinengenauigkeit exakt}$$

Die Ableitung $f'(x)$ ist nicht immer explizit bekannt.

Eine Idee ist, sie zu approximieren mithilfe des Differenzenquotienten:

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

Damit ergibt sich

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

d.h. $x^{(k+1)}$ ist die Nullstelle der Sekante durch $f(x^k)$ und $f(x^{(k-1)})$.

5.4.7 Iterationsschritt des Sekantenverfahrens

$$x^{(k+1)} = \frac{x^{(k-1)} f(x^{(k)}) - x^{(k)} f(x^{(k-1)})}{f(x^{(k)}) - f(x^{(k-1)})} \quad (5.4.2)$$



Geometrische Veranschaulichung des Sekantenverfahrens

Satz 5.4.8 (Konvergenz des Sekantenverfahrens). Sei $f \in C^2([a, b])$ und $x^* \in (a, b)$ eine einfache Nullstelle.

Dann konvergiert das Sekantenverfahren in einer Umgebung von x^* superlinear mit Ordnung

$$p = \frac{1}{2}(1 + \sqrt{5}) = 1,618.$$

Beweis. Siehe z.B. [HH94; SB90, Zwischenwertsatz, Fibonacci-Folge] □

zu Beispiel 5.4.6: Das Sekantenverfahren benötigt einen zweiten Startwert, z.B.

$$\begin{aligned} x^{(1)} &= 0,7 \\ \Rightarrow x^{(3)} &= 0,7034674 && \text{auf 7 Stellen exakt} \\ x^{(6)} & && \text{bis auf Maschinengenauigkeit exakt} \end{aligned}$$

Bemerkung 5.4.9.

- a) Das Verfahren ist keine Fixpunktiteration. Es benötigt $x^{(k)}$ und $x^{(k-1)}$ für $x^{(k+1)}$ (**Mehrschrittverfahren**)
- b) Die Berechnung von $f(x)$ und $f'(x)$ ist im Allgemeinen sehr teuer. Das Sekanten-Verfahren benötigt pro Iteration nur eine Funktionsauswertung, das Newton-Verfahren hingegen zwei.
Also sind zwei Iterationen des Sekanten-Verfahrens so teuer wie eine des Newton-Verfahrens.
Bei gleichem Aufwand konvergiert das Sekanten-Verfahren daher lokal schneller mit der Konvergenzordnung

$$p^2 = 2,618 \dots$$

für $x^{(k)} \rightarrow x^{(k+2)}$ als das Newton-Verfahren (siehe auch Beispiel 5.4.6).

Beispiel: Sei $f : \mathbb{R}^n \rightarrow \mathbb{R}$ und $f(x)$ die erste Komponente von $A^{-1}x$. Diese n-dimensionale Funktionsauswertung benötigt $\mathcal{O}(n^3)$ flops.

- c) Die Sekantenmethode ist i.A. nicht stabil, denn für $f(x^{(k)}) \approx f(x^{(k+1)})$ können Stellenauslöschungen im Nenner auftreten.
Stabilere Varianten, wie z.B. **regula falsi**, haben eine geringere Konvergenzordnung.

5.5 Das Newton-Verfahren im Mehrdimensionalen

Wie im 1-dimensionalen wird $f : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$ linearisiert

$$f(\bar{x}) \approx f(x) + Df(x)(\bar{x} - x) \quad (5.5.1)$$

mit

$$Df(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \dots & \frac{\partial f_n}{\partial x_n}(x) \end{pmatrix} \quad (\text{genannt: die Jacobi-Matrix von } f)$$

Falls nun die Jacobi-Matrix $Df(x)$ invertierbar ist und $f(\bar{x}) = 0$ gilt, folgt

$$\bar{x} = x - [Df(x)]^{-1} \cdot f(x)$$

5.5.1 Iterationsschritt des Newton-Verfahrens

$$x^{(k+1)} = x^{(k)} - [Df(x^{(k)})]^{-1} \cdot f(x^{(k)}) \quad (5.5.2)$$

5.5.2 Newton-Verfahren

```

setze Startwert  $x$ 
 $i = 0$ 
 $fx = f(x)$ 
while „Abbruchkriterium“
|    $Dfx = Df(x)$ 
|   Löse2  $Dfx \cdot d = -fx$ 
|    $x = x + d$ 
|    $fx = f(x)$ 
|    $i = i + 1$ 
end

```

Bemerkung 5.5.3. Ein Newton-Iterationsschritt (5.5.2) wird also aufgeteilt in Berechnung der sogenannten **Newton-Korrektur**

$$Df(x^{(k)})\Delta x^{(k)} = -f(x^{(k)}) \quad (5.5.3)$$

und dem **Korrekturschritt**

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)} \quad (5.5.4)$$

²entspricht $Ax = b$

5.5.4 Aufwand pro Iteration

n eindimensionale Funktionsauswertungen für $f(x)$

n^2 eindimensionale Funktionsauswertungen für $Df(x)$

26.11.2014

$\mathcal{O}(n^2)$ flops (i.d.R.) zum Lösen eines GLS

Bemerkung 5.5.5. Das Newton-Verfahren ist **affin-invariant**, d.h. die Folge $(x^{(k)})$ ist zu gegebenem $x^{(0)}$ unabhängig davon, ob $f(x) = 0$ oder $\tilde{f}(x) := A \cdot f(x) = 0$ mit regulärem $A \in \mathbb{R}^{n \times n}$ gelöst wird. Dies gilt, da

$$\begin{aligned} [D\tilde{f}(x)]^{-1} \cdot \tilde{f}(x) &= [A \cdot Df(x)]^{-1} \cdot (A \cdot f(x)) \\ &= [Df(x)]^{-1} \cdot f(x) \end{aligned}$$

und damit ist die Newton-Korrektur $\Delta x^{(k)}$ affin-invariant.

Satz 5.5.6. Sei $\Omega \subset \mathbb{R}^n$ offen und $f : \Omega \rightarrow \mathbb{R}^n$ in $C^2(\Omega)$. Sei $x^* \in \Omega$ eine Nullstelle f mit einer invertierbaren Jacobi-Matrix $Df(x^*)$. Dann existiert eine Umgebung von x^* , so dass das Newton-Verfahren für jeden Startwert $x^{(0)}$ in dieser Umgebung quadratisch gegen x^* konvergiert.

Beweis. Kann wie im eindimensionalen durchgeführt werden.

Aber Vorsicht: $D^2 f(x)$ ist eine **bilineare Abbildung** in $\mathcal{L}(\mathbb{R}^n, \mathcal{L}(\mathbb{R}^n, \mathbb{R}^n))$.

Wir zeigen die Behauptung induktiv über die quadratische Konvergenz.

Da $Df(x^*)$ invertierbar ist und $f \in C^2(\Omega)$, existiert nach dem Satz über implizite Funktionen eine Umgebung $\overline{B_\varepsilon(x^*)} \subset \Omega$, auf der $Df(x)$ invertierbar und stetig ist.

Sei

$$c := \sup_{x \in B_\varepsilon(x^*)} \|[Df(x)]^{-1}\|$$

und

$$w := \sup_{x \in B_\varepsilon(x^*)} \|D^2 f(x)\|$$

Für $x^{(k)} \in B_\varepsilon(x^*)$ ist $x^{(k)} + t(x^* - x^{(k)}) \in B_\varepsilon(x^*)$ für $t \in [0, 1]$ und

$$h^{(k)}(t) := f(x^{(k)} + t(x^* - x^{(k)})) \quad \forall t \in [0, 1]$$

ist wohldefiniert und in $C^2([0, 1], \mathbb{R}^n)$.

Wie in 5.4.2 folgt

$$\begin{aligned} x^{(k+1)} - x^* &= [Df(x^{(k)})]^{-1} \left(f(x^*) - f(x^{(k)}) - Df(x^{(k)})(x^* - x^{(k)}) \right) \\ &= [Df(x^{(k)})]^{-1} \left(h^{(k)}(1) - h^{(k)}(0) - Dh^{(k)}(0) \cdot 1 \right) \\ &= [Df(x^{(k)})]^{-1} \int_0^1 D^2 h^{(k)}(1-t) dt \end{aligned} \quad () \text{Restglieddarst. der Taylorentv}$$

5.5 Das Newton-Verfahren im Mehrdimensionalen

Das Ziel ist nun zu zeigen, dass $\|x^{(k+1)} - x^*\| \leq c \cdot \|x^{(k)} - x^*\|^2$. Mit den Definitionen von oben wird die Ungleichung zu

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq c \cdot \frac{1}{2} \sup_{t \in [0,2]} \|D^2 h^{(k)}(t)\| \\ &\leq c \cdot \frac{1}{2} w \|x^{(k)} - x^*\|^2 \end{aligned} \quad (5.5.5)$$

und zwar für alle $x^{(k)} \in B_\varepsilon(x^*)$, wie noch gezeigt wird.

Sei nun $\delta \leq \varepsilon$, so dass $\frac{1}{2} \cdot c \cdot w < 1$ gilt, so folgt induktiv für $x^{(0)} \in B_\delta(x^*)$ mit (5.5.5)

$$\begin{aligned} \|x^{(k+1)} - x^*\| &\leq \frac{1}{2} w \delta^2 < \delta \\ \Rightarrow x^{(k+1)} &\in B_\delta(x^*) \subseteq B_\varepsilon(x^*) \end{aligned}$$

Auf $x^{(k+1)}$ ist der nächste Iterationsschritt anwendbar und mit (5.5.5) folgt quadratische Konvergenz.

Es bleibt zu zeigen:

$$\|D^2 h(t)\| \leq w \|x^{(k)} - x^*\|^2 \quad \forall t \in [0, 1], \quad h : [0, 1] \rightarrow \mathbb{R}^n, \quad h = \begin{pmatrix} h_1 \\ \vdots \\ h_n \end{pmatrix}$$

Hierfür betrachte

$$\begin{aligned} Dh_i^{(k)}(t) &= \underbrace{Df_i \left(x^{(k)} + t(x^* - x^{(k)}) \right)}_{\in \mathbb{R}^{1 \times n}} \cdot (x^* - x^{(k)}) \in \mathbb{R} \\ D^2 h_i^{(k)}(t) &= (x^* - x^{(k)})^T \cdot \underbrace{D^2 f_i \left(x^{(k)} + t(x^* - x^{(k)}) \right)}_{\in \mathbb{R}^{n \times n}} \cdot (x^* - x^{(k)}) \in \mathbb{R} \end{aligned}$$

□

Unter genaueren Voraussetzungen kann die Existenz von x^* gezeigt und eine Umgebung $B_r(x^*)$ explizit angegeben werden.

Dies liefert folgender

Satz (Satz von Kantorowitsch). *Sei $f : \Omega \rightarrow \mathbb{R}^n$, $\Omega_0 \subset \mathbb{R}^n$ konvex, f stetig differenzierbar auf Ω_0 und erfülle für ein $x^{(0)} \in \Omega_0$ folgendes:*

- a) $\|Df(x) - Df(y)\| \leq \gamma \|x - y\|$ für alle $x, y \in \Omega_0$
- b) $\|[Df(x^{(0)})]^1\| \leq \beta$
- c) $\|[Df(x^{(0)})]^1 f(x^{(0)})\| \leq \alpha$

mit den Konstanten $h = \alpha\beta\gamma$, $r_{\pm} = \frac{1 \pm \sqrt{1-2h}}{h} \alpha$.

Dann hat f , falls $h \leq \frac{1}{2}$ und $B_{r_-}(x^{(0)}) \subset \Omega$, genau eine Nullstelle x^* in $\Omega_0 \cap B_{r_+}(x^{(0)})$. Weiterhin bleibt die Folge der Newton-Iterierten in $B_{r_-}(x^{(0)})$ und konvergiert gegen x^* .

Beweis. z.B. in Ortega/Rheinhold (2000) □

5.6 Abbruchkriterien beim Newton-Verfahren

- 1) Limitiere die Anzahl der Iterationen, u.a. um Endlosschleifen durch fehlerhafte Programme auszuschließen.
- 2) Breche ab, wenn das Verfahren nicht konvergiert, d.h. wenn $x^{(k)}$ nicht im Konvergenzbereich bleibt.
- 3) Breche ab, wenn das Ergebnis genau genug ist, d.h. der Fehler $e^{(k)} := \|x^* - x^{(k)}\|$ klein genug ist.

5.6.1 Der Monotonietest

Beim Newton-Verfahren sollte die Funktion g der zugehörigen Fixpunktiteration eine Kontraktion sein, d.h. es muss ein $\kappa \in (0, 1)$ für alle k geben mit

$$\begin{aligned} \|\Delta x^{(k)}\| &= \|x^{(k+1)} - x^{(k)}\| \\ &= \|g(x^{(k)}) - g(x^{(k-1)})\| \\ &\leq \kappa \|x^{(k)} - x^{(k-1)}\| = \kappa \|\Delta x^{(k-1)}\| \end{aligned} \quad (5.6.1)$$

Als Abbruchkriterium für eine (mögliche) Divergenz des Verfahrens wähle z.B. $\kappa = \frac{1}{2}$ und breche ab, falls

$$\|\Delta x^{(k)}\| > \frac{1}{2} \|\Delta x^{(k-1)}\| \quad (5.6.2)$$

Um im Mehrdimensionalen eine vielleicht unnötig (teure) Berechnung von $Df(x^{(k)})$ bzw. von $\Delta x^{(k)}$ zu vermeiden, kann $\Delta x^{(k)}$ durch

$$\overline{\Delta x}^{(k)} = -[Df(x^{(k-1)})]^{-1} \cdot f(x^{(k)}) \quad (5.6.3)$$

approximiert werden. $Df(x^{(k-1)})$ und eine Zerlegung liegt bereits aus der Berechnung von $\Delta x^{(k-1)}$ vor. Ebenso ist $f(x^{(k)})$ bekannt. Die Lösung von (5.6.3) benötigt daher nur $\mathcal{O}(n^2)$ flops. Statt (5.6.2) kann dann auch auf

$$\|\overline{\Delta x}^{(k)}\| \geq \frac{1}{2} \|\Delta x^{(k-1)}\| \quad (5.6.4)$$

getestet werden.

5.6.2 Kriterium für erreichte Konvergenz

Es ist $f(x^*)$ gesucht, also teste hierauf. Das residuumbasierte Kriterium

$$\|f(x^{(k)})\| \leq Tol \quad (5.6.5)$$

ist nur bedingt anwendbar, denn nach 5.5.5 ist das Verfahren affin-invariant. Demnach bleibt $(x^{(k)})_{k \in \mathbb{N}}$ gleich, ob nun $f(x)$ oder $\tilde{f}(x) = \alpha f(x)$ betrachtet wird.

Aber für \tilde{f} bricht (5.6.5) das Verfahren ab, falls $|\alpha| \cdot \|f(x^{(k)})\| \leq Tol$.

Affin-invariant ist dagegen der Ansatz

$$\|\Delta x^{(k)}\| = \|x^{(k+1)} - x^{(k)}\| = \|[Df(x^{(k)})]^{-1} f(x^{(k)})\| \leq Tol. \quad (5.6.6)$$

(5.6.6) kann aufgrund der quadratischen Konvergenz (nur) für große k auch mit (5.3.5) der Approximation des Fehlers $\|x^* - x^{(k)}\|$ motiviert werden.

5.7 Varianten des Newton-Verfahrens

$Df(x^{(k)})$ steht nicht immer analytisch zur Verfügung. Die exakte Jacobi-Matrix wird häufig durch eine andere Matrix B approximiert, z.B. durch Differenzenquotienten oder sogenanntes „automatisches Differenzieren“. Der Iterationsschritt lautet dann

$$\begin{aligned} \text{löse } B^{(k)} d^{(k)} &= -f(x^{(k)}) \\ x^{(k+1)} &= x^{(k)} + d^{(k)} \end{aligned} \quad (5.7.1)$$

Um den Aufwand zu verringern kann $Df(x^{(k)})$ durch $Df(x^{(0)})$ approximiert werden.

5.7.1 Iterationsschritt des vereinfachten Newton-Verfahrens

$$x^{(k+1)} = x^{(k)} - [Df(x^{(0)})]^{-1} f(x^{(k)}) \quad (5.7.2)$$

Das Verfahren konvergiert nur noch lokal linear. Der Aufwand je Iteration ist jedoch erheblich geringer.

5.7.2 Das Broyden-Verfahren

01.12.2014

Das Broyden-Verfahren ist eine Verallgemeinerung des Sekantenverfahrens auf $n > 1$. $Df(x^{(k)})$ wird durch den „Differenzenquotienten“ approximiert, d.h.

$$B^{(k)} \underbrace{(x^{(k)} - x^{(k-1)})}_{:=p^{(k-1)}} = \underbrace{f(x^{(k)}) - f(x^{(k-1)})}_{:=q^{(k-1)}} \quad (5.7.3)$$

$B^{(k)}$ ist jedoch nicht eindeutig durch (5.7.3) festgelegt. Das Broyden-Verfahren bestimmt $B^{(k)}$ rekursiv durch eine Aufdatierung mit einer Rang-1-Matrix, auch „rang-1-update“ ($C_{\text{neu}} = C_{\text{alt}} + M$ mit $\text{rang}(M) = 1$).

Ein Iterationsschritt des Broyden-Verfahrens ist

$$\begin{aligned} d^{(k)} &= -[B^{(k)}]^{-1} f(x^{(k)}) \\ x^{(k+1)} &= x^{(k)} + d^{(k)} \\ p^{(k)} &:= d^{(k)} \quad \text{nach (5.7.1)} \\ q^{(k)} &:= f(x^{(k+1)}) - f(x^{(k)}) \\ B^{(k+1)} &= B^{(k)} + \frac{1}{p^{(k)T} \cdot p^{(k)}} \cdot \left(q^{(k)} - \underbrace{B^{(k)} p^{(k)}}_{= -f(x^{(k)}) \text{ nach (5.7.1)}} \right) p^{(k)T} \end{aligned} \quad (5.7.4)$$

Hierfür muss $x^{(0)}$ und $B^{(0)}$ gegeben sein. Unter bestimmten Voraussetzungen konvergiert das Verfahren lokal superlinear [siehe SB90, dortige Referenzen]. Der fleißige Leser vergewissere sich, dass für (5.7.4) auch (5.7.3) gilt.

5.7.3 Das gedämpfte Newton-Verfahren

Es gilt

$$f(x^*) = 0 \iff \min_{x \in \mathbb{R}^n} \frac{1}{2} \|f(x)\|_2^2 \quad (5.7.5)$$

Betrachte nun die Funktion $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$ mit

$$\Phi(x) := \frac{1}{2} \|f(x)\|_2^2 = \frac{1}{2} f(x)^T f(x)$$

Für Φ ist die Newton-Korrektur $d^{(k)} := \Delta x^{(k)} := -[Df(x)]^{-1} f(x)$ in $x^{(k)}$ eine **Abstiegsrichtung**, d.h. für $\mu > 0$ klein genug gilt

$$\Phi(x^{(k)} + \mu d^{(k)}) < \Phi(x^{(k)}) \quad (5.7.6)$$

denn

$$\begin{aligned} \left. \frac{d}{d\mu} \Phi(x + \mu d) \right|_{\mu=0} &= [f(x + \mu d)^T Df(x + \mu d) d]_{\mu=0} \\ &= -f(x)^T f(x) \\ &< 0 \end{aligned} \quad \text{für } f(x) \neq 0$$

Die Idee ist nun, statt $\mu = 1$ wie im Newton-Verfahren ein „geeignetes“ $\mu \in (0, 1]$ zu wählen und

$$x^{(k+1)} = x^{(k)} + \mu d^{(k)} \quad (5.7.7)$$

entsprechend zu setzen, d.h. dämpfe d mit Schrittweite μ .

Ein möglicher „Eignungstest“ ist

$$\|f(x^{(k)} + \mu d^{(k)})\| \leq (1 - \frac{1}{2}\mu) \|f(x^{(k)})\| \quad (5.7.8)$$

Und eine mögliche Strategie um μ zu bestimmen ist

1. Setze $\mu = 1$.
2. Halbiere μ rekursiv solange, bis (5.7.8) gilt.

Es sind allerdings effektivere Dämpfungsstrategien bekannt!

Es gibt also eine äußere Iteration (k) um x^* zu bestimmen und eine Innere, um für jedes (k) ein geeignetes μ zu berechnen. Die innere Schleife ist mit n eindimensionalen Funktionsauswertungen „billig“.

Unter bestimmten Voraussetzungen ist **globale** Konvergenz gewährleistet.

6 Interpolation

Es seien diskrete Werte $f_i = f(t_i)$ und eventuell $f_i^{(j)} = f^{(j)}(t_i)$ an den Punkten t_i gegeben (z.B. Messdaten).

Gesucht ist nun eine zugehörige Funktion φ .

Anwendungsfeld solcher Probleme ist z.B. CAD (computer aided design).

Eine naheliegende Forderung ist die **Interpolationseigenschaft**

$$\varphi^{(j)}(t_i) = f_i^{(j)} \quad \text{gegeben für alle } i, j,$$

d.h. φ und f sollen an den **Knoten** oder **Stützstellen** t_i übereinstimmen. Die $f_i^{(j)}$ heißen Stützwerte.



Verschiedene Grafen zu Stützstellen-Stützwert-Paaren

φ soll zusätzlich meist leicht an einer beliebigen Stelle t auswertbar sein.

Es bietet sich z.B. an

- Polynome: $\varphi(t) = a_0 + \dots + a_n t^n$
- rationale Funktion: $\varphi(t) = \frac{p(t)}{q(t)}$
(insbesondere, falls Pole vorliegen)
- Spline, d.h. stückweise Polynome
- trigonometrische Funktionen, Fouriertransformationen:
 $\varphi(t) = \sum_{j=1}^{\infty} a_j e^{-2\pi i j t}$
(insbesondere für periodische Funktionen)

6.1 Polynom-Interpolation

Wir definieren als

$$\mathcal{P}_n := \left\{ p \in \mathbb{R}[x] \mid p = \sum_{i=0}^n a_i x^i \text{ mit } a_i \in \mathbb{R} \right\} \quad (6.1.1)$$

den Raum aller Polynome mit $\deg(p) \leq n$.

Die Monome $1, x, \dots, x^n$ bilden eine Basis von \mathcal{P}_n und es gilt $\dim \mathcal{P}_n = n + 1$.

6.1a) Lagrangesche Interpolationsformel und Lemma von Aitken

Satz 6.1.1. Zu beliebigen $n + 1$ Stützpunkten $(x_i, f_i)_{i=0, \dots, n}$ mit $x_i \neq x_k$ für $i \neq k$, gibt es genau ein $p \in \mathcal{P}$ mit $p(x_i) = f_i$ für $i = 0, \dots, n$.

p heißt **Interpolationsspolynom** und wird mit $p(f|x_0, \dots, x_n)$ bezeichnet.

Beweis.

a) *Eindeutigkeit:* Seien $p_1, p_2 \in \mathcal{P}_n$ mit $p_1(x_i) = p_2(x_i) = f_i$ für $i = 0, \dots, n$. Dann ist $p := p_1 - p_2 \in \mathcal{P}_n$ mit mindestens $n + 1$ Nullstellen.

Daraus folgt bereits, dass $p \equiv 0$, also $p_1 \equiv p_2$.

b) *Existenz:* Betrachte die Polynome $L_i \in \mathcal{P}_n$ für $i = 0, \dots, n$ mit

$$L_i(x_k) = \delta_{ik} \quad \text{für } i, k = 0, \dots, n \quad (6.1.2)$$

Diese sind gegeben durch

$$L_i(x) := \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \quad (6.1.3)$$

und sind nach a) eindeutig.

Sie heißen **Lagrange-Polynome**.

Die **Lagrange-Interpolationsformel**

$$p(x) = \sum_{i=0}^n f_i L_i(x) \quad (6.1.4)$$

bestimmt somit $p(f|x_0, \dots, x_n)$, da $p \in \mathcal{P}_n$ und $p(x_i) = f_i$ für $i = 0, \dots, n$. □

Mit (6.1.4) folgt weiterhin

a) p hängt linear von den Stützwerten f_i ab.

b) Die Lagrange-Polynome bilden eine Basis des \mathcal{P}_n .

Die Lagrange-Interpolationsformel ist, wenn auch für theoretische Fragen günstig, für praktische Zwecke zu rechenaufwändig.

Zur Auswertung des Interpolationspolynoms $p(f | x_0, \dots, x_n)$ an einer festen Stelle \bar{x} , d.h. um den Wert $p(\bar{x})$ zu berechnen, benötigt man:

Lemma 6.1.2 (Lemma von Aitken). *Für das Interpolationspolynom $p(f | x_0, \dots, x_n)$ gilt die Rekursionsformel*

$$p(f | x_0, \dots, x_n)(x) = \frac{(x_0 - x)p(f | x_1, \dots, x_n)(x) - (x_n - x)p(f | x_0, \dots, x_{n-1})(x)}{(x_0 - x_n)} \quad (6.1.5)$$

für $n > 0$.

Beweis. Sei die rechte Seite von (6.1.5) definiert als $\varphi(x)$. Dann ist $\varphi \in \mathcal{P}_n$ und

$$\begin{aligned} \varphi(x_i) &= \frac{(x_0 - x_i)f_i - (x_n - x_i)f_i}{x_0 - x_n} = f_i \quad \text{für } i = 1, \dots, n-1 \\ \varphi(x_0) &= \frac{0 - (x_n - x_0)f_0}{x_0 - x_n} = f_0 \quad \text{für } i = 0 \\ \varphi(x_n) &= \frac{(x_0 - x_n)f_n - 0}{x_0 - x_n} = f_n \quad \text{für } i = n \end{aligned}$$

Damit ist $p(f | x_0, \dots, x_n) = \varphi$ aufgrund der Eindeutigkeit. □

Weiterhin gilt

03.12.2014

$$p(f | x_i)(x) = f_i \quad \forall x \in \mathbb{R} \quad (6.1.6)$$

Definiere nun für ein festes \bar{x}

$$P_{ik} := p(f | x_{i-k}, \dots, x_i)(\bar{x}) \quad (6.1.7)$$

Dann lässt sich der Wert

$$p(\bar{x}) = p(f | x_0, \dots, x_n)(\bar{x}) = P_{nn}$$

nach (6.1.5) wie folgt berechnen:

6.1.3 Schema von Neville

Setze $P_{i0} = f_i$ für $i = 0, \dots, n$ und

$$P_{ik} = P_{i,k-1} + \frac{\bar{x} - x_i}{x_i - x_{i-k}} \cdot (P_{i,k-1} - P_{i-1,k-1}) \quad \text{für } 1 \leq k \leq i \leq n \quad (6.1.8)$$

$x_i \backslash k$	0	1	...
x_0	f_0	\searrow	
x_1	f_1	\rightarrow	P_{11}
\vdots	\vdots	P_{21}	\rightarrow
\vdots	\vdots		P_{22}
\vdots	\vdots		\ddots
x_n	f_n	P_{n1}	\dots
			P_{nn}

(6.1.8) benötigt weniger Multiplikationen als (6.1.5) und ist deutlich billiger als (6.1.4). Pro Auswertung an einer Stelle \bar{x} sind $\frac{n(n+1)}{2}$ Multiplikationen und Divisionen notwendig. Soll das Interpolationspolynom an mehreren Stellen ausgewertet werden, ist das Schema von Neville zu teuer.

6.1b) Newtonsche Interpolationsformel

Betrachte folgende Basisdarstellung eines Polynoms $p \in \mathcal{P}_n$

$$p(x) = \sum_{i=0}^n a_i \prod_{j=0}^{i-1} (x - x_j) \quad (6.1.9)$$

Damit wird p zu

$$p(x) = \left(\dots \left(\left((a_n(x - x_{n-1}) + a_{n-1}) \cdot (x - x_{n-2}) + a_{n-2} \right) \cdot (x - x_{n-3}) + a_{n-3} \right) \dots \right) \quad (6.1.10)$$

Es ergibt sich für bekannte $a_i \in \mathbb{R}$:

6.1.4 Das Horner-Schema zur Auswertung $p(x)$

```

p = a_n
for k = n - 1 : -1 : 0
|   p = p(x - x_k) + a_k
end

```

Pro Auswertung benötigt es n Multiplikationen.

Zur Bestimmung der notwendigen Koeffizienten a_0, \dots, a_n von $p(f | x_0, \dots, x_n)$ kann folgende Vorwärtssubstitution verwendet werden

$$\begin{aligned} f_0 &= p(x_0) = a_0 \\ f_1 &= p(x_1) = a_0 + a_1(x_1 - x_0) \\ &\vdots \\ f_n &= p(x_n) = a_0 + \dots + a_n(x_n - x_0) \cdots (x_n - x_{n-1}) \end{aligned}$$

Es geht jedoch noch billiger, wie gleich zu sehen ist.

Definition 6.1.5.

a) Die folgenden Polynome heißen **Newton-Polynome**

$$\begin{aligned} w_0(x) &:= 1 \\ w_i(x) &:= \prod_{j=0}^{i-1} (x - x_j) \quad \text{für } i \geq 1 \end{aligned}$$

b) Die n -te dividierte Differenz $[x_0, \dots, x_n]f$ ist definiert durch

$$\begin{aligned} [x_i]f &:= f_i \\ [x_i, \dots, x_n]f &:= \frac{[x_{i+1}, \dots, x_n]f - [x_i, \dots, x_{n-1}]f}{x_n - x_i} \end{aligned} \quad (6.1.11)$$

für $x_n \neq x_i$.

Satz 6.1.6. Sei $p(f | x_0, \dots, x_n)(x) = \sum_{i=0}^n a_i w_i(x) = a_n x^n + \sum_{i=0}^{n-1} c_i x^i$, dann gilt

$$a_n = [x_0, \dots, x_n]f$$

und für jede Permutation $\sigma \in S_n$

$$[x_0, \dots, x_n]f = [x_{\sigma(0)}, \dots, x_{\sigma(n)}]f$$

Somit ist $[x_0, \dots, x_n]f$ wie p unabhängig von der Reihenfolge der Knoten und es gilt für $x_l \neq x_j$

$$[x_i, \dots, x_k]f = \frac{[x_i, \dots, \widehat{x_j}, \dots, x_k]f - [x_i, \dots, \widehat{x_l}, \dots, x_k]f}{x_l - x_j} \quad (6.1.12)$$

wobei $\widehat{}$ bedeutet, dass dieser Knoten weggelassen wird.

Beweis. Wir zeigen den Satz per Induktion.

Induktionsanfang mit $n = 0$:

$$p(f | x_0)(x) = f_0 = a_0 = [x_0]f$$

6 Interpolation

Induktionsschritt $n \rightarrow n+1$: Nach dem Lemma von Aitken gilt (6.1.5), also

$$p(f|x_0, \dots, x_{n+1})(x) = \frac{(x_0 - x)p(f|x_1, \dots, x_{n+1})(x) - (x_{n+1} - x)p(f|x_0, \dots, x_n)(x)}{(x_0 - x_{n+1})}$$

Nach Induktionsvoraussetzung gilt also für den Koeffizienten zu x^{n+1} , welcher a_{n+1} ist,

$$\begin{aligned} a_{n+1} &= \frac{-[x_1, \dots, x_{n+1}]f + [x_0, \dots, x_n]f}{x_0 - x_{n+1}} \\ &= [x_0, \dots, x_{n+1}]f \end{aligned}$$

□

Satz 6.1.7 (Newtonsche Darstellung). *Es gilt*

$$p(f|x_0, \dots, x_n)(x) = \sum_{i=0}^n [x_0, \dots, x_i]f \cdot w_i(x) \quad (6.1.13)$$

und $\{w_k | k \in \{0, \dots, n\}\}$ bildet Basis von \mathcal{P}_n .

Beweis. Wir zeigen (6.1.10) per Induktion.

$n=0$ ist klar.

$n-1 \rightarrow n$: Sei $p_n(x) := p(f|x_0, \dots, x_n)(x) = [x_0, \dots, x_n]f \cdot w_n(x) + q_{n-1}(x)$ mit $q_{n-1} \in \mathcal{P}_{n-1}$ und $q_{n-1}(x_i) = p_{n-1}(x_i) = f_i$ für $i = 0, \dots, n-1$, da $w_n(x_i) = 0$.

Damit folgt

$$q_{n-1} = p(f|x_0, \dots, x_{n-1}) = \sum_{i=0}^{n-1} [x_0, \dots, x_i]f \cdot w_i(x)$$

nach Induktionsvoraussetzung. Es folgt Gleichung (6.1.13).

$\{w_0, \dots, w_n\}$ sind linear unabhängig und jedes Polynom $\varphi \in \mathcal{P}_n$ lässt sich wegen $q(x) = p(q|x_0, \dots, x_n)(x)$ durch (6.1.13) darstellen. □

Entsprechend zum Schema von Neville für $p(\bar{x})$ ergibt sich zur Berechnung der Koeffizienten $a_i = [x_0, \dots, x_i]f$ in (6.1.10) folgendes Schema:

6.1.8 Das Schema der dividierten Differenzen

$$\begin{array}{ccccccc} x_0 & f_0 = [x_0]f & & \searrow & & & \\ x_1 & f_1 = [x_1]f & \rightarrow & [x_0, x_1]f & & \searrow & \\ \vdots & \vdots & & \vdots & & \rightarrow & [x_0, x_1, x_2]f \\ \vdots & \vdots & & \vdots & & \vdots & \ddots \\ x_n & f_n = [x_n]f & \rightarrow & [x_{n-1}, x_n]f & \rightarrow & \dots & \rightarrow [x_0, \dots, x_n]f \end{array}$$

(markiert sind die benötigten Werte)

Der Aufwand zur Berechnung aller Koeffizienten a_i beträgt

$$\frac{n(n+1)}{2} \text{ Divisionen}$$

Kombiniert mit dem Horner-Schema ergibt sich zur Auswertung von p an m Stellen ein Aufwand von

$$\frac{n(n+1)}{2} \text{ Divisionen} + m \cdot n \text{ Multiplikationen}$$

Das Schema von Neville benötigt dagegen

$$m \cdot \frac{n(n+1)}{2} \text{ Multiplikationen und Divisionen}$$

Bemerkung 6.1.9. Da $p(f|x_0, \dots, x_n)$ linear von den Stützwerten f_i abhängt, d.h. $p(\alpha f + \beta g|x_0, \dots, x_n) = \alpha p(f|x_0, \dots, x_n) + \beta p(g|x_0, \dots, x_n)$ gilt, folgt somit auch

$$[x_i, \dots, x_k](\alpha f + \beta g) = \alpha[x_i, \dots, x_k]f + \beta[x_i, \dots, x_k]g$$

Satz 6.1.10 (Leibnizregel für dividierte Differenzen). *Seien x_0, \dots, x_n paarweise verschieden. Dann gilt*

$$[x_0, \dots, x_n](g \cdot h) = \sum_{i=0}^n [x_0, \dots, x_i]g \cdot [x_i, \dots, x_n]h \quad (6.1.14)$$

Beweis. Es seien

$$G(x) := \sum_{i=0}^n [x_0, \dots, x_i]g \cdot w_i(x) = p(g|x_0, \dots, x_n)(x)$$

$$H(x) := \sum_{j=0}^n [x_j, \dots, x_n]h \cdot \tilde{w}_j(x) = p(h|x_n, \dots, x_0)(x)$$

mit $\tilde{w}_j(x) = \prod_{l=j+1}^n (x - x_l) \in \mathcal{P}_{n-j}$.

Dann folgt

$$(G \cdot H)(x_i) = g(x_i)h(x_i) \quad \forall i \leq n$$

und

$$G \cdot H = \sum_{i,j=0}^n [x_0, \dots, x_i]g \cdot [x_j, \dots, x_n]h \cdot w_i \tilde{w}_j \in \mathcal{P}_{2n}$$

Wegen

$$w_i(x_k) \tilde{w}_j(x_k) = \prod_{l=0}^{i-1} (x_k - x_l) \prod_{l=j+1}^n (x_k - x_l) = 0$$

6 Interpolation

für alle k , falls $i \geq j + 1$, interpoliert auch

$$P := \sum_{0 \leq i \leq j \leq n} [x_0, \dots, x_i] g \cdot [x_j, \dots, x_n] h \underbrace{w_i \tilde{w}_j}_{\in \mathcal{P}_{n+i-j}} \in \mathcal{P}_n$$

die Werte $g(x_k), h(x_k)$ für $k = 0, \dots, n$ und es gilt $P \in \mathcal{P}_n$.

Aufgrund der Eindeutigkeit gilt

$$p(g \cdot h | x_0, \dots, x_n) = P = \sum_{i=0}^n [x_0, \dots, x_i] g \cdot [x_i, \dots, x_n] h \cdot x^n + \sum_{l=0}^{n-1} c_l x^l$$

und Gleichung (6.1.14) folgt. □

6.1c) Hermite-Interpolation

08.12.2014

Gegeben seien die Knoten $x_0 < x_1 < \dots < x_m$ und die Werte $f_i^{(k)}$ für $k = 0, \dots, n_i - 1$ mit $i = 1, \dots, m$. Dann sind

$$n + 1 := \sum_{i=0}^m n_i$$

Werte gegeben.

Definition 6.1.11. Ein Polynom p mit $p \in \mathcal{P}_n$ und der Interpolationseigenschaft

$$p^{(k)}(x_i) = f_i^{(k)} \tag{6.1.15}$$

für $i = 0, \dots, m$ und $k = 0, \dots, n_i - 1$ heißt **Hermite-Interpolationspolynom** p .

Satz 6.1.12. Das Hermite Interpolationspolynom existiert und ist eindeutig.

Beweis.

Eindeutigkeit: Seien $p_1, p_2 \in \mathcal{P}_n$ mit (6.1.15), dann gilt $q = p_1 - p_2 \in \mathcal{P}_n$ und q hat mit Vielfachheit gezählt mindestens $(n + 1)$ Nullstellen.

Damit folgt bereits $q \equiv 0$.

Existenz: Die Existenz kann wie in Satz 6.1.1 mit verallgemeinerten Lagrangepolynomen gezeigt werden.

Alternativ: Für $p(x) = c_0 + c_1 x + \dots + c_n x^n$ ergibt sich mit (6.1.15) ein lineares GLS mit $(n + 1)$ Unbekannten und $(n + 1)$ Gleichungen. Da dieses aufgrund der Eindeutigkeit injektiv ist, ist das GLS nicht singulär und damit existiert für beliebige $f_i^{(k)}$ eine Lösung. □

Führe nun virtuelle Knoten ein:

$$\overbrace{x_0 = \dots = x_0}^{n_0} \underset{=:t_0}{=} < \overbrace{x_1 = \dots = x_1}^{n_1} \underset{=:t_{n_0+n_1-1}}{=} < \overbrace{x_m = \dots = x_m}^{n_m} \underset{=:t_{n-n_m+1}}{=}$$

und ersetze die Bedingung (6.1.15) durch

$$p^{(s_j)}(t_j) = f^{(s_j)}(t_j) \quad \text{für } j = 0, \dots, n \quad (6.1.16)$$

dann gilt

$$s_j = \max \{r \in \mathbb{N} \mid t_j = t_{j-r}\} \quad (6.1.17)$$

Beispiel 6.1.13.

$$\begin{array}{llll} x_0 = 0 & f_0^{(0)} = -1 & f_0^{(1)} = -2 & \Rightarrow n_0 = 2 \\ x_1 = 1 & f_1^{(0)} = 0 & f_1^{(1)} = 10 & f_1^{(2)} = 40 \Rightarrow n_1 = 3 \end{array} \implies n = 6$$

$$\begin{aligned} t_0 = t_1 = x_0 = 0, & \quad t_2 = t_3 = t_4 = x_1 = 1 \\ s_0 = 0, & \quad s_1 = 1, \quad s_2 = 0, \quad s_3 = 1, \quad s_4 = 2 \end{aligned}$$

Bezeichne wiederum mit $p(f \mid t_0, \dots, t_n)$ das Interpolationspolynom in \mathcal{P}_n , welches (6.1.16) erfüllt.

Lemma 6.1.14 (Lemma von Aitken). *Es gelten die Formeln*

a) falls $t_i = \dots = t_{i+k} = x_l$:

$$p(f \mid t_i, \dots, t_{i+k})(x) = \sum_{r=0}^k \frac{f^{(r)}(x_l)}{r} (x - x_l)^r \quad (6.1.18)$$

b) falls $t_i < t_{i+k}$:

$$p(f \mid t_i, \dots, t_{i+k})(x) = \frac{(x - t_i) \cdot p(f \mid t_{i+1}, \dots, t_{i+k})(x) - (x - t_{i+k}) \cdot p(f \mid t_i, \dots, t_{i+k-1})(x)}{t_{i+k} - t_i} \quad (6.1.19)$$

Beweis. In beiden Fällen wird nachgewiesen, dass die rechte Seite das zugehörige Hermite-Interpolationsproblem löst.

Aufgrund der Eindeutigkeit von $p \in \mathcal{P}_k$ folgt dann die Behauptung. \square

Definition 6.1.15. Die verallgemeinerte **dividierte Differenz** ist definiert durch

$$[t_i, \dots, t_{i+k}]f := \frac{1}{k!} f^{(k)}(x_l) \quad \text{für } t_i = \dots = t_{i+k} = x_l$$

6 Interpolation

und wie in (6.1.11) d.h.

$$[t_i, \dots, t_{i+k}]f = \frac{[t_{i+1}, \dots, t_{i+k}]f - [t_i, \dots, t_{i+k-1}]f}{t_{i+k} - t_i} \quad \text{für } t_i < t_{i+k}$$

Wie in 6.1.6 und 6.1.7 folgt aus dem Lemma von Aitken die **Newtonsche-Darstellung** (6.1.13)

$$p(f | t_0, \dots, t_n)(x) = \sum_{i=0}^n [t_0, \dots, t_i]f \cdot w_i(x)$$

mit $w_i(x) = \prod_{j=0}^{i-1} (x - t_j)$.

Ebenso folgt die Leibnizformel für $t_0 \leq t_1 \leq \dots \leq t_n$ sowie $g, h \in C^n(I)$ aufgrund der Stetigkeit der dividierten Differenzen bzgl. t_i .

Letzteres wäre noch zu zeigen.

Beispiel 6.1.16 (Fortsetzung von 6.1.131).

	x_i	$[t_i]f$	$[t_i, t_{i+1}]f$	\dots		
t_0	0	-1				
t_1	0	-1	-2			
t_2	1	0	1	3		
t_3	1	0	10	9	6	
t_4	1	0	10	10	11	5

Damit folgt für p

$$\begin{aligned} p(f | t_0, \dots, t_4)(x) &= -1 \\ &\quad - 2(x-0) \\ &\quad + 3(x-0)(x-0) \\ &\quad + 6(x-0)(x-0)(x-1) \\ &\quad + 5(x-0)(x-0)(x-1)(x-1) \\ &= -1 - 2x + 3x^2 + 6x^2(x-1) + 5x^2(x-1)^2 \end{aligned}$$

6.1d) Fehlerabschätzung bei Polynominterpolation

Der Fehler $f(x) - p(x)$ kann nicht abgeschätzt werden, falls nicht mehr als die Funktionswerte f_i von f bekannt sind.

Satz 6.1.17 (Fehlerdarstellung mit dividierten Differenzen). *Es gilt die Fehlerdarstellung für alle f*

$$f(\bar{x}) - p(f | t_0, \dots, t_n)(\bar{x}) = [t_0, \dots, t_n, \bar{x}]f \cdot w_{n+1}(\bar{x}) \quad (6.1.20)$$

Beweis.

$$f(\bar{x}) - p(f|t_0, \dots, t_n)(\bar{x}) = p(f|t_0, \dots, t_n)(\bar{x}) + [t_0, \dots, t_n, \bar{x}]f \cdot w_{n+1}(\bar{x})$$

aufgrund der Newtonschen Darstellung (6.1.13). \square

Satz 6.1.18 (Restglieddarstellung). *Sei f $(n+1)$ -mal stetig differenzierbar, so gibt es zu jedem \bar{x} eine Zahl $\xi(\bar{x})$ aus dem kleinsten Intervall I , welches x_0, \dots, x_m enthält, so dass*

$$f(\bar{x}) - p(f|t_0, \dots, t_n)(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot w_{n+1}(\bar{x}) \quad (6.1.21)$$

Beweis. Sei $\bar{x} \neq x_i$ (sonst trivial)



Veranschaulichung zur Restglieddarstellung

10.12.2014

Definiere

$$F(x) := f(x) - p(f|t_0, \dots, t_n)(x) - K w_{n+1}(x) \quad (6.1.22)$$

mit $K = \text{const}$ so gewählt, dass $F(\bar{x}) = 0$.

Aufgrund von $p^{(k)}(x_i) = f^{(k)}(x_i)$ und $w_{n+1}^{(k)}(x_i) = 0$ wegen $w_{n+1}(x) = (x-t_0) \cdots (x-t_n) = \prod_{l=0}^n (x-x_l)^{n_l}$ gilt

$$F^{(k)}(x_i) = 0 \quad \text{für } i = 0, \dots, m$$

sowie

$$F(\bar{x}) = 0$$

Da $F(x_i) = 0$ für $i = 0, \dots, m$ und $F(\bar{x}) = 0$, folgt

$$\exists \rho_0^{(1)}, \dots, \rho_m^{(1)} \in I, : F'(\rho_i^{(1)}) = 0$$

nach dem Satz von Rolle mit paarweise verschiedenen $\rho_i^{(1)}$.

F' hat also $(m+1) + \#\{i | n_i > 1\}$ paarweise verschiedene Nullstellen. Nach dem Satz von Rolle hat F'' also $n + \#\{i | n_i > 1\}$ paarweise verschiedene Nullstellen in I , die ungleich zu den x_i sind für die $n_i > 2$.

6 Interpolation

Weiterhin gilt $F''(x_i) = 0$, falls $n_i > 2$.
Sukzessive folgt damit für $F^{(n+1)}$ die Existenz von

$$m + 1 - n + \underbrace{\#\{i \mid n_i > 1\} + \#\{i \mid n_i > 2\} + \cdots + \#\{i \mid n_i > n\}}_{=n-m}$$

paarweise verschiedene Nullstellen in I^{n-m} .

Also existiert mindestens eine Nullstelle ξ von $F^{(n+1)} = f^{(n+1)} - K(n+1)!$. Damit ist

$$K = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Mit $F(\bar{x}) = 0$ folgt die Behauptung. \square

Folgerung 6.1.19. Für $f \in C^n(I)$ mit $I = [\min_{i \leq m} x_i, \max_{i \leq m} x_i]$ gibt es ein $\xi \in I$ mit

$$[t_0, \dots, t_n]f = \frac{f^{(n)}(\xi)}{n!} \quad (6.1.23)$$

Beweis. Für $f \in C^{n+1}(I)$ folgt aus Beispiel 6.1.16 und (6.1.16)

$$K = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

womit (6.1.23) folgt. \square

Wie in der obigen Fehlerdarstellung zu sehen ist, spielt nicht nur $f^{(n+1)}$ bzw. $[t_0, \dots, t_n, \bar{x}]f$ eine Rolle für den Fehler, sondern mit $w_{n+1}(\bar{x})$ auch die Verteilung der Knoten. Falls die Knoten frei wählbar sind und

$$\|f - p(f \mid t_0, \dots, t_n)\|_{C([a,b])}$$

klein ist, wird gewünscht, dass

$$\|w_{n+1}\|_{C([a,b])} := \|w_{n+1}\|_{\infty} = \max_{x \in [a,b]} |w_{n+1}(x)|$$

minimal ist, d.h. es wird

$$\min_{t_0, \dots, t_n \in \mathbb{R}} \max_{x \in [a,b]} |(x - t_0) \cdots (x - t_n)|$$

gesucht.

Falls $t_i = x_i$, kann gezeigt werden [z.B. DH08; FH07], dass für $[a, b] = [-1, 1]$ das zugehörige w_{n+1} , das sogenannte **Tschebyscheff-Polynom**

$$T_{n+1}(x) := \cos((n+1) \arccos(x)) \in \mathcal{P}_{n+1} \quad (6.1.24)$$

ist. Dessen Nullstellen, die **Tschebyscheff-Punkte**, sind

$$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right) \quad \text{für } i = 0, \dots, n \quad (6.1.25)$$

Weiterhin ist der führende Koeffizient von T_{n+1} genau 2^n , sodass

$$\|w_{n+1}\|_{C([-1,1])} = 2^{-n} \quad (6.1.26)$$

für die Tschebyscheff-Punkte gilt.

Für andere Intervalle $[a, b]$ wird die lineare Transformation

$$x = \frac{1}{2}(t(b-a) + (b+a))$$

von $[-1, 1]$ auf $[a, b]$ durchgeführt.

Wird die Fehlerabschätzung bzgl. der $\|\cdot\|_2$ -Norm, d.h.

$$\|f\|_2 := \left(\int_a^b |f(t)|^2 dt \right)$$

betrachtet, erweisen sich auf $[-1, 1]$ die Nullstellen des **Legendre-Polynoms** P_{n+1} als minimal [HH94], wobei

$$P_n(x) = \frac{n!}{(2n)!} \frac{d^n}{dx^n} ((x^2 - 1)^n)$$

Achtung: Die Folge der Interpolationspolynome, die sich bei gleichmäßiger Knotenverteilung ergeben, konvergieren **nicht** gegen f mit wachsender Anzahl der Knoten!

Ein Gegenbeispiel ist $f(x) = |x|$ in $[-1, 1]$:

Die Folge $\{p(f|x_0, \dots, x_n)(x)\}_{n \in \mathbb{N}}$ mit x_i gleichmäßig verteilt, divergiert für alle Werte $0 < |x| < 1$.

Ein weiteres Gegenbeispiel ist $f(x) = \frac{1}{1+x^2}$ in $[-5, 5]$ (siehe Übungsaufgabe).

Es gilt sogar:

Satz 6.1.20 (Faber). *Zu jeder Folge von Intervalleinteilungen von $[a, b]$ gibt es eine stetige Funktion f , so dass die Interpolationspolynome auf $[a, b]$ nicht gleichmäßig gegen f konvergiert (bzgl. $\|\cdot\|_\infty$).*

Basierend auf dem Satz von Jackson gilt:

Satz 6.1.21 (Konvergenz bei Tschebyscheff-Knoten). *Sei $f \in C([-1, 1])$ Lipschitz-stetig. Dann konvergieren die Interpolationspolynome zu den Tschebyscheff-Knoten gleichmäßig gegen f .*

Andererseits gilt [siehe HH94]:

6 Interpolation

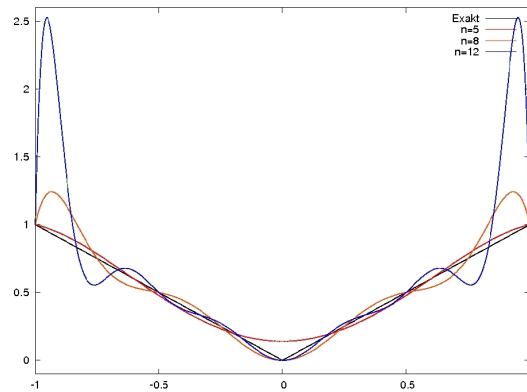


Abbildung 6.1: Gegenbeispiel zur Konvergenz: äquidistante Gitter für Betragsfunktion

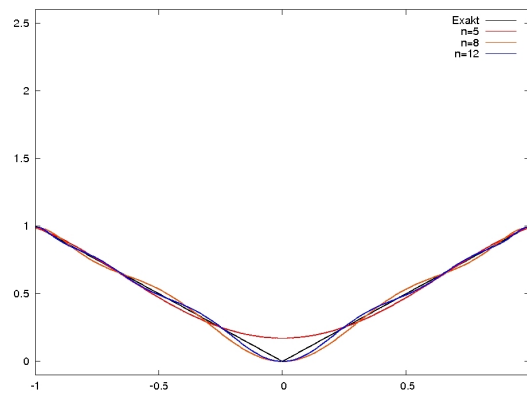


Abbildung 6.2: Gegenbeispiel zur Konvergenz: Tschebyscheff-Punkte für Betragsfunktion

Satz 6.1.22 (Marcinkiewicz). *Zu jeder Funktion $f \in C([a, b])$ kann eine Folge von Intervalleinteilungen angegeben werden, so dass die Folge der Interpolationspolynome gleichmäßig gegen f konvergiert.*

Im Gegensatz zur $\|\cdot\|_\infty$ -Norm (auch Tschebyscheff-Norm genannt) gilt für die $\|\cdot\|_2$ -Norm [siehe HH94]:

Satz 6.1.23. *Sei $f \in C([-1, 1])$ und seien x_0, \dots, x_n die Nullstellen des Legendre-Polynoms $L_n + 1$. Dann gilt*

$$\lim_{n \rightarrow \infty} \|f - p(f | x_0, \dots, x_n)\|_2 = 0$$

6.2 Stückweise polynomiale Approximation durch Splines

Eine hohe Anzahl von Knoten garantiert keine „gute“ Approximation durch Interpolationspolynome. In der Regel treten dann hohe Schwingungen auf, die unerwünscht sind. Die Idee der stückweisen polynomialen Approximation ist, jeweils einige wenige Stützwerte polynomial zu interpolieren und diese „glatt“ zusammenzusetzen.



Veranschaulichung stückweiser polynomialer Approximation

- stückweise linear und stetig ist am einfachsten
- stückweise kubisch und global zweimal stetig differenzierbar ist zur graphischen Aufarbeitung am geeignetsten, da das Auge noch Unstetigkeiten in der Krümmung erkennt.

6.2a) Splines und zwei verschiedene Basen

Definition 6.2.1. Sei $\Delta = \{x_0, \dots, x_n\}$ ein Gitter mit paarweise verschiedenen Knoten $a = x_0 < \dots < x_n = b$.

Ein (Polynom-) **Spline von Ordnung** $k \geq 2$ (d.h. der Grad der Teilstückpolynome ist maximal $(k-1)$) ist eine Funktion s mit $s \in C^{(k-2)}([a, b])$ und $s \in \mathcal{P}_{k-1}([x_i, x_{i+1}])$ für $i = 0, \dots, n-1$.

Der Raum der Splines von Ordnung k bzgl. Δ sei mit $S_{k,\Delta}$ bezeichnet.

Für $k = 1$ ist der Raum definiert durch

$$S_{1,\Delta} := \{s : [a, b] \rightarrow \mathbb{R} \mid s|_{[x_i, x_{i+1}]} \in \mathcal{P}_0 \text{ für } 0 \leq i \leq n\}$$

Für $k = 2$ ergeben sich **lineare Splines** und für $k = 4$ die **kubischen Splines**.

Offensichtlich gilt

a) $S_{k,\Delta}$ ist ein linearer Vektorraum.

b) $\mathcal{P}_{k-1} \subseteq S_{k,\Delta}$

Natürlich lässt sich ein Spline stückweise als Polynom darstellen, es hat also $n \cdot k$ Koeffizienten. Gewünscht ist aber eine Basisdarstellung, welche dann $n+k-1$ Koeffizienten hat.

6 Interpolation

Satz 6.2.2. Die Monome x^l und die **abgebrochenen Potenzen**

$$(x - x_i)_+^l := \begin{cases} (x - x_i)^l & x \geq x_i \\ 0 & x < x_i \end{cases} \quad (6.2.1)$$

bilden eine Basis

$$\mathcal{B} = \{1, x, \dots, x^{k-1}, (x - x_1)_+^{k-1}, \dots, (x - x_{n-1})_+^{k-1}\}$$

des Splineraumes $S_{k,\Delta}$. Insbesondere gilt

$$\dim S_{k,\Delta} = k + n - 1 \quad (6.2.2)$$



Dimension eines Splineraumes

Beweis. siehe Übungsaufgabe

□

6.2.3 Nachteile der Splineraumbasis

- a) Basisfunktionen haben keine lokalen Träger, d.h. zur Auswertung von $s(\bar{x})$ müssen alle Basisfunktionen ausgewertet werden, was relativ teuer ist.
- b) Falls $x_i \approx x_{i+1}$ sind $(x - x_i)_+^{k-1}$ und $(x - x_{i+1})_+^{k-1}$ nahezu linear unabhängig. Damit ist die Basis schlecht konditioniert bzgl. Störungen in b_i .

Das Ziel ist nun eine bessere Basis zu konstruieren.

Beispiel 6.2.4.

- a) $S_{1,\Delta}$ sind stückweise konstante Funktionen



Basis eines $S_{1,\Delta}$ -Splines aus stückweise konstanten Elementen

6.2 Stückweise polynomiale Approximation durch Splines

Die zugehörige Basis wie oben beschrieben wäre

$$\{1, (x - x_i)_+^0\}$$

Eine in diesem Fall geeignetere Basis ist

$$N_{i,1}(x) := \chi_{[x_{i-1}, x_i)}(x) = \begin{cases} 1 & \text{falls } x \in [x_{i-1}, x_i) \\ 0 & \text{sonst} \end{cases},$$

womit die Darstellung eines $S_{1,\Delta}$ -Splines zu

$$s(x) = \sum_{j=1}^{n+1} f_{j-1} N_{j,1}(x)$$

wird.

b) $S_{2,\Delta}$ sind stückweise lineare, global stetige Funktionen. Eine Basis wie in 6.2.2 ist

$$\{1, x^1, (x - x_i)_+^1\}$$



Basis eines $S_{2,\Delta}$ -Splines mit Hutfunktion (siehe auch 6.2.6 für allg. Hutfunktionen)

Hier kann eine Basis wie oben durch sog. **Hutfunktionen** konstruiert werden

$$N_{i,2}(x_{j-1}) = \delta_{i,j}$$

$$N_{i,2}(x) = \begin{cases} \frac{1}{x_{i-1}-x_{i-2}}(x - x_{i-2}) & x \in [x_{i-2}, x_{i-1}] \\ -\frac{1}{x_{i-1}-x_{i-2}}(x - x_{i-1}) + 1 & x \in [x_{i-1}, x_i] \end{cases}$$



allgemeine Hutfunktion am Punkt x_i eines $S_{2,\Delta}$ -Splines

Ein Spline, welches f in x_i interpoliert hat dann die Form

$$s(x) = \sum_{j=1}^{n+1} f_{i-1} N_{i,2}(x)$$

6 Interpolation

Definition 6.2.5. Sei $t_1 \leq \dots \leq t_m$ eine beliebige Folge von Knoten x_i .

Dann sind die **B-Splines** $n_{j,l}$ für $l = 1, \dots, m$ und $j = 1, \dots, m - l$ rekursiv erklärt durch

$$\begin{aligned} N_{j,1}(x) &:= \chi_{[t_j, t_{j+1}]}(x) \text{ für } t_{j+1} \neq t_j \\ N_{p,1}(x) &= \chi_{[t_p, t_m]}(x) \text{ für } p = \min \{i \mid t_{i+1} = t_m\} \end{aligned} \quad (6.2.3)$$

sowie für $l > 1$

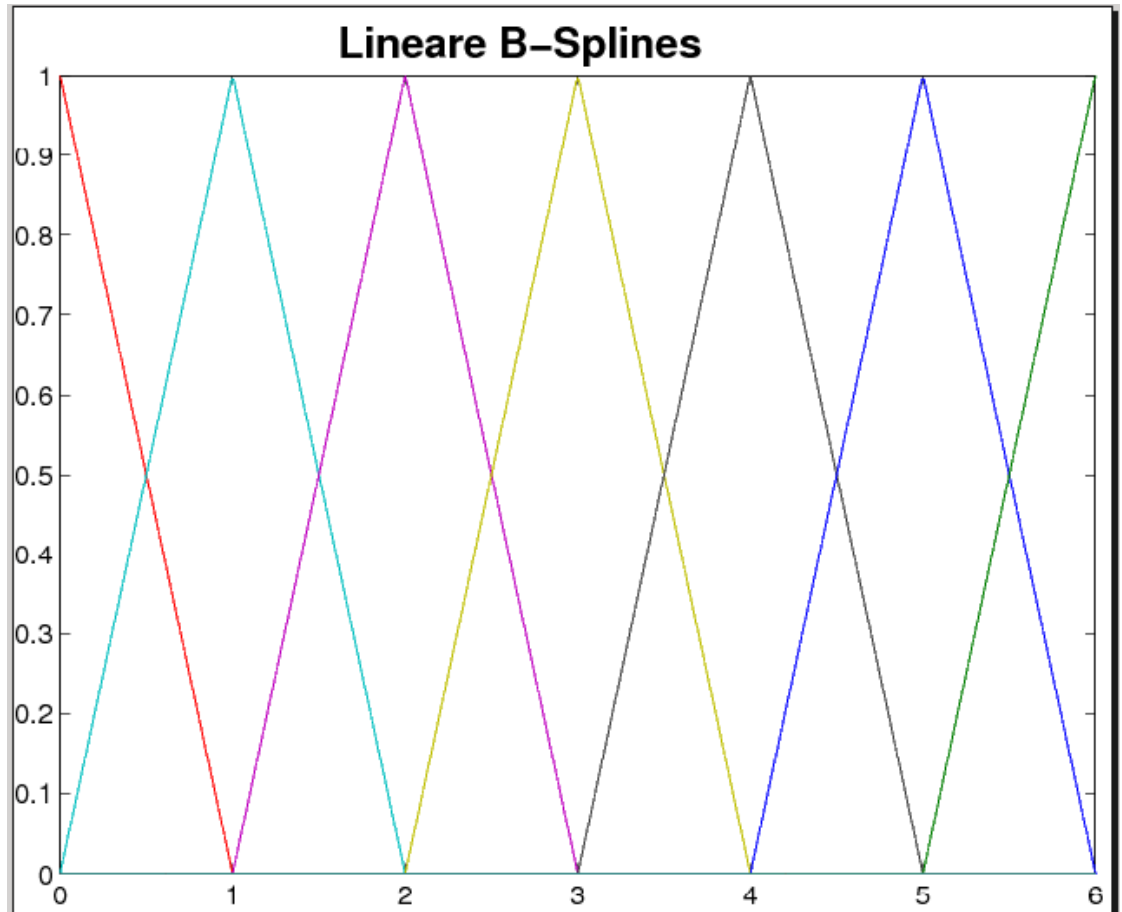
$$N_{j,l}(x) = \frac{x - t_j}{t_{j+l-1} - t_j} N_{j,l-1}(x) + \frac{x - t_{j+1}}{t_{j+l} - t_{j+1}} N_{j+1,l-1}(x) \quad (6.2.4)$$

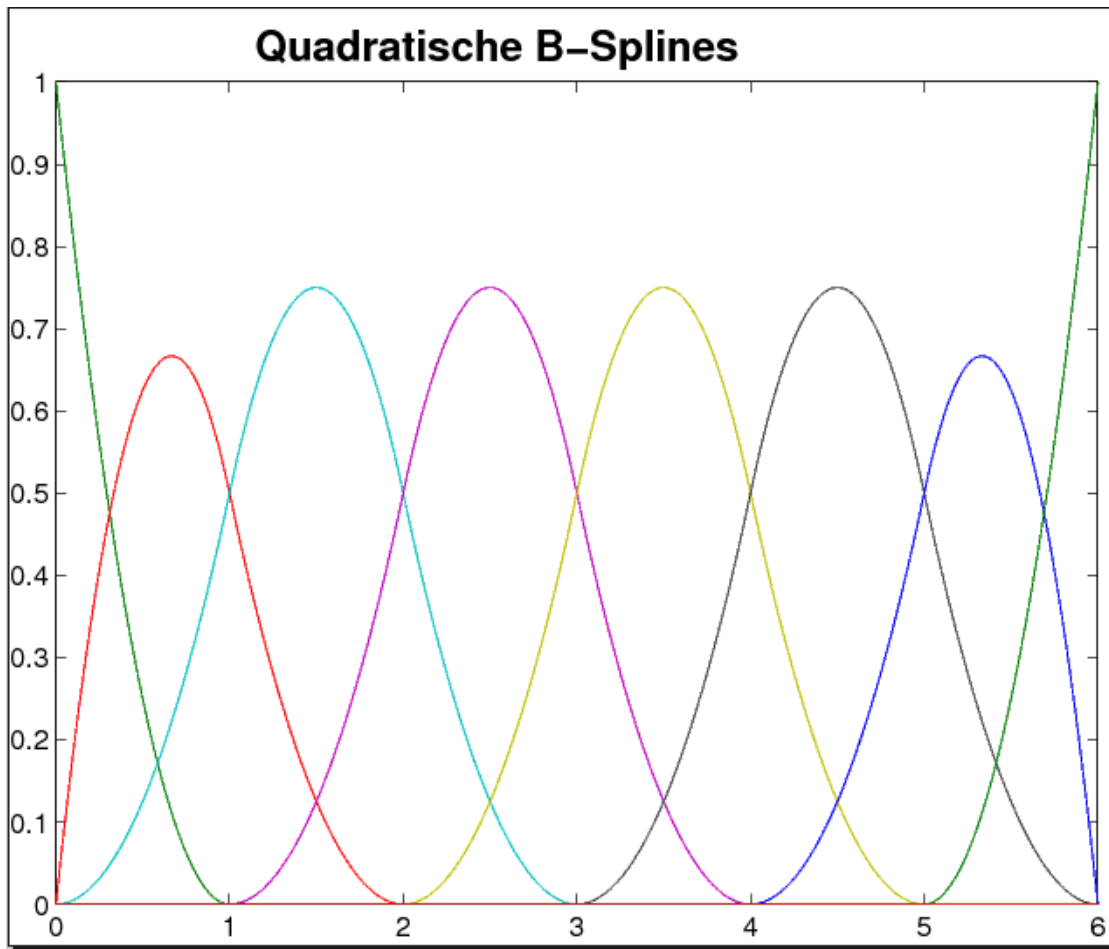
Hierbei gilt die Konvention $\frac{0}{0} = 0$, denn für $t_j = t_{j+l-1}$ gilt $N_{j,1} \equiv 0$ und auch $N_{j,l-1} \equiv 0$.

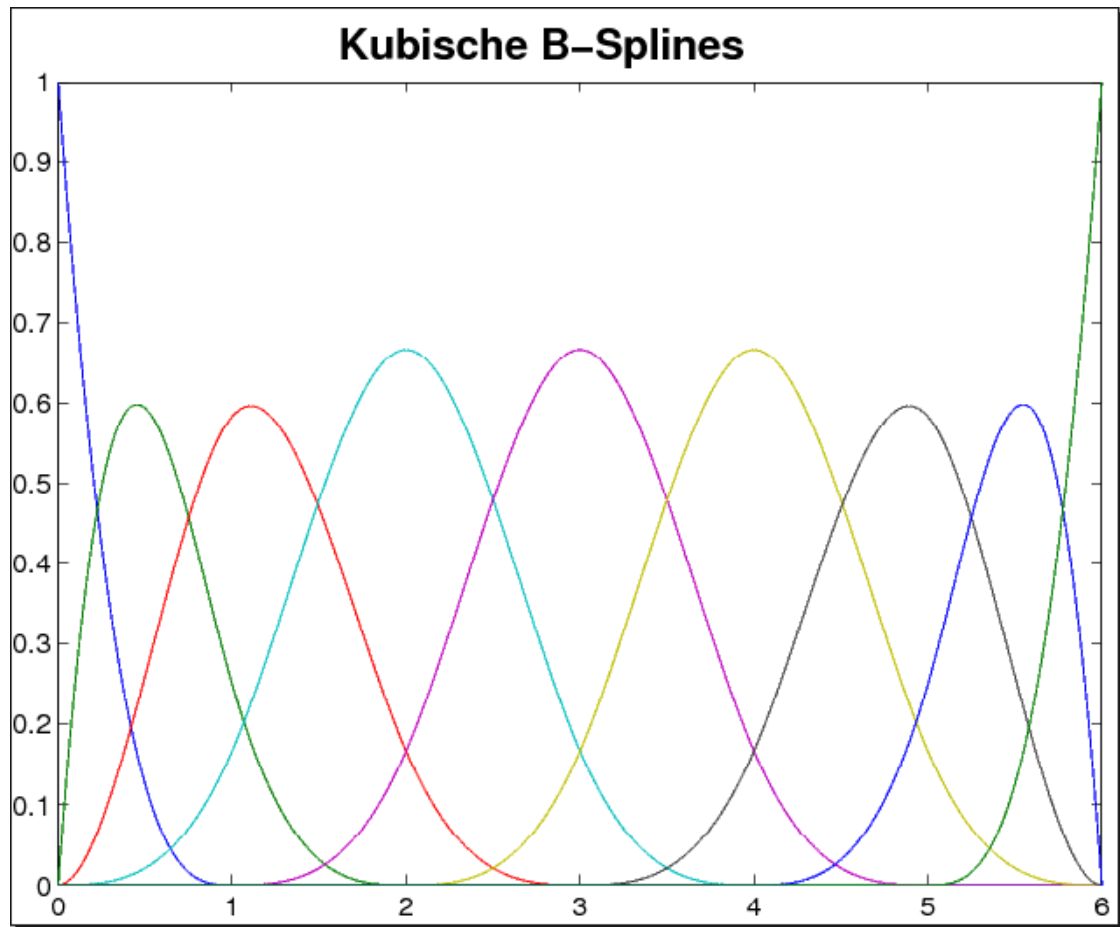
Ziel ist nun u.a. mit diesem eine Basis von $S_{k,\Delta}$ anzugeben, die zudem gut konditioniert ist. Vorher jedoch werden noch einige Eigenschaften der B-Splines untersucht.

6.2.6 Gestalt der B-Splines

Bei einem B-Spline $N_{j,k}$ entspricht j dem Ort und $k - 1$ entspricht dem Polynomgrad.







Korollar 6.2.7. *Es gilt*

a) $\text{supp} N_{j,k} := \overline{\{x \in [a, b] \mid N_{j,k}(x) \neq 0\}} \subset [t_j, t_{j+1}]$ (lokaler Träger)

b) $N_{j,k} > 0 \quad \forall x \in (t_j, t_{j+1})$ (nicht negativ)

c) $N_{j,k}|_{[t_l, t_{l+1}]} \in \mathcal{P}_{k-1}$

Beweis. siehe Übungsaufgabe

□

6.2.8 Auswertung von s an der Stelle \bar{x}

Sei $\bar{x} \in [t_j, t_{j+1}]$ und $s(x) = \sum_{i=1}^N c_i N_{i,k}(x)$. Da nur $N_{j,1}(\bar{x}) \neq 0$ ist, können aufgrund von (6.2.4) auch nur $N_{j-l+1,l}(\bar{x}), \dots, N_{j,k}(\bar{x})$ verschieden von Null sein, da

$j \setminus l$	t_j	1	2	3	k
1	t_1	$N_{1,1}(\bar{x}) = 0$	0	0	0
		0	\vdots	\vdots	\vdots
		\vdots			0
$j - k + 1$	t_{j-k+1}	\dots	\dots	\dots	$N_{j-k+1,k}(\bar{x})$
\vdots	\vdots				\vdots
			0	$N_{j-2,3}(\bar{x})$	$N_{j-2,k}(\bar{x})$
$j - 1$	t_{j-1}	0	$\longrightarrow N_{j-1,2}(\bar{x})$	$\longrightarrow N_{j-1,3}(\bar{x})$	$N_{j-1,k}(\bar{x})$
j	t_j	$N_{j,1}(\bar{x}) = 1$	$\longrightarrow N_{j,2}(\bar{x})$	$\longrightarrow N_{j,3}(\bar{x})$	$\longrightarrow \dots N_{j,k}(\bar{x})$
		0	0		\vdots
		\vdots	\vdots		
	t_{j+k}	0	0		0

Daraus folgt, dass $N_{i,k}(\bar{x}) \neq 0$ für höchstens $i = j - k + 1, \dots, j$.

Nach der Rekursionsformel baut $N_{i,k}$ auf $N_{i,1}, \dots, N_{i+(k-1),1}$ auf und benutzt t_i, \dots, t_{i+k} . Da in der Rekursionsformel nur nichtnegative Vielfache nichtnegativer Zahlen addiert werden, ist das Verfahren zur Bestimmung der $N_{i,k}(\bar{x})$ numerisch sehr stabil.

Lemma 6.2.9. Falls $t_j < t_{j+k}$ und $x \neq t_m$, so gilt

$$N_{j,k}(\bar{x}) = (t_{j+k} - t_j) [t_j, \dots, t_{j+k}](\bullet - x)_+^{k-1} \quad (6.2.5)$$

$[t_j, \dots, t_{j+k}]f$ mit $f(t) := (t - x)_+^{k-1}$

Beweis. kein Beweis.

Idee: Per Induktion über k . Nutze hierfür

$$(t - x)_+^k = (t - x)(t - x)_+^{k-1}, \quad k \geq 1,$$

sowie $[t_j, \dots, t_i](\bullet - x) = 0$ für $i > j + 1$ und die Leibnizregel. \square

Zu gegebenem Splineraum $S_{j,\Delta}$ mit

$$\Delta a = x_0 < x_1 < \dots < x_n = b \quad (6.2.6)$$

Setzt man nun

$$T : t_1 = \dots = t_k < t_{k+1} < \dots < t_{k+n} = t_{n+2k+1}, \quad (6.2.7)$$

lässt sich folgender Satz zeigen:

Satz 6.2.10.

17.12.2014

6 Interpolation

1. Es gilt $\mathcal{P}_{k-1} \subset \text{span} \{N_{1,k}, \dots, N_{n+k-1,k}\}$
2. Die B-Splines bilden eine Zerlegung der Eins auf $[a, b]$, d.h.

$$1 \equiv \sum_{j=1}^{n+k-1} N_{j,k}(x) \quad \forall x \in [a, b]$$

3. Weiterhin sind die B-Splines (lokal) linear unabhängig, d.h. falls

$$\sum_{l=1}^{n+k-1} c_l N_{l,k}(x) = 0 \quad \forall x \in (c, d) \subset [a, b],$$

folgt $c_j = 0$ für alle j mit $(c, d) \cap \underbrace{(t_j, t_j + k)}_{=\text{supp} N_{j,k}} \neq \emptyset$.



Lineare Unabhängigkeit der B-Splines

Beweis.

a), b) folgen aus der Marschen-Identität [siehe z.B. DH08]

c) O.B.d.A. enthalte (c, d) keine Knoten t_i (sonst zerlege (c, d) in Teilintervalle). Dann folgt $(c, d) \subseteq (t_l, t_{l+1})$. Nach a) lässt sich jedes Polynom $p \in \mathcal{P}_{k-1}$ durch die B-Splines $N_{l,k}$ darstellen. Auf (c, d) sind nur $k = \dim \mathcal{P}_{k-1}$ B-Splines von Null verschieden. Daher müssen diese linear unabhängig sein.

□

Folgerung 6.2.11. Die B-Splines $\{N_{1,k}, \dots, N_{n+k-1,k}\}$ bilden eine Basis von $S_{k,\Delta}$.

Die Koeffizienten c_i der Basisdarstellung von $s \in S_{k,\Delta}$

$$s(x) = \sum_{i=1}^{n+k-1} c_i N_{ik}(x)$$

heißen **de Boor-Punkte** von s .

Die Funktionswerte $s(x)$ sind somit eine Konvexkombination der de Boor-Punkte c_i .

Bemerkung 6.2.12. Die B-Spline Basis ist eine gut konditionierte Basis, d.h. die Auswertung von Splines mittels ihrer B-Spline Darstellung ist gut konditioniert. Es gilt mit $N := n + k - 1$

$$D_k \max_{j=1, \dots, N} |c_j| \leq \left\| \sum_{j=1}^N c_j N_{jk} \right\|_{\infty} \leq \max_{j=1, \dots, N} |c_j|$$

Die Konstante D_k hängt nur von der Ordnung k ab, nicht von der Lage der Knoten.

6.2b) Splineinterpolation

6.2.13 Splineinterpolation allgemein

Gegeben seien $N = n + k - 1 = \dim S_{k,\Delta}$ Stützpunkte (ξ_i, f_i) mit

$$\xi_1 < \dots < \xi_N.$$

Gesucht sind nun Splines $s \in S_{k,\Delta}$ mit $s(\xi_i) = f_i$ für alle $i = 1, \dots, N$. Das ist äquivalent zu

$$\begin{aligned} \sum_{j=1}^N c_j N_{j,k}(\xi_i) &= f_i && \text{für } i = 1, \dots, N \\ \Leftrightarrow Ac &= f && \text{mit } A = (N_{j,k}(\xi_i))_{i,j=1,\dots,N} \end{aligned} \quad (6.2.8)$$

6.2.14 Lineare B-Splines

Sei $k = 2$, $N = n + 1$, $S_{2,\Delta}$ der Splineraum zu einem Δ , welches konstruiert werden soll.



Stützstellen für eine konstruierte Zerlegung Δ

Sei $x_0 := \xi_1 < x_1 := \xi_2 < \dots < x_n := \xi_{n+1}$, dann folgt

$$N_{i,2}(\xi_j) = N_{i,2}(t_{j+1}) = \delta_{i,j}$$

Also ist in diesem Fall die Matrix $A = (N_{j,k}(\xi_i))_{i,j \leq n} = I$ also $c_i = f_i = f(\xi_i) = f(x_{i-1})$. Der zugehörige Spline s hat dann die Form

$$\begin{aligned} s(x) &= \sum_{i=1}^N f_i N_{i,2}(x) \\ &= \sum_{j=0}^N f(x_j) N_{j+1,2}(x) \end{aligned}$$

Zur Auswertung von s wird 6.2.8 verwendet. A sei definiert als

$$A := (N_{j,k}(\xi_i))_{i,j=1,\dots,N}$$

und hat folgende Eigenschaften

6 Interpolation

- A besitzt Bandstruktur, da ξ_i höchstens in k Intervallen $[t_j, t_{j+k}]$ liegen kann.
- A ist regulär, also $N_{ik}(\xi_i) \neq 0$ für $i = 1, \dots, N$.
(nach dem Satz von Schoenberg und Whitney, 1953)
- A ist total positiv, d.h. für alle quadratischen Untermatrizen B gilt $\det(B) \geq 0$.
(nach Karlin, 1968)

Daraus kann man folgern, dass die Gaußelimination für Bandmatrizen ohne Pivotsuche durchführbar ist, falls $N_{ik}(\xi_i) \neq 0$ für $i = 1, \dots, N$.

6.2.15 Kubische B-Spline-Interpolation

Sei $k = 4$ und entsprechend $N = n + 3$ und seien die Stützpunkte zur Interpolation wie in 6.2.14 gegeben durch

$$\xi_i = x_{i-1} \quad \text{für } i = 1, \dots, n+1.$$

(Daraus folgt $N \neq n + 3$ und 6.2.13 ist nicht anwendbar.)

Zur eindeutigen Bestimmung von $s \in S_{4,\Delta}$ fehlen zwei Bedingungen.
Typischerweise werden zusätzlich zu

$$s(x_i) = f(x_i) \quad \text{für } i = 0, \dots, n$$

eine der folgenden Randbedingungen gefordert:

- i) $s'(a) = f'(a)$ und $s'(b) = f'(b)$:

vollständige oder Hermitesche kubische Spline-Interpolation

- ii) $s''(a) = s''(b) = 0$ („natürliche“ Endbedingungen):

natürliche kubische Spline-Interpolation

(d.h. s kann linear außerhalb von $[a, b]$ glatt fortgesetzt werden)

- iii) $s'(a) = s'(b)$ und $s''(a) = s''(b)$, falls f periodisch mit Periode $b - a$ ist:

periodische kubische Spline-Interpolation.

Die drei Randbedingungen werden aufgrund physikalischer Eigenschaften gefordert.
Hierzu ein Beispiel:

Es beschreibe $y(t)$ die Lage eines homogenen, isotropen Stabes (z.B. eine dünne Holzlatte).
Dann misst

$$E(y) = c \int_a^b \left(\frac{y''(t)}{(1 + y'(t)^2)^{\frac{3}{2}}} \right)^2 dt = c \int_a^b (\kappa(t))^2 dt$$

(wobei κ die Krümmung der Kurve y in der Ebene ist) die Biegeenergie.

Aufgrund des **Hamiltonschen Prinzips** stellt sich der Stab so ein, dass $E(y)$ minimiert

6.2 Stückweise polynomiale Approximation durch Splines

wird.

Unter der Annahme $|y'(t)| \ll 1$ für $t \in [a, b]$ wird E linearisiert zu $c \int_a^b y''(t)^2 dt$. Also wird annähernd eine Funktion $y \in C^2$ gesucht, welche $\|y''\|_2^2$ minimiert.

Obige Splines haben gerade diese Eigenschaft, denn:

Satz 6.2.16. *Sei s ein interpolierender, kubischer Spline von f in den Knoten x_i und $y \in C^2$ eine beliebige Funktion von f , so dass*

$$\left[s''(t) \cdot (y'(t) - s'(t)) \right]_{t=a}^b = 0$$

Dann gilt $\|s''\|_2 \leq \|y''\|_2$.

Beweis. Es gilt

$$\begin{aligned} \|y''\|_2^2 &= \|s'' + (y'' - s'')\|_2^2 \\ &= \|s''\|_2^2 + 2 \underbrace{\int_a^b s''(y'' - s'') dt}_{=0} + \|y'' - s''\|_2^2 \\ &\quad \text{wie gezeigt wird} \end{aligned}$$

da

$$\int_a^b s''(y'' - s'') dt = \sum_{i=1}^N \left(\left[s''(y' - s') \right]_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} s'''(y' - s') dt \right)$$

($S_{4,\Delta} \subseteq C^2([a, b])$, $s'''(x) \equiv d_{i-1} = \text{const}$, da $s \in \mathcal{P}_3([x_{i-1}, x_i])$)

$$\begin{aligned} &= \underbrace{\left[s''(y' - s') \right]_a^b}_{=0} - \sum_{i=1}^N d_{i-1} \int_{x_{i-1}}^{x_i} (y'(t) - s'(t)) dt \\ &\quad \text{nach Voraussetzung} \\ &= - \sum_{i=1}^N d_{i-1} \left((y(x_i) - s(x_i)) - (y(x_{i-1}) - s(x_{i-1})) \right) \\ &= 0 \end{aligned}$$

da y, s Interpolationen von f sind.

Mit den Randbedingungen i), ii) und iii) ist die Voraussetzung von 6.2.16 erfüllt. \square

Korollar 6.2.17. *Es existiert genau ein interpolierender kubischer Spline $s \in S_{4,\Delta}$, welcher die Randbedingungen aus 6.2.16 erfüllt. Weiterhin gilt für jede interpolierende Funktion $y \in C^2([a, b])$, die derselben Randbedingung genügt*

$$\|s''\|_2 \leq \|y''\|_2$$

6 Interpolation

Beweis. Die Anzahl der Forderungen ist $N = \dim S_{4,\Delta}$. Es ergibt sich also ein quadratisches lineares Gleichungssystem und es genügt Injektivität zu zeigen.

Sei $f \equiv 0$, dann erfüllt $y \equiv 0$ alle Forderungen. für alle Lösungen $s \in S_{4,\Delta}$ gilt daher

$$\|s''\|_2 \leq \|y''\|_2 = 0 \quad \text{nach 6.2.16}$$

Also gilt $s'' \equiv 0$ und s ist stückweise kubische Funktion mit $s \in C^2([a, b])$ und $s(x_i) = 0$. Damit folgt $s \equiv 0$, was die Injektivität zeigt. \square

6.2.18 Berechnung der natürlichen Splines mittels B-Splines

22.12.2014

$$A := \begin{pmatrix} \frac{x_2+x_1-2x_0}{x_2-x_0} & -\frac{x_1-x_0}{x_2-x_0} & & & & & \\ N_{2,4}(x_1) & N_{3,4}(x_1) & N_{4,4}(x_1) & & & & 0 \\ & \ddots & \ddots & \ddots & & & \\ & & N_{i+1,4}(x_i) & N_{i+2,4}(x_i) & N_{i+3,4}(x_i) & & \\ & & & \ddots & \ddots & \ddots & \\ 0 & & & N_{n,4}(x_{n-1}) & N_{n+1,4}(x_{n-1}) & \ddots & N_{n+2,4}(x_{n-1}) \\ & & & & -\frac{x_n-x_{n-1}}{x_n-x_{n-2}} & & \frac{2x_n-x_{n-2}-x_{n-1}}{x_n-x_{n-2}} \end{pmatrix}$$

Die Werte $N_{j,4}(x_i)$ lassen sich nach 6.2.8 berechnen.

Insbesondere ergibt sich für äquidistantes Gitter

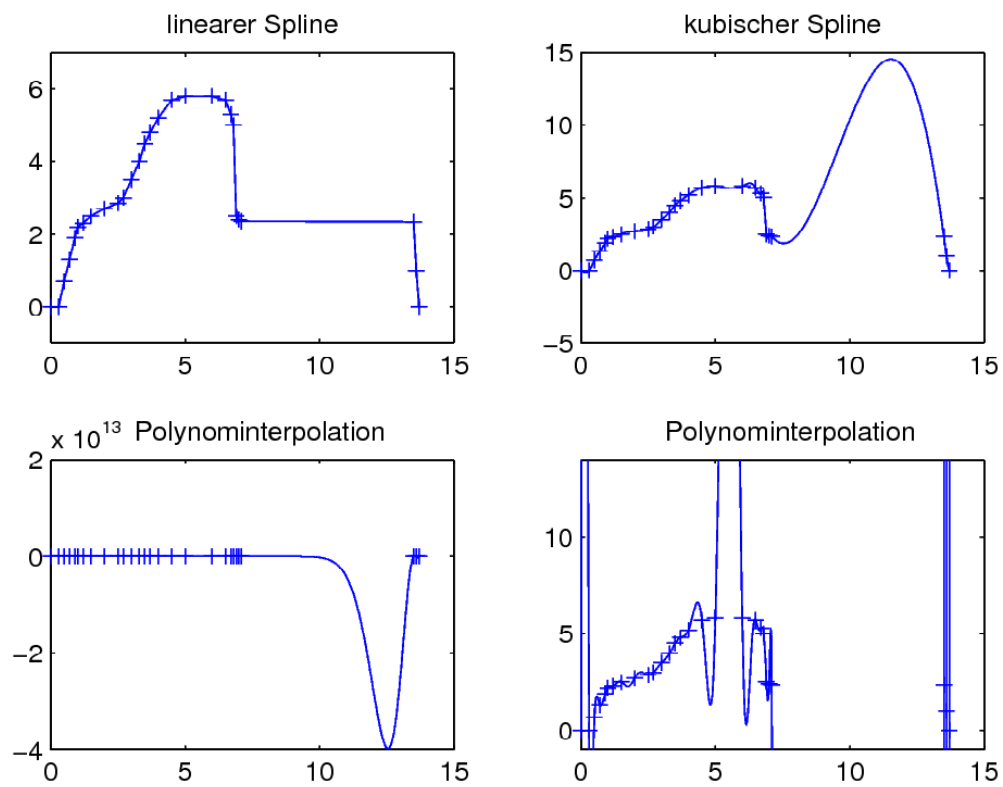
$$\frac{1}{12} \begin{pmatrix} 18 & -6 & & & & & \\ 3 & 7 & 2 & & & & \\ & 2 & 8 & 2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 2 & 8 & 2 & \\ & & & & 2 & 7 & 3 \\ & & & & & -6 & 18 \end{pmatrix} \begin{pmatrix} c_2 \\ \vdots \\ c_{n+2} \end{pmatrix} = \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix}$$

(siehe Übung).

Beispiel 6.2.19 (Laster). Das folgende Beispiel zeigt, dass sowohl die Glattheit der Funktion wie auch die Knotenverteilung einen Einfluß hat, da die zugehörige Funktion nicht differenzierbar ist, bzw. die Ableitungen betragsmäßig sehr groß sind.

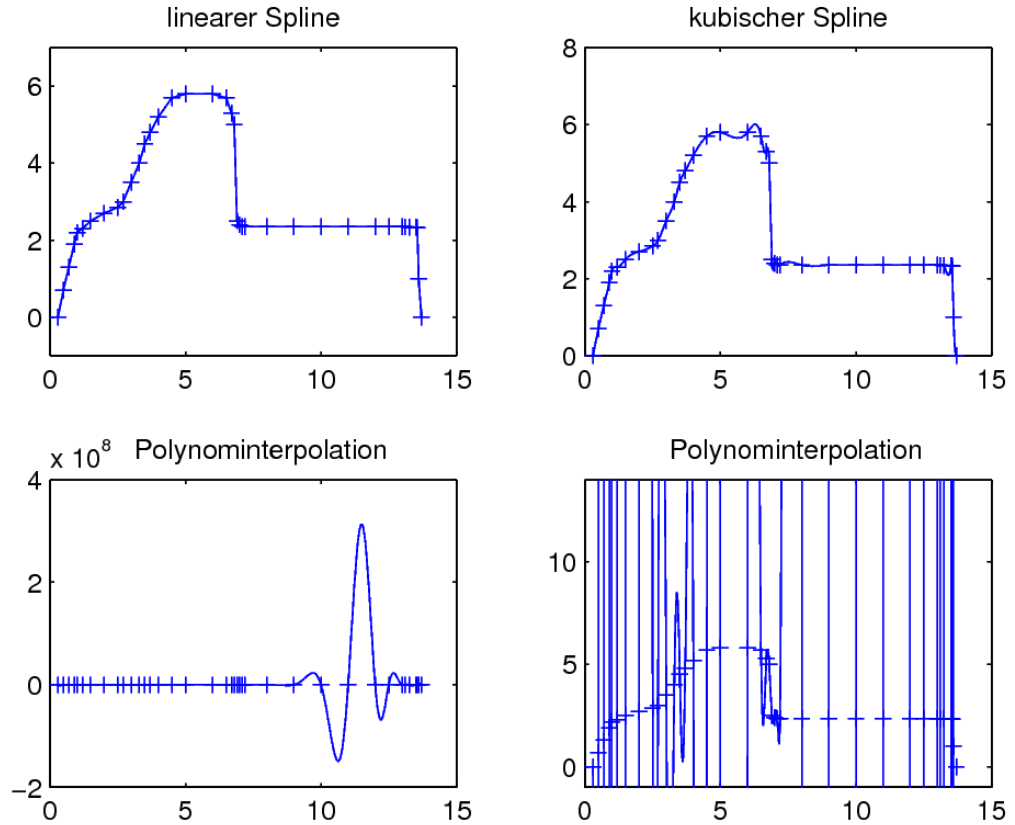
Natürliche Spline-Interpolation versus polynomiale Interpolation:

6.2 Stückweise polynomiale Approximation durch Splines



Mit mehr Stützstellen:

6 Interpolation



Satz 6.2.20 (Fehlerabschätzung für kubische Splines). Sei $f \in C^4([a, b])$ und $s \in S_{4,\Delta}$ der interpolierende, vollständige, kubische Spline zu äquidistanten Knoten mit $h = x_{i+1} - x_i$. Dann gilt für $k = 0, 1, 2, 3$

$$\|f^{(k)} - s^{(k)}\|_{\infty} \leq c_k h^{4-k} \|f^{(4)}\|_{\infty}$$

mit $c_0 = \frac{5}{384}$, $c_1 = \frac{1}{24}$, $c_2 = \frac{3}{8}$, $c_3 = 1$, wobei c_0 und c_1 optimal sind.

Beweis. [siehe z.B. SB90, (in allgemeiner Form, d.h. auch für nicht gleichmäßiges Gitter)] Die optimalen c_k wurden von Hall und Meyer (1976) bewiesen. Oft wird mit Stetigkeitsmodulen gearbeitet. \square

Für eine wachsende Anzahl der Knoten ergibt sich also Konvergenz, anders als bei polynomialer Interpolation.

Satz 6.2.21 (Fehlerabschätzung für lineare Splines). Sei $f \in C^2([a, b])$ und $s \in S_{2,\Delta}$ der interpolierende Spline an den Stützstellen $a = x_0 < x_1 < \dots < x_n = b$. Dann gilt mit $h = \max_{i=0, \dots, n-1} |x_{i+1} - x_i|$

$$\|f - s\|_{\infty} \leq \frac{h^2}{8} \|f''\|_{\infty}.$$

6.2 Stückweise polynomiale Approximation durch Splines

Beweis. siehe Übungsaufgabe

□

Weitere Literatur zu Splines: [Boo01]

7 Numerische Integration/Quadratur

7.1 Einführung und einfache Quadraturformeln

Häufig sind Integrale

$$I(f) := \int_a^b f(t) dt$$

nicht analytisch zu berechnen oder die benötigte analytische Darstellung des unbestimmten Integrals ist kompliziert. Ziel ist $I(f)$ durch eine **Numerische Quadratur**

$$\hat{I}(f)$$

zu approximieren.

Der Begriff numerische Integration ist allgemeiner und beschreibt das numerische Lösen von

$$y'(t) = f(t, y(t)) \quad \text{mit } y(a) = c$$

Die numerische Quadratur $\hat{I}(f)$ sollte

- a) effizient ausführbar sein,
- b) viele Eigenschaften der Integration beibehalten und
- c) $I(f)$ gut approximieren.

Zu c) einige Eigenschaften des Riemann-Integrals:

Seien $f, g \in C([a, b])$, dann ist $I : C([a, b]) \rightarrow \mathbb{R}$ eine positive Linearform, d.h. I ist linear und

$$f \geq 0 \quad \Rightarrow \quad I(f) \geq 0 \tag{7.1.1}$$

Weiterhin gilt

$$\int_a^b f(t) dt + \int_b^c f(t) dt = \int_a^c f(t) dt \tag{7.1.2}$$

Diese sollen nun möglichst beibehalten werden.

Definition 7.1.1. Ein Verfahren heißt konvergent von der Ordnung p (hat die Konvergenzordnung p), falls der Fehler (hier $|I(f) - \hat{I}(f)|$) mit $\mathcal{O}(h^p)$ gegen Null geht für $h \rightarrow 0$.

7.1.2 Mittelpunktregel



Veranschaulichung der Mittelpunktregel

Zerlege $[a, b]$ in Teilintervalle $[x_i, x_{i+1}]$ für $i = 0, \dots, n-1$ der Länge $h_i = x_{i+1} - x_i$, d.h.

$$a = x_0 < x_1 < \dots < x_n = b$$

und setze zur Approximation von $I(f)$

$$\hat{I}_M(f) := \sum_{i=0}^{n-1} f\left(\frac{x_{i+1} + x_i}{2}\right) \cdot (x_{i+1} - x_i) \quad (7.1.3)$$

Für ein äquidistantes Gitter ergibt sich mit $h = \frac{b-a}{n}$

$$\hat{I}_M(f) = h \sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right) \quad (7.1.4)$$

Die **Fehlerschranke** (Konvergenzordnung) für $f \in C^2([a, b])$

$$|I(f) - \hat{I}_M(f)| \leq \frac{h^2}{24} (b-a) \|f''\|_\infty \quad (7.1.5)$$

ergibt sich aus der Taylorentwicklung

$$f(t) = f\left(x_i + \frac{h}{2}\right) + f'\left(x_i + \frac{h}{2}\right) \cdot \left(t - \left(x_i + \frac{h}{2}\right)\right) + \frac{1}{2} f''(\xi_i(t)) \cdot \left(t - \left(x_i + \frac{h}{2}\right)\right)^2$$

mit einer Zwischenstelle $\xi_i(t)$ zwischen t und $\left(x_i + \frac{h}{2}\right)$, denn

$$\begin{aligned} |I(f) - \hat{I}_M(f)| &\leq \sum_{i=0}^{n-1} \left| \int_{x_i}^{x_{i+1}} f(t) dt - h \cdot f\left(x_i + \frac{h}{2}\right) \right| \\ &= \left| \int_{x_i}^{x_{i+1}} \left(f(t) - f\left(x_i + \frac{h}{2}\right) \right) dt \right| \end{aligned}$$

da $\int_{x_i}^{x_{i+1}} \left(t - \left(x_i + \frac{h}{2}\right) \right) dt = 0$

$$\begin{aligned} &= \sum_{i=0}^{n-1} \left| \int_{x_i}^{x_{i+1}} \frac{1}{2} f''(\xi_i(t)) \cdot \left(t - \left(x_i + \frac{h}{2}\right) \right)^2 dt \right| \\ &\leq \|f''\|_\infty \cdot \frac{1}{2} \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} \left(t - \left(x_i + \frac{h}{2}\right) \right)^2 dt \\ &= \frac{h^2}{24} (b-a) \|f''\|_\infty \end{aligned}$$

Für andere Interpolationspunkte als den Mittelpunkt (z.B. einer der Eckpunkt) wird $\mathcal{O}(h^2)$, d.h. die Ordnung 2, nicht erreicht.

7.1.3 Trapezsumme

Interpoliere stückweise linear



Veranschaulichung Trapezsumme

$$(x_{i+1} - x_i) \cdot \frac{f(x_i) + f(x_{i+1})}{2} \quad (\text{Trapezregel})$$

$$\Rightarrow \hat{I}_T(f) = \frac{1}{2} \sum_{i=0}^{n-1} (f(x_i) + f(x_{i+1})) \cdot (x_{i+1} - x_i) \quad (7.1.6)$$

und für äquidistante Gitter ergibt sich die **Trapezsumme**

$$\hat{I}_T(f) = h \cdot \left(\frac{1}{2} f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2} f(x_n) \right) \quad (7.1.7)$$

Die Konvergenz folgt durch die Riemannschen Unter- bzw. Obersummen $R_u^{(k)}(f)$ und $R_o^{(k)}(f)$ wegen

$$I(f) \leftarrow R_u^{(k)}(f) \leq \hat{I}_T(f) \leq R_o^{(k)}(f) \rightarrow I(f) \quad \text{für } h \rightarrow 0$$

Später zeigen wir noch die Konvergenzordnung für $f \in C^2([a, b])$

$$|I(f) - \hat{I}_T(f)| \leq \frac{h^2}{24} (b-a) \|f''\|_\infty \quad (7.1.8)$$

Offensichtlich erfüllen $\hat{I}_M(f)$ und $\hat{I}_T(f)$ die Eigenschaften aus 7.1.1. Weiterhin sind sie für lineare Funktionen exakt, d.h. berechnen die genauen Integralwerte.

7.2 Interpolatorische Integrationsformeln

(auch Newton-Cotes-Formeln)

Die Idee ist nun, f auf $[a, b]$ polynomial zu interpolieren an verschiedenen Stellen $a \leq x_0 \leq x_1 \leq \dots \leq x_n < b$ und dann das Integral zu berechnen, d.h.

$$\begin{aligned} \hat{I}(f) &:= I(p(f | x_0, \dots, x_n)) \\ &= \sum_{i=0}^n f(x_i) \cdot \int_a^b L_i(t) dt \\ &= \sum_{i=0}^n w_i f(x_i) \end{aligned} \quad (7.2.1)$$

mit den Gewichten

$$w_i := \int_a^b L_i(t) dt. \quad (7.2.2)$$

Offensichtlich sind diese Quadraturformeln exakt für Polynome vom Grad weniger oder gleich n .

Es gilt sogar:

Lemma 7.2.1. *Zu $n + 1$ paarweise verschiedenen Knoten x_0, \dots, x_n gibt es genau eine Quadraturformel*

$$\hat{I}(f) = \sum_{i=0}^n w_i f(x_i)$$

mit $w_i \in \mathbb{R}$, welche für alle Polynome $p \in \mathcal{P}_n$ exakt ist.

Beweis. $L_i \in \mathcal{P}_n$, also

$$\begin{aligned} I(L_i) &= \hat{I}(L_i) = \sum_{j=0}^n w_j L_i(x_j) \\ &= \sum_{j=0}^n w_j \delta_{ij} \\ &= w_i \end{aligned} \quad \text{für } i = 0, \dots, n$$

Die Gewichte sind damit eindeutig bestimmt. □

Definition 7.2.2. \hat{I} heißt **abgeschlossene Quadraturformel**, falls $a = x_0$ und $b = x_n$. \hat{I} heißt **offen**, falls $a < x_0$ und $x_n < b$. (\hat{I}_M ist also eine offene Quadraturformel.)

7.2.3 Newton-Cotes-Formeln

Abgeschlossene, interpolatorische Quadraturformeln mit äquidistanter Stützstellenverteilung x_0, \dots, x_n heißen **Newton-Cotes-Formeln der Ordnung n** . Mit $h = \frac{b-a}{n}$, $x_i = a + i \cdot h$ und Substitution $z := \frac{t-a}{h}$ gilt

$$w_i := h \cdot \int_0^n \prod_{\substack{j=0 \\ j \neq i}}^n \frac{z-j}{i-j} dz. \quad (7.2.3)$$

Diese sind unabhängig von der Lage der Stützstellen und liegen in Form von

$$w_i = \frac{b-a}{n \cdot s} \sigma_i$$

tabellarisch vor. Es gilt $n \cdot s$, $\sigma_i \in \mathbb{Z}$ und

$$\hat{I}_n(f) = \frac{(b-a)}{n \cdot s} \cdot \sum_{i=0}^n \sigma_i f(x_i). \quad (7.2.4)$$

Da sie exakt sind für $p \in \mathcal{P}_n$, können $\frac{\sigma_i}{n \cdot s}$ z.B. aus dem linearen Gleichungssystem für $[a, b] = [0, 1]$

$$\int_0^1 x^k dx = \frac{1}{n \cdot s} \sum_{i=0}^n \sigma_i x_i^k \quad \text{für } k = 0, \dots, n$$

7.2.4 Tabelle (Newton-Cotes-Gewichte)

$n \backslash \sigma_i$	σ_1	...				$n \cdot s$	
1	1	1				2	Trapezregel
2	1	4	1			6	Simpson-Regel
3	1	3	3	1		8	$\frac{3}{8}$ -Regel
4	7	32	12	32	7	90	Milue-Regel
\vdots							
8		...	-4540	...			

Positive Funktionen haben kein positives Integral ab einem n_0 !!

07.01.2015

Daraus folgt, dass die Monotonie (Positivität) bei abgeschlossenen Newton-Cotes Formeln i.A. nicht gewährleistet ist, wenn Polynome von hohem Grad exakt integriert werden sollen. Diese sind dann daher ungeeignet.

Satz 7.2.5. Sei $\hat{I}_n(f)$ die abgeschlossene Newton-Cotes-Formel der Ordnung n . Dann gilt:

a) $\sum_{i=0}^n w_i = b - a$

b) $w_i = w_{n-i}$

c) Ist n gerade, so ist \hat{I}_n exakt für Polynome bis zum Grad $n+1$ nicht nur n .

Beweis. siehe Übungsblatt

□

Satz 7.2.6. Sei $\hat{I}(f) := \sum_{i=0}^n w_i f(x_i)$ mit $w_i \in \mathbb{R}$ und $a \leq x_0 < x_1 < \dots < x_n \leq b$. Dann ist der Integrationsfehler R mit

$$R(f) := \hat{I}(f) - \int_a^b f(x) dx \quad (7.2.5)$$

7 Numerische Integration/Quadratur

eine lineare Abbildung, für die folgende Darstellungsformel gilt:
Falls $R(p) = 0$ für alle $p \in \mathcal{P}_l$, dann gilt für alle $f \in C^{l+1}([a, b])$

$$R(f) = \int_a^b f^{(l+1)}(t) K(t) dt \quad (7.2.6)$$

mit

$$\begin{aligned} K(t) &:= \frac{1}{l!} R((\bullet - t)_+^l) \\ &= \frac{1}{l!} \left(\sum_{i=0}^n w_i (x_i - t)_+^l - \int_a^b (x - t)_+^l dx \right) \end{aligned} \quad (7.2.7)$$

$K(t)$ heißt **Peano-Kern des Funktionals R** .

Beweis. Es gilt

$$f(x) = f(a) + f'(a)(x-a) + \dots + \frac{f^{(l)}(a)}{l!} (x-a)^l + r_l(x)$$

mit Restglied

$$\begin{aligned} r_l(x) &= \frac{1}{l!} \int_a^x f^{(l+1)}(t) \cdot (x-t)^l dt \\ &= \frac{1}{l!} \int_a^b f^{(l+1)}(t) \cdot (x-t)_+^l dt \end{aligned}$$

Da $R(p) = 0$ für alle $p \in \mathcal{P}_l$ folgt

$$\begin{aligned} R(f) &= R(r_l) = \frac{1}{l!} R\left(\int_a^b f^{(l+1)}(t) \cdot (\bullet - t)_+^l dt\right) \\ &= \frac{1}{l!} \left\{ \sum_{i=0}^n w_i \int_a^b f^{(l+1)}(t) \cdot (x_i - t)_+^l dt - \int_a^b \int_a^b f^{(l+1)}(t) \cdot (x - t)_+^l dt dx \right\} \end{aligned}$$

Mit dem Satz von Fubini gilt

$$\begin{aligned} R(f) &= \frac{1}{l!} \int_a^b f^{(l+1)}(t) \left\{ \sum_{i=0}^n w_i \cdot (x_i - t)_+^l - \int_a^b (x - t)_+^l dx \right\} dt \\ &= \frac{1}{l!} \int_a^b f^{(l+1)}(t) \cdot R((\bullet - t)_+^l) dt \end{aligned}$$

□

Folgerung 7.2.7. Falls der Peano-Kern K auf $[a, b]$ ein konstantes Vorzeichen hat, gilt nach dem Mittelwertsatz der Integralrechnung

$$R(f) = f^{(l+1)}(\xi) \int_a^b K(t) dt \quad \text{für ein } \xi \in (a, b) \quad (7.2.8)$$

7.2 Interpolatorische Integrationsformeln

Wähle nun ein g , für das $R(f)$ und $g^{(l+1)}$ bekannt oder leicht berechenbar sind, z.B. $g(x) := x^{l+1}$. Hier ist

$$\int_a^b K(t) dt = \frac{R(g)}{(l+1)!} = \frac{R(x^{l+1})}{(l+1)!}$$

Dann gilt für ein beliebiges $f \in C^{l+1}([a, b])$

$$R(f) = \frac{R(x^{l+1})}{(l+1)!} f^{(l+1)}(\xi) \quad \text{für ein } \xi \in (a, b) \quad (7.2.9)$$

Satz 7.2.8 (Approximationsfehler). *Im Fall der Newton-Cotes-Formeln besitzen die Peano-Kerne konstante Vorzeichen. Somit gilt für $f \in C^{n+j+1}$ mit $j = \begin{cases} 0 & \text{für } n \text{ ungerade} \\ 1 & \text{für } n \text{ gerade} \end{cases}$*

$$\begin{aligned} R_n(f) &= \hat{I}_n(f) - \int_a^b f(x) dx \\ &= \frac{R(x^{n+j+1})}{(n+j+1)!} f^{(n+j+1)}(\xi) \end{aligned} \quad (7.2.10)$$

Beweis. Der Beweis des konstanten Vorzeichens gilt [steffensen]. Nach Satz 7.2.5c) gilt:

$$R_n(p) = 0 \quad \forall p \in \mathcal{P}_{n+j}$$

Mit (7.2.9) und $l = n + j$ folgt dann die Aussage (7.2.10) □

7.2.9 Tabelle: Fehler für Newton-Cotes-Formeln

n		$R_n(f)$
1	Trapezregel	$(b-a)^3 \cdot \frac{1}{12} f^{(2)}(\xi)$
2	Simpsonregel	$\frac{(b-a)^5}{2} \cdot \frac{1}{90} f^{(4)}(\xi)$
3	$\frac{3}{8}$ -Regel	$\frac{(b-a)^5}{3} \cdot \frac{3}{80} f^{(4)}(\xi)$
4	Milue-Regel	$\frac{(b-a)^7}{4} \cdot \frac{8}{945} f^{(6)}(\xi)$

Die Erhöhung des Polynomgrades n ergibt also i.A. keine bessere Approximation. Dies hängt auch von der Länge des Intervalls $[a, b]$ ab. Stattdessen werden die Newton-Cotes-Formeln auf Teilintervalle angewendet, d.h. sie bilden die Grundlage für stückweise polynomiale interpolatorische Integrationsformeln.

7.2.10 Iterierte (wiederholte) Newton-Cotes-Formeln

Betrachte die Intervallunterteilung

$$a = t_0 < t_1 < \dots < t_N = b$$

und approximiere

$$I(f) = \sum_{l=0}^{N-1} \int_{t_l}^{t_{l+1}} f(t) dt$$

durch wiederholtes Anwenden einer Newton-Cotes-Formel $\hat{I}_n(f)$ auf die Teilintervalle $[t_l, t_{l+1}]$ ($\hat{I}_n(f)$), d.h.

$$\hat{I}(f) = \sum_{l=0}^{N-1} \hat{I}_n^{(l)}(f) = \sum_{l=0}^{N-1} \frac{t_{l+1} - t_l}{n \cdot s} \sum_{i=0}^n \sigma_i f(t_l + i\tilde{h}_l) \quad (7.2.11)$$

mit $\tilde{h}_l := (t_{l+1} - t_l) \cdot \frac{1}{n}$. Für äquidistante Knoten $t_i = a + ih$ mit $h = \frac{b-a}{N}$ ergibt die Anwendung der Trapezregel ($n = 2$) die Trapezsumme $\hat{I}_T(f) = h \left(\frac{1}{2} f(t_0) + \sum_{i=1}^{N-1} f(t_i) + \frac{1}{2} f(t_N) \right)$ nach (7.1.7). Ferner lässt sich die Konvergenzordnung (7.1.8) zeigen.

Lemma 7.2.11. Sei $f \in C^2([a, b])$, dann existiert ein $\tau \in [a, b]$, so dass

$$\hat{I}_T(f) - I(f) = \frac{b-a}{12} h^2 f''(\tau) \quad (7.2.12)$$

Beweis. Es gilt $t_{l+1} - t_l = h = \frac{b-a}{N}$. Daraus folgt

$$\hat{I}_T(f) - I(f) = (t_{l+1} + t_l)^3 \cdot \frac{1}{12} f''(\xi_l)$$

nach Tabelle 7.2.9 und mit $\xi_l \in [t_l, t_{l+1}]$. Weiterhin gilt dann

$$\begin{aligned} \hat{I}_T(f) - I(f) &= h^3 \frac{1}{12} \sum_{i=0}^{N-1} f''(\xi_i) \\ &= \frac{b-a}{12} h^2 \cdot \underbrace{\frac{1}{N} \sum_{i=0}^{N-1} f''(\xi_i)}_{:=d} \\ &= \frac{b-a}{12} h^2 f''(\tau) \end{aligned}$$

da mit $\min_{z \in [a, b]} f''(z) \leq d \leq \max_{z \in [a, b]} f''(z)$ aus dem Zwischenwertsatz die Existenz eines $\tau \in [a, b]$ folgt mit $f''(\tau) = d$ \square

Die Trapezsumme ist also ein Quadraturverfahren zweiter Ordnung (Polynome vom Grad < 2 werden exakt integriert) und der Integrationsfehler konvergiert mit der Ordnung 2 (wegen h^2) gegen 0 für $N \rightarrow \infty$.

7.2.12 Iterierte (oder summierte) Simpsonregel (Keplersche Fassregel)

$$\hat{I}_S(f) = \frac{1}{6} \sum_{l=0}^{N-1} (t_{l+1} - t_l) \cdot \left(f(t_l) + 4f\left(\frac{t_l + t_{l+1}}{2}\right) + f(t_{l+1}) \right) \quad (7.2.13)$$

Für äquidistantes Gitter ergibt sich daher für $t_l = a + l \cdot h$

$$\hat{I}_S(f) = \frac{1}{8} h \cdot \left\{ f(t_0) + 4f\left(\frac{t_1 + t_0}{2}\right) + 2f(t_1) + 3f\left(\frac{t_2 + t_1}{2}\right) + \dots + 2f(t_{N-1}) + 4f\left(\frac{t_{N-1} + t_N}{2}\right) \right\} \quad (7.2.14)$$

und wie eben die Fehlerabschätzung für $f \in C^4([a, b])$

$$\hat{I}_S(f) - I(f) = \frac{b-a}{2880} h^4 f^{(4)}(\tau) \quad \text{mit } \tau \in [a, b] \quad (7.2.15)$$

Zu beachten ist, dass aufgrund von $f\left(\frac{t_l + t_{l+1}}{2}\right)$ doppelt so viele Funktionsauswertungen nötig sind wie bei der Trapezsumme, d.h. $2N + 1$ insgesamt!

Bei einer geraden Anzahl N von Intervallen kann mit der Simpsonregel jedoch eine iterierte Regel aufgestellt werden, welche wie die Trapezregel $N + 1$ Funktionsauswertungen benötigt. Wende hierfür die Simpsonregel auf $[t_{2l}, t_{2l+2}]$ an bzw. setze $h = 2\frac{b-a}{N}$.

Es ergibt sich mit $\bar{h} = \frac{b-a}{N}$

$$S(f) = \frac{\bar{h}}{3} [f(t_0) + 4f(t_1) + 2f(t_2) + 4f(t_3) + \dots + 2f(t_{N-1}) + 4f(t_{N-1}) + f(t_N)] \quad (7.2.16)$$

und es gilt die Fehlerabschätzung für $f \in C^4([a, b])$

$$S(f) - I(f) = \frac{b-a}{180} \cdot \bar{h}^4 f^{(4)}(\tau) \quad \text{mit } \tau \in [a, b] \quad (7.2.17)$$

da $\bar{h} = 2h$.

7.3 Extrapolationsmethode und klassische Rombert-Quadratur

Basiert auf der asymptotischen Entwicklung des Approximationsfehlers der Trapezsumme:

Satz 7.3.1 (Euler-Maclaurinsche Summenformel). Sei $f \in C^{2n+2}([a, b])$ und $\hat{I}_T^h(f)$ die Trapezsumme für äquidistante Knoten $x_l = a + lh$ mit $h = \frac{b-a}{N}$, d.h.

$$\hat{I}_T^h(f) = h \left\{ \frac{1}{2} f(a) + \sum_{i=1}^{N-1} f(a + lh) + \frac{1}{2} f(b) \right\} \quad (7.3.1)$$

Dann gilt

$$T(h) := \hat{I}_T^h(f) = \tau_0 + \tau_1 h^2 + \tau_2 h^4 + \dots + \tau_n h^{2n} + R_{n+1}(h) h^{2n+1} \quad (7.3.2)$$

wobei

$$\tau_0 = I(f) = \int_a^b f(x) dx \quad (7.3.3)$$

das gesuchte Integral ist, und

$$\tau_k = \frac{B_{2k}}{(2k)!} \left(f^{(2k-1)}(b) - f^{(2k-1)}(a) \right) \quad k = 1, \dots, n$$

mit den Bernoulli-Zahlen B_{2k} sind, und

$$R_{n+1}(h) = \frac{B_{2n+2}}{(2n+2)!} (b-a) f^{(2n+2)}(\xi(h))$$

mit $a < \xi(h) < b$ eine in h gleichmäßige, beschränkte Funktion ist.

Beweis. Der Beweis ist in [SB90] oder [HH94] zu finden. □

7.3.2 Idee der Extrapolation

Gesucht ist $T(0) = \tau_0 = I(f)$. Berechne hierfür für paarweise verschiedene h_0, \dots, h_n die Werte $T(h_i)$, bestimme dann das Interpolationspolynom

$$p(x) = a_0 + a_1 x + \dots + a_n x^n \quad (7.3.4)$$

mit

$$p(h_i^2) = T(h_i) = \hat{I}_T^{h_i}(f) \quad \text{für } i = 0, \dots, n \quad (7.3.5)$$

Dann ist $p(h^2)$ eine Approximation der Funktion $T(h)$ auf $[\min_i h_i, \max_i h_i]$. „Extrapoliere“ nun von diesem Intervall auf die Stelle $h = 0$ und nutze den extrapolierten Wert $p(0)$ als Näherung für $T(0) = I(f)$.



Veranschaulichung von Extrapolation

Index

- Äquilibrierung
 - Spalten-, 55
 - Zeilen-, 55
- abgebrochene Potenz, 112
- Abstiegsrichtung, 94
- affin-invariant, 90
- Algorithmus, 48
- Approximationsfehler, 133
- Ausgleichsproblem
 - linear, 59, 61
- Banachscher Fixpunktsatz, 81
- Basis, 30
- Bildraum, 62
- Bisektionsverfahren, 78
- Broyden-Verfahren, 93
- Cholesky-Zerlegung, 58, 59, 64
- de Boor-Punkte, 118
- dividierte Differenzen, 101
 - Fehlerdarstellung, 106
 - Leibnizregel, 103
 - Schema, 102
 - verallgemeinerte Form, 105
- double, 30
- Dreieckszerlegung, 13, 17
- elementar ausführbar, 47
- entkoppelt, 43
- Euler-Maclaurinsche Summenformel, 135
- Fehler, 29, 34
 - absoluter, 34
 - absoluter Rundungsfehler, 32
 - Fortpflanzung, 48
 - relativer, 34
- Fehlerschranke, 128
- Fixpunktiteration, 78, 79
- floating point, 29, 30
- floating point operations, 17
- flops, 17
- Frobeniusmatrix, 20
- Güte
 - Algorithmus, 50
- Gauß-Eliminator, 17
- Gaußsches Eliminationsverfahren, 13, 19
- Genauigkeit
 - Maschinengenauigkeit, 32
 - relative Rechengenauigkeit, 32
- Givens-QR-Algorithmus, 69
- Givens-Rotation, 68
- Gleitkommazahl, 30
- Hamiltonsches Prinzip, 120
- Hermite-Interpolationspolynom, 104
- Horner-Schema, 100
- Householder Reflexion, 72
- Householdervektoren, 73
- Hutfunktion, 113
- Implementation, 48
- integer, 29
- Integrationsformel
 - interpolatorische/Newton-Cotes-Formeln, 129
- Interpolation
 - eigenschaft, 97
 - polynom, 98
 - Konvergenz, 109
 - Lagrange-Formel, 98

Index

- iterativer Linearisierungsprozess, 87
- Kondition
 - Addition, 37
 - gut/schlecht konditioniert, 36
 - komponentenweise, 43
 - Matrix, 40
 - normweise, absolut, 36
 - normweise. relativ, 36
- Kontraktion, 80
- Konvergenz
 - global, 83
 - linear, 83
 - lokal, 83
 - Ordnung, 82, 84
 - quadratisch, 83
 - superlinear, 83
- längenerhaltend, 65
- Lagrange-Polynome, 98
- Landau-Symbole, 18
- Legendre-Polynom, 109
- Lemma von Aitken, 99, 105
- LR-Zerlegung, 19
- Mantisse, 30
- Maximum-Likelihood-Methode, 61
- Mehrschrittverfahren, 88
- Nachiteration, 56
- Neumannsche Reihe, 41
- Neville-Schema, 100
- Newton-Cotes-Formeln
 - wiederholt, 134
- Newton-Polynome, 101
- Newton-Verfahren, 85
 - Korrekturschritt, 89
 - mehrdimensional, 89
 - Newton-Korrektur, 89
 - vereinfacht, 93
- Norm, 34, 108
 - Euklidische Norm, 34, 109
 - Frobeniusnorm, 35
 - Hölder-Norm, 34
 - Matrixnorm, 35
 - Maximumsnorm, 34
 - Spaltensummennorm, 36
 - submultiplikative, 35
 - Summennorm, 34
 - verträglich, 35
- normalisierte Gleitkommazahl, 30
- normweise Kondition, 36
- Nullstellenbestimmung, 77
- Numerische Quadratur, 127
- orthogonale Projektion, 62
- Orthogonalisierung, 66
- p-Norm, 35
- Peano-Kern, 132
- Permutationsmatrix, 23
- Pivotelement, 16
- Pivotisierung, 22
 - halbmaximale, 22
 - partielle, 22
 - Spalten-, 22
 - vollständige, 23
 - Zeilen-, 23
- Polynomraum, 98
 - Newtonsche Darstellung, 102
- Problem, 34
- Projektionssatz, 62
- Quadraturformel
 - abgeschlossen, 130
 - Ordnung, 134
- Rückwärtsanalyse, 50
- Rückwärtssubstitution, 15
- Realisierung, 47
- Rechenaufwand, 17, 18
- Restglieddarstellung, 107
- Rundungsfehler, 29
- scharf, 38
- Sekantenverfahren, 87
- Simpsonregel, 135
- Singulärwert, 36
- Skalierung
 - Spalten-, 55

- Zeilen-, 55
- spd Matrix, 56
- Spline, 111
 - B-Splines, 114
 - kubisch, 111
 - linear, 111
- Splinaumbasis, 112
 - $S_{1,\Delta}$ -Basis, 112
 - $S_{2,\Delta}$ -Basis, 113
- Stützstelle, 97
- Stützwert, 97
- Stabilität, 50
- strikt diagonal dominant, 52

- Tangentenverfahren, 85
- Toleranz, 93
- Trapezregel, 129
- Tschebyscheff-Polynom, 108
- Tschebyscheff-Punkte, 109

- unipotent, 58

- Verfahren von Crout, 21
- Vorwärtselimination, 13, 25
- Vorwärtssubstitution, 13, 15, 17

- Zahlendarstellung, 29
- Zeilensummennorm, 36

Literatur

- [Boo01] Carl de Boor. A Practical Guide to Splines. Applied Mathematical Sciences. Springer New York, 2001. ISBN: 9780387953663.
- [DH08] Peter Deuffhard und Andreas Hohmann. Numerische Mathematik. 1. 4. Aufl. de Gruyter Lehrbuch. Eine algorithmisch orientierte Einführung. Walter de Gruyter & Co., Berlin, 2008. ISBN: 978-3-11-020354-7.
- [DR08] Wolfgang Dahmen und Arnold Reusken. Numerik für Ingenieure und Naturwissenschaftler. 2. Aufl. Springer-Lehrbuch. Springer-Verlag, Berlin, 2008. ISBN: 978-3-540-76493-9.
- [FH07] R.W. Freund und R.H.W. Hoppe. Stoer/Bulirsch: Numerische Mathematik 1. Springer-Lehrbuch Bd. 1. Springer-Verlag Berlin Heidelberg, 2007. ISBN: 9783540453901. URL: <https://books.google.de/books?id=2aYfBAAAQBAJ>.
- [GO96] Gene Golub und James M. Ortega. Scientific computing. Eine Einführung in das wissenschaftliche Rechnen und Parallele Numerik, Übersetzung des englischsprachigen Originals von 1993. B. G. Teubner, Stuttgart, 1996. ISBN: 3-519-02969-3. DOI: 10.1007/978-3-322-82981-8.
- [HH94] Günther Hämmerlin und Karl-Heinz Hoffmann. Numerische Mathematik. 4. Aufl. Springer-Lehrbuch. Grundwissen Mathematik. Springer-Verlag, Berlin, 1994. ISBN: 3-540-58033-6. DOI: 10.1007/978-3-642-57894-6.
- [Pre+02] William H. Press u. a. Numerical recipes in C++. The art of scientific computing, Second edition, updated for C++. Cambridge University Press, Cambridge, 2002. ISBN: 0-521-75033-4.
- [SB90] Josef Stoer und Roland Bulirsch. Numerische Mathematik. 2. 3. Aufl. Springer-Lehrbuch. Eine Einführung—unter Berücksichtigung von Vorlesungen von F. L. Bauer. Springer-Verlag, Berlin, 1990. ISBN: 3-540-51482-1. DOI: 10.1007/978-3-662-22250-8.
- [SW05] Robert Schaback und Holger Wendland. Numerische Mathematik. Springer-Lehrbuch. Springer-Verlag Berlin Heidelberg, 2005. ISBN: 9783540267058. URL: <https://books.google.de/books?id=wdgmBAAAQBAJ>.