

Numerische Mathematik

Mitschrift der Vorlesung
bei Prof. Dr. Wilfried Gleich
2001/2002

Stand: 25. Januar 2002

Vorwort

Rechtliches

Copyright (c) Frank Abbühl (Achmed-Bilbir-Bünbur-Üzgür-Hassan-Laila)
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".¹

Anregungen

Dieses Skript ist noch lange nicht fertig. Es fehlen noch zahllose Beispiele, Bilder und Grafiken, ja sogar einige Kapitel. In vielen Fällen könnten genauere Erklärungen nicht schaden. Außerdem enthält das Skript noch zahllose inhaltliche Fehler und ist auch noch weit von einem perfekten Layout entfernt. All dies sind verbesserungswürdige Punkte und jeder, den die Lust packt, ein bißchen mit L^AT_EXherumzuxperimentieren, der sei hiermit aufgefordert, zur Verbesserung des Skriptes beizutragen. Der gesamte Quellcode ist zu finden unter <http://www.neuerezepte.de/phrank/num1.html>. Ich würde nur darum bitten, mir die entsprechend verbesserten Versionen auch wieder zukommen zu lassen.

Danksagung

Herzlich Danken möchte ich Prof. Dr. Wilfried Gleich, nach dessen Vorlesung dieses Skript erstellt wurde. Dank gebührt auch der Anita und dem Nau, die mir mit einer engelsgeduld oft wochenlang ihre Mitschriften ausgeliehen haben, weil ich mal wieder die Vorlesung verschlafen hatte. Großer Dank gebührt auch dem Daniel Egger und dem Rainer, die durch diverse Patches und ergänzungen dieses Skript um wichtige Inhalte ergänzt haben. Weiterhin möchte ich allen Danken, die mich moralisch oder durch konstruktive Kritik, Hinweise auf Fehler etcetera unterstützt haben.

Viel Spaß beim Lernen und viel Erfolg bei der Prüfung, da phrank.

¹ Der genaue Text findet sich unter <http://www.fsf.org/copyleft/fdl.html>

Inhaltsverzeichnis

1 Fehleranalyse	1
1.1 Fehlerarten	1
1.2 Maschinenzahlen und Maschinengenauigkeit	1
1.2.1 Maschinengenauigkeit	2
1.3 Rundungsfehler bei Gleitpunktoperationen	2
1.4 Fehlerfortpflanzung	2
1.5 Fehlermaßzahlen	2
1.5.1 Begriffe	2
1.5.2 Abbruchkriterien	3
1.5.3 Numerische Stabilität	3
2 Lineare Gleichungssysteme	4
2.1 Gestaffelte Gleichungssysteme	4
2.1.1 Obere Dreiecksform	4
2.1.2 Untere Dreiecksform	5
2.2 Das Gauß'sche Eliminationsverfahren	5
2.2.1 Aufgabenstellung	5
2.2.2 Der Algorithmus	6
2.2.3 Dreieckszerlegung	6
2.2.4 Pivotelementsuche	7
2.2.5 Gleichungssysteme mit mehreren Rechten Seiten	8
2.2.6 Nachiteration	8
2.3 Vektor- und Matrixnormen	9
2.4 Verträglichkeitsbedingungen	10
2.4.1 Fehlerabschätzung, "Kondition" einer Matrix	11
2.4.2 Einfluss geringfügig geänderter Eingangsdaten	11
2.4.3 Anwendung auf Pivotstrategie	12
2.5 Verfahren von Cholesky	12
2.5.1 Vorbereitungen	12
2.5.2 Herleitung des Verfahrens	13
2.5.3 Der Algorithmus	13
2.5.4 Vorgehensweise	14
2.6 Iterationsverfahren	14
2.6.1 Gesamtschrittverfahren nach Jacobi	14
2.6.2 Einzelschrittverfahren nach Seidel	15

3	Interpolation	16
3.1	Lagrange-Interpolation	16
3.1.1	Aufgabenstellung	16
3.1.2	Eindeutigkeit	16
3.1.3	Existenz	17
3.1.4	Äquidistante Stützstellen	18
3.2	Newton-Interpolation	18
3.2.1	Aufgabenstellung	18
3.2.2	Bildungsgesetz der dividierten Differenzen	19
3.2.3	Rechenschema	19
3.2.4	Algorithmus	20
3.2.5	Berechnung eines Zwischenwertes	20
3.2.6	Sonderfall: Äquidistante Stützstellen	20
3.3	Interpolation mit rationalen Funktionen	20
3.3.1	Aufgabenstellung	20
3.3.2	Herleitung	21
3.4	Spline-Interpolation	21
3.4.1	Kubische Splines	21
3.4.2	Natürliche Splines	23
3.5	Parameter-Splines	23
3.5.1	Ebene Kurven	23
3.5.2	Raumkurven	24
4	Approximation	25
4.1	Die Gauß'sche Fehlerquadratmethode	25
4.1.1	Aufgabenstellung	25
4.1.2	Aufstellen der Fehlergleichungen	26
4.2	Lösung über Normalgleichungen	26
4.2.1	Summe der Fehlerquadrate als Quadratische Form	26
4.2.2	Aufstellen der Normalgleichungen	26
4.2.3	Nachiteration	27
4.2.4	Fehlerbetrachtung	28
4.3	Orthogonalisierungsverfahren	28
4.3.1	Prinzip des Verfahrens	29
4.3.2	Das Householder-Verfahren	29
4.3.3	Fehlerbetrachtung	30
4.3.4	QR Zerlegung einer Matrix	30
5	Fourier-Analyse	32
5.1	Fourier-Analyse analytisch gegebener Funktionen	32
5.1.1	Aufgabenstellung	32
5.2	Verfahren von Goertzl	32
5.2.1	Algorithmus von Goertzel	33
5.2.2	Rechenaufwand	34
5.3	Fast Fourier Transformation (FFT)	34
5.3.1	Aufgabenstellung	34
5.3.2	Herleitung	34
5.3.3	Komplexe Darstellung	35
5.3.4	Rechenzeitbetrachtung	35

6	Numerische Integration	37
6.1	Interpolatorische Quadraturformeln	37
6.1.1	Über Lagrange-Interpolation	37
6.1.2	Berechnung der Gewichtskoeffizienten ω_i	38
6.2	Romberg-Integration	40
6.2.1	Herleitung des Romberg-Schemas	40
6.2.2	Das Romberg-Schema	41
6.2.3	Abbrechkriterium	41
6.2.4	Rechenaufwand	41
6.3	Gauß'sche Quadraturformel	43
6.3.1	Transformation auf beliebige Intervallgrenzen	43
6.4	Doppelintegrale	44
7	Numerische Lösung gewöhnlicher Differentialgleichungen	45
7.1	Aufgabenstellung	45
7.1.1	Einteilung der Verfahren	45
7.2	Verfahren von Euler/Cochoy	45
7.3	Verbessertes Polygonzugverfahren	46
7.4	Verfahren von Runge-Kutta	46
7.5	Systeme von Differentialgleichungen 1. Ordnung	46
7.5.1	Differentialgleichungen höherer Ordnung	46
7.5.2	Polygonzugverfahren	47
7.5.3	Verfahren von Runge-Kutta	47
8	Lösung nichtlinearer Gleichungen	48
8.1	Interallhalbierung	48
8.2	Regula Falsi	49
8.3	Newton-Verfahren	50
8.4	Picard-Iterationsverfahren	50
	Aufgabentypen	52
	Taylor-Reihen	53

Kapitel 1

Fehleranalyse

1.1 Fehlerarten

Man unterscheidet folgende Kategorien von Fehlern:

- Fehler in den Eingabewerten, zum Beispiel den Meßdaten
- *Rundungsfehler* aufgrund beschränkter Rechengenauigkeit
- *Verfahrensfehler*, entstehend durch numerische Verfahren
- *Abbrechfehler*, ein Spezialfall von Verfahrensfehlern, der durch das Abbrechen eines ansonsten unendlichen Algorithmus entsteht

1.2 Maschinenzahlen und Maschinengenauigkeit

Ganze Dualzahlen nennt man *Integerzahlen*. Zur näherungsweise Darstellung von reellen Zahlen verwendet man die sogenannten *Gleitpunktzahlen*. Diese werden intern in Form von *Mantisse* und *Exponent* dargestellt:

m_0	m_1	m_2	\dots	m_i	e_0	e_1	e_2	\dots	e_k
Mantisse					Exponent				

$$z = \text{Mantisse} \cdot 2^{\pm \text{Exponent}}$$

Die Mantisse bestimmt die Rechengenauigkeit, der Exponent den Zahlenbereich. Indem festgelegt wird, daß immer gilt: $\frac{1}{2} \leq \text{Mantisse} < 1$ wird ein zusätzliches Bit eingespart.

Aus der Forderung:

$$2^x = 10; x = 3.32$$

folgt:

d binären Mantissestellen entsprechen $\frac{d}{3.32}$ Dezimalstellen

1.2.1 Maschinengenauigkeit

Sei A die Menge aller Maschinenzahlen (endlich)

Ist $x \in R$, so bezeichnen wir mit

$rd(x)$: Approximation von x durch die am nächsten benachbarte Maschinenzahl; $rd(x) \in A$, Runden im üblichen Sinn.

Bei t verfügbaren dezimalen Mantissenstellen ist

$$|x - rd(x)| \leq 0.5 \cdot 10^{-t}$$

Definition 1.1 (Maschinengenauigkeit)

$$\varepsilon_{masch} = \frac{1}{2} \cdot 10^{-t}$$

oder mit anderen Worten:

$$\varepsilon_{masch} = \min\{g \in A \mid 1 + g > 1 \text{ und } 1 + g \in A\}$$

... die kleinste Zahl, die zu 1 addiert eine Maschinenzahl > 1 ergibt.

1.3 Rundungsfehler bei Gleitpunktoperationen

Sind $x, y \in A$, so sind $x + y$, $x - y$, $x \cdot y$, x/y im allgemeinen keine Maschinenzahlen, es treten somit Rundungsfehler auf. Achtung: Die Differenz fast gleich großer Zahlen kann zu katastrophalen Resultaten führen!

1.4 Fehlerfortpflanzung

*Konditionszahlen*¹ sind Verstärkungsfaktoren der relativen Eingangsfehler. Bei sehr großen Konditionszahlen spricht man von einem "schlecht konditionierten Problem", bei Konditionszahlen nahe der Eins spricht man von einem "gut konditionierten Problem"².

Die Rechenoperationen Addition, Multiplikation und Division sind sehr gut konditioniert, die Subtraktion ist für fast gleich große Werte sehr schlecht konditioniert.

1.5 Fehlermaßzahlen

1.5.1 Begriffe

Gegeben sind der exakte Wert x_{exakt} und der Näherungswert $x_{\text{naeherung}}$.

Definition 1.2 (absoluter Fehler)

$$|\Delta x| = |x_{\text{naeherung}} - x_{\text{exakt}}|$$

¹siehe Anhang A

²10 ist in der Nähe von 1

Definition 1.3 (relativer Fehler)

$$\left| \frac{\Delta x}{x_{\text{exakt}}} \right| = \left| \frac{x_{\text{naeherung}} - x_{\text{exakt}}}{x_{\text{exakt}}} \right|$$

Beispiel 1.1 Gegeben: Eine von n Variablen abhängige Funktion $f(x_1, x_2, \dots, x_{\text{naeherung}})$ sowie die Fehler in x_1, x_2, \dots, x_n . Gesucht wird der Absolute Fehler ε_f .

$$\varepsilon_f = \sum_{i=1}^n \left| \frac{\delta f}{\delta x_i} \cdot \frac{x_i}{f} \right| \left| \frac{\Delta x_i}{x_i} \right|$$

1.5.2 Abbruchkriterien

Sei $x_0, x_1, x_2, \dots, x_k$ eine Folge von Näherungswerten für x_{exakt} . Dann verwendet man für den exakten Wert x_{exakt} den bisher besten Wert der Folge (x_k) und als Näherungswert $x_{\text{naeherung}}$ den zweitbesten (x_{k-1}) .

$$|\varepsilon_x| := \left| \frac{\Delta x}{x_{\text{exakt}}} \right| = \left| \frac{x_k - x_{k-1}}{x_{\text{exakt}}} \right|$$

Abbruchkriterium der Folge $x_0, x_1, \dots, x_{\text{naeherung}}$ falls

$$\left| \frac{x_{\text{naeherung}} - x_{k-1}}{x_k} \right| < \varepsilon$$

mir $\varepsilon \geq \varepsilon_{\text{masch}}$ (Maschinengenauigkeit)

1.5.3 Numerische Stabilität

Definition 1.4 (numerische Stabilität) Ein Algorithmus für ein numerisches Verfahren heißt stabil, wenn ein für einen Rechenschritt zugelassener Fehler in den Folgeschritten abnimmt oder in der gleichen Größenordnung bleibt.

Kapitel 2

Lineare Gleichungssysteme

Die Aufgabenstellung, der sich dieses Kapitel widmet, ist die Lösung von linearen Gleichungssystemen der Form $A\vec{x} = \vec{b}$, wobei eine Koeffizientenmatrix $A \in \mathbb{R}^{n \times n}$ und der Lösungsvektor $\vec{b} \in \mathbb{R}^n$ gegeben sind. Gesucht wird der Vektor $\vec{x} \in \mathbb{R}^n$.

Da der Rechenaufwand für die *Cramer'sche Regel* mit $(n^2 - 1)n! + n$ Operationen für die numerische Praxis ungeeignet ist, wird auf diese nur beim Lösen von Gleichungssystemen mit Parametern zurückgegriffen, die ansonsten nicht lösbar wären.

Man unterscheidet

- *direkte Verfahren* Die Lösung wird in einer zuvor bekannten Anzahl von Rechenoperationen erzielt. Solche Verfahren sind fast immer anwendbar (und meistens mit dem Namen Gauß verknüpft).
- *indirekte Verfahren* umfassen theoretisch unendlich viele Rechenoperationen und sind nur für Matrizen mit speziellen Eigenschaften anwendbar. Abbruch des numerischen Verfahrens über Konvergenzabfragen.

2.1 Gestaffelte Gleichungssysteme

2.1.1 Obere Dreiecksform

Ein Gleichungssystem $R\vec{x} = \vec{b}$ mit $R \in \mathbb{R}^{n \times n}$ und $\vec{b}, \vec{x} \in \mathbb{R}^n$ hat die *Obere Dreiecksform*, wenn es folgendermaßen aussieht:

$$\begin{array}{ccccccc} r_{11}x_1 & + & r_{12}x_2 & + & \dots & + & r_{1n}x_n & = & b_1 \\ & & r_{22}x_2 & + & \dots & + & r_{2n}x_n & = & b_2 \\ & & & & \ddots & & \vdots & & \vdots \\ & & & & & & r_{nn}x_n & = & b_n \end{array}$$

Die Lösung erfolgt durch *Rückwärtssubstitution*, falls alle Elemente auf der Hauptdiagonalen ungleich Null sind ($r_{ii} \neq 0, i = 1, 2, \dots, n$).

$$x_i = \frac{1}{r_{ii}} \left(b_i - \sum_{k=i+1}^n r_{ik}x_k \right)$$

Ausgeschrieben ergibt die Formel folgendes Bild:

$$\begin{aligned} x_n &= \frac{1}{x_{nn}} b_n \\ x_{n-1} &= \frac{1}{r_{n-1,n-1}} (b_{n-1} - r_{n-1,n} x_n) \\ &\vdots \\ x_1 &= \frac{1}{r_{11}} (b_1 - (r_{12} x_2 + r_{13} x_3 + \dots + r_{1n} x_n)) \end{aligned}$$

Die *Determinante* der oberen Dreiecksmatrix ist das Produkt der Elemente der Hauptdiagonalen:

$$\det(R) = \prod_{i=1}^n r_{ii}$$

2.1.2 Untere Dreiecksform

Ein Gleichungssystem $L \cdot x = b$ besitzt die *Untere Dreiecksform*, wenn es wie folgt aussieht:

$$\begin{array}{rccccccc} l_{11} \cdot x_1 & & & & & & = & b_1 \\ l_{21} \cdot x_1 & + & l_{22} \cdot x_2 & & & & = & b_2 \\ \vdots & & \vdots & & \ddots & & & \vdots \\ l_{n1} \cdot x_1 & + & l_{12} \cdot x_2 & + & \dots & + & l_{nn} x_n & = & b_n \end{array}$$

Die Lösung erfolgt durch *Vorwärtssubstitution*, falls gilt: $l_{ii} \neq 0, i = 1, 2, \dots, n$

$$x_i = \frac{1}{l_{ii}} \left(b_i - \sum_{k=1}^{i-1} l_{ik} x_k \right)$$

Analog zur oberen Dreiecksmatrix entspricht die Determinante der unteren Dreiecksmatrix dem Produkt der Elemente der Hauptdiagonalen:

$$\det(L) = \prod_{i=1}^n l_{ii}$$

2.2 Das Gauß'sche Eliminationsverfahren

2.2.1 Aufgabenstellung

Gegeben sind eine Matrix $A \in \mathbb{R}^{nn}$, der Lösungsvektor $\vec{b} \in \mathbb{R}^n$. Es wird vorausgesetzt, daß die Matrix A regulär ist ($\det(A) \neq 0$ beziehungsweise $\text{Rang}(A) = n$). Gesucht wird der Lösungsvektor $\vec{x} \in \mathbb{R}^n$ gemäß $A \vec{x} = \vec{b}$.

In n Eliminationsschritten wird das Gleichungssystem in ein gestaffeltes Gleichungssystem mit oberer Dreiecksform transformiert. $A \vec{x} = \vec{b}$ wird $R \vec{x} = \vec{c}$, wobei R auf dem Platz von A und \vec{c} auf dem Platz von \vec{b} abgespeichert wird. Anschließend wird $R \vec{x} = \vec{c}$ durch Rückwärtseinsetzen gelöst.

2.2.2 Der Algorithmus

1. Im k -ten Schritt ($k = 1, 2, \dots, n-1$) wird von jeder der Zeilen i mit $i > k$ ein geeignetes Vielfaches der k -ten Zeile subtrahiert, so daß die Elemente unterhalb des Hauptdiagonalelements verschwinden. Dabei bitte nicht vergessen, die Rechte Seite mit einzubeziehen!
2. Lösen des gestaffelten Gleichungssystems durch Rückwärtseinsetzen.

Etwas formaler ausgedrückt gestaltet sich der Algorithmus wie folgt:

```
# Parameter
n      : Integer           # Anzahl der Gleichungen
A[n,n] : Real             # Koeffizientenmatrix
x[n]   : Real             # Lösungsvektor
b[n]   : Real             # Rechte Seite
# Herstellen der oberen Dreiecksmatrix
for k := 1 to n-1:       # Zeilen durchlaufen
  for i := k+1 to n:     # Spalten durchlaufen
    c := A[i,k] / A[k,k] # Eliminationskoeffizient
    for j := k+1 to n:   # Zeilenaddition durchführen
      A[i,j] -= c * A[k,j]
    A[i,k] := 0          # nach Konstruktion
    b[i] -= c * b[k]     # rechte Seite
# Rückwärtseinsetzen
for i := n downto 1:
  c := 0                 # Summe bilden
  for k := i+1 to n:
    c += A[i,k] * x[k]
  x[i] := (b[i] - c) / A[i,i] # Lösungsvektor
```

2.2.3 Dreieckszerlegung

Modifiziert man den Algorithmus so, daß man die Eliminationskoeffizienten l_{ik} anstelle von $a_{ik} = 0$ einträgt, so entsteht unterhalb der Hauptdiagonalen von A eine untere Dreiecksmatrix L , bei der lediglich die Hauptdiagonalelemente fehlen:

$$\begin{array}{ccccc}
 a_{11} & a_{12} & \dots & a_{1,n-1} & a_{1n} \\
 l_{21} & a_{22} & \dots & a_{2,n-1} & a_{2n} \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 l_{n-1,1} & l_{n-2,2} & \dots & a_{n-1,n-1} & a_{n-1,n} \\
 l_{n1} & l_{n2} & \dots & l_{n,n-1} & a_{nn}
 \end{array}$$

Satz 2.1 (Dreieckszerlegung) *Bildet man die untere Dreiecksmatrix L aus den Eliminationskoeffizienten l_{ik} mit $k = 1, \dots, n-1$ und $i = k+1, \dots, n$ des Gauß'schen Eliminationsverfahrens und setzt die Elemente der Hauptdiagonalen gleich Eins ($l_{ii} = 1, i = 1, \dots, n$), so gilt die Dreieckszerlegung $A = LR$*

Die Lösung des linearen Gleichungssystems $A \cdot \vec{x} = \vec{b}$ erfolgt in drei Schritten, wobei die Dreieckszerlegung zuerst durchgeführt wird:

1. Dreieckszerlegung von A mit einem Rechenaufwand von $\frac{1}{3}(n^3 - n)$

$$A = L \cdot R \Leftrightarrow L \cdot \underbrace{R\vec{x}}_y = \vec{b}$$

2. Berechnen eines Hilfsvektors \vec{y} durch Vorwärtseinsetzen, mit Rechenaufwand $\frac{1}{2}(n^2 + n)$

$$L\vec{y} = \vec{b}$$

3. Berechnen von x durch Vorwärtseinsetzen mit Rechenaufwand $\frac{1}{2}(n^2 + n)$

$$R\vec{x} = \vec{y}$$

Ein Vorteil gegenüber der vorhergehenden Lösung besteht nur, falls mehrere Gleichungssysteme mit derselben Koeffizientenmatrix A , aber verschiedenen rechten Seiten b vorliegen, dann sind nämlich nur noch die Schritte 2 und 3 erforderlich.

2.2.4 Pivotelementsuche

Diagonalstrategie

Satz 2.2 (Diagonaldominanz) Die Eigenschaft Diagonaldominant überträgt sich beim Gauß'schen Eliminationsverfahren auf die reduzierte Matrix.

Spaltenmaximumstrategie

1. Bestimme p so, daß gilt:

$$|a_{pk}| = \max |a_{ik}|$$

(= betragsgrößtes Element der k -ten Spalte der Restmatrix)

2. falls $p \neq k$:

- (a) vertausche Zeile p mit Zeile k (inklusive rechte Seite)
- (b) $\det(A) := -\det(A)$

Programmierungstechnisch werden die Zeilenvertauschungen nicht explizit durchgeführt, sondern in einem Merkvektor \vec{u} festgehalten, der mit den Werten $\vec{u} = (1, 2, 3, 4, \dots, n)^T$ vorbesetzt ist. Die Vertauschung der Zeilen 2 und 4 würde sich wie folgt niederschlagen: $\vec{u} = (1, 4, 3, 2, \dots, n)^T$.

Relatives Spaltenmaximum

1. Bestimme Index p so, daß gilt:

$$\frac{|a_{pk}|}{\sum_{j=k}^n |a_{pj}|} = \max_{i \geq k} \frac{a_{ik}}{\sum_{j=k}^n |a_{ij}|}$$

Das Pivotelement ist das Element, das dem Betrage nach zur Summe der Beträge der zugehörigen Zeile am größten ist.

2. falls $p \neq k$: wie zuvor

Gesamtpivotsuche

1. Pivot = betragsgrößtes Element der Restmatrix
bestimme Indizes p, q so, daß gilt:

$$|a_{pq}| = \max_{i,j \geq k} |a_{ij}|$$

2. falls $p \neq k$:
 - (a) vertausche Zeile p mit Zeile k , inklusive rechte Seite
 - (b) $\det(A) = -\det(A)$
3. falls $p \neq q$:
 - (a) vertausche Spalte q mit Spalte k
 - (b) vertausche x_q mit x_k
 - (c) $\det(A) = -\det(A)$

2.2.5 Gleichungssysteme mit mehreren Rechten Seiten

Wie wir weiter vorne bereits kennengelernt haben, kann man beim Lösen von Gleichungssystemen, bei denen nur die Rechte Seite \vec{b} variiert die Koeffizientenmatrix A beibehalten und so einen Menge Rechenaufwand vermeiden. Faßt man die Lösungsvektoren x_1, x_2, \dots, x_m zu den Rechten Seiten b_1, b_2, \dots, b_m sowie die Rechten Seiten selbst als Matrix auf, so kann man schreiben:

$$\begin{aligned} & A \cdot x_1 = b_1, Ax_2 = b_2, \dots, Ax_m = b_m \\ \Leftrightarrow & A \cdot \underbrace{(x_1, x_2, \dots, x_m)}_X = \underbrace{(b_1, b_2, \dots, b_m)}_B \\ \Leftrightarrow & A \cdot X = B \end{aligned}$$

Nimmt man anstelle einer beliebigen Matrix B die Einheitsmatrix E , so ergibt die Lösung X des Gleichungssystems $AX = E$ die *inverse Matrix* von A . In der Praxis existieren jedoch leistungsfähigere Verfahren zur *Matrixinversion*. Prinzipiell werden Invertierungen aufgrund numerischer Instabilitäten so weit wie möglich vermieden.

2.2.6 Nachiteration

Beim numerischen Lösen von Gleichungssystemen schleichen sich unweigerlich Rechenungenauigkeiten ein. Die *Nachiteration* dient zur Verbesserung dieser Lösung. Die im Gauss-Verfahren ermittelte Näherungslösung nennen wir nun x_N . Der Korrektuversuch beginnt mit dem folgenden Ansatz:

$$\vec{x}_E = \vec{x}_N + \Delta x$$

Setzt man die Näherungslösung x_N in das Gleichungssystem ein und vergleicht das Ergebnis mit der Rechten Seite, erhält man den einen Fehlervektor r , den sogenannten *Residuenvektor*.

$$\vec{r} = \vec{b} - A\vec{x}_N$$

Dann gilt nach Aufgabenstellung:

$$A\vec{x}_E = A(\vec{x}_N + \Delta\vec{x}) = A\vec{x}_N + A\Delta\vec{x} = \vec{b}$$

Falls der Residuenvektor $\vec{r} \neq 0$ (im Rahmen der Rechengenauigkeit) gewinnt man die Korrektur $\Delta\vec{x}$ aus dem linearen Gleichungssystem $A\Delta\vec{x} = \vec{r}$.

Zur Berechnung von $\Delta\vec{x}$ verwendet man dieselbe Koeffizientenmatrix A . Dann ergibt sich:

$$\vec{x}_E := \vec{x}_N + \Delta\vec{x}$$

Eventuell sind weitere Korrekturen möglich.

Den Hauptrechenaufwand bereitet mit $O(n^3)$ immer noch die Dreieckszerlegung der Koeffizientenmatrix, die Nachiteration fällt mit $O(n^2)$ kaum ins Gewicht.

Nachiteration als Algorithmus

Sei \vec{x} die Näherungslösung von $A\vec{x} = \vec{b}$.

1. Berechne $\vec{r} = \vec{b} - A\vec{x}$
2. Falls $|\vec{r}| \leq \frac{n}{|\vec{b}|} \cdot \varepsilon_{\text{Masch.}}$, Beende Algorithmus
3. Löse $A\Delta\vec{x} = \vec{r}$
4. Falls die Korrektur brauchbar ist, setze $\vec{x} = \vec{x} + \Delta\vec{x}$

2.3 Vektor- und Matrixnormen

Vektor- und Matrixnormen dienen als Hilfsmittel beim Feststellen sowohl der numerischen Fehler beim Lösen von linearen Gleichungssystemen als auch der numerischen Stabilität (Änderung der Ergebnisse bei geringfügiger Änderung der Eingangsdaten).

Definition 2.1 (Vektornorm) *Unter einer Vektornorm versteht man ein reelles Maß $\|\vec{x}\| \in \mathbb{R}$ für die Vektorelemente eines Vektors $\vec{x} \in \mathbb{R}^n$. Folgende Eigenschaften müssen erfüllt sein:*

- $\|\vec{x}\| > 0$ für alle $\vec{x} \neq 0$ und $\|\vec{x}\| = 0$ für alle $\vec{x} = 0$
- $\|c\vec{x}\| = |c| \cdot \|\vec{x}\|$ für alle $c \in \mathbb{R}$, $\vec{x} \in \mathbb{R}^n$
- $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$ für alle $\vec{x}, \vec{y} \in \mathbb{R}^n$

Diese drei Eigenschaften sind leicht verifizierbar. Einige Beispiele für gebräuchliche Vektornormen seien hier genannt:

- Euklid-Norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_{k=1}^n x_k^2} = |\vec{x}|$$

	$\ A\ _G$	$\ A\ _Z$	$\ A\ _S$	$\ A\ _F$
$\ x\ _\infty$	ja	ja	nein	nein
$\ x\ _1$	ja	nein	ja	nein
$\ x\ _2$	ja	nein	nein	ja

- L_1 -Norm:
 $\|\vec{x}\|_1 = \sum_{k=1}^n |x_k|$
- Maximum-Norm:
 $\|\vec{x}\|_\infty = \max_{k=1}^n |x_k|$

Definition 2.2 (Matrixnorm) *Unter einer Matrixnorm versteht man ein reelles Maß $\|A\| \in \mathbb{R}$ für die Elemente einer Matrix $A \in \mathbb{R}^{n \times n}$, das folgende Eigenschaften erfüllt:*

- $\|A\| > 0 \forall A; \|A\| = 0$ nur für $A =$ Nullmatrix
- $\|c \cdot A\| = |c| \cdot \|A\| \forall c \in \mathbb{R}^1$
- $\|A + B\| \leq \|A\| + \|B\|$
- $\|A \cdot B\| \leq \|A\| \cdot \|B\|$

Beispiele für Matrixnormen:

- Gesamtnorm (betragsgrößtes Element):
 $\|A\|_G = n \cdot \max_{i,d} a_{ik}$
- Zeilensummennorm (betragsgrößte Zeile):
 $\|A\|_Z = \|A\|_\infty = \max \sum_k |a_{ik}|$
- Spaltensummennorm (betragsgrößte Spalte):
 $\|A\|_S = \|A\|_1 = \max_k \sum_{i=1}^n a_{ik}$
- Frobeniusnorm:
 $\|A\|_F = \|A\|_2 = \sqrt{\sum_{i=1}^n \sum_{k=1}^n a_{ik}^2}$

2.4 Verträglichkeitsbedingungen

Welche Vektornorm darf mit welcher Matrixnorm kombiniert werden?

Definition 2.3 (Verträglichkeit) $\|A\|$ heißt verträglich mit $\|x\|$, falls gilt:

$$\|A \cdot x\| \leq \|A\| \cdot \|x\| \forall x \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$$

Verträglichkeitstabelle der bisher genannten Vektor- und Matrixnormen:

2.4.1 Fehlerabschätzung, "Kondition" einer Matrix

Relativer Fehler der erhaltenen Lösung

Wir betrachten ein Gleichungssystem $Ax = b$. Zur Berechnung der exakten Lösung benötigt man die Inverse von A : $x_E = A^{-1}b$.

Numerische Lösung: x_N

$$r := b - Ax_N$$

Dann ist:

$$\begin{aligned} x_E &= x_N + \Delta x \\ \Rightarrow A \cdot x_E &= A \cdot x_N + A \cdot \Delta x = b \\ \Rightarrow b - A \cdot x_N &= A \cdot \Delta x \\ b - A \cdot x_N &= r \end{aligned}$$

"absoluter Fehler":

$$\Delta x = A^{-1} \cdot r$$

relativer Fehler:

$$\frac{\|\Delta x\|}{\|x_E\|} = \frac{\|A^{-1} \cdot r\|}{\|A^{-1} \cdot b\|} = \frac{\|A^{-1}\| \cdot \|r\|}{\|A^{-1} \cdot b\|} = \frac{\|A\| \cdot \|A^{-1}\| \cdot \|b\|}{\|A\| \cdot \|A^{-1} \cdot b\|} \Leftarrow \frac{\|A\| \cdot \|A^{-1}\| \cdot \|b\|}{\| \underbrace{A \cdot A^{-1}}_{\text{Einheitsmatrix}} \cdot b\|} = \frac{\|A\| \cdot \|A^{-1}\|}{\|b\|} \cdot \|r\|$$

Ergebnis: Relativer Fehler bei der Lösung von $A \cdot x = b$:

$$\frac{\|\Delta x\|}{\|x_E\|} \Leftarrow \underbrace{\|A\| \cdot \|A^{-1}\|}_{\chi(A)} \cdot \frac{\|r\|}{\|b\|}$$

mit

$$r := b - A \cdot x_N$$

$\chi(A) := \|A\| \cdot \|A^{-1}\|$ "Kondition" von A

2.4.2 Einfluss geringfügig geänderter Eingangsdaten

Zu lösen:

$$A \cdot x = b$$

geringfügige Änderung der Eingangsdaten:

$A + \Delta A$ statt A

$b + \Delta b$ statt b

führt auf

$x + \Delta x$ statt x

längere Herleitung liefert:

$$\frac{\|\Delta x\|}{\|x\|} \Leftarrow \frac{\chi(A)}{1 - \chi(A) \cdot \frac{\|\Delta A\|}{\|A\|}} \cdot \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right)$$

mit

$$\chi(A) = \|A\| \cdot \|A^{-1}\|$$

enthält auch $\Delta A = \text{Nullmatrix}$ oder $\Delta b = 0$

Abschätzung: Sei $1 - \chi(A) \cdot \frac{\|\Delta A\|}{\|A\|} \approx 1$

$$\chi(A) = 10^\alpha; \underbrace{\frac{\|\Delta A\|}{\|A\|} = ca5 \cdot 10^{-d}; \frac{\|\Delta b\|}{\|b\|} = ca5 \cdot 10^{-d}}_{\text{relativer Fehler der Eingangsdaten}}$$

dann:

$$\frac{\|\Delta x\|}{\|x\|} \leq 10^\alpha \cdot (5 \cdot 10^{-d} + 5 \cdot 10^{-d}) = 10^{-d+\alpha+1}$$

Interpretation: $\alpha + 1$ Dezimalstellen gehen verloren

2.4.3 Anwendung auf Pivotstrategie

Eine Matrix ist im Rahmen der Maschinengenauigkeit singular, wenn das Pivotelement gleich Null ist:

$$\frac{|a_{kk}|}{\|A\|} \leq \varepsilon \Rightarrow \text{Matrix singular}$$

2.5 Verfahren von Cholesky

Das Verfahren dient als Alternative zum Gauß'schen Eliminationsverfahren zur effizienten Dreieckszerlegung von symmetrischen, positiv definiten Matrizen.

2.5.1 Vorbereitungen

Definition 2.4 (symmetrisch) Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt symmetrisch, wenn gilt:

$$A = A^T$$

Definition 2.5 (positiv definit) Eine Matrix $A \in \mathbb{R}^{n \times n}$ ist positiv definit, wenn gilt:

$$x^T \cdot A \cdot x > 0 \forall x \in \mathbb{R}^n, x \neq 0$$

Definition 2.6 (diagonaldominant) Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt diagonaldominant, wenn der Absolutwert des Diagonalelementes jeder Zeile größer ist als die Summe der Beträge der restlichen Elemente in dieser Zeile:

$$|a_{ik}| > \sum_{k=1, k \neq i}^n |a_{ik}|, i = 1, 2, \dots, n$$

Folgende Matrizen sind nach Konstruktion positiv definit:

- $A = C^T \cdot C$ mit $A \in \mathbb{R}^{n \times n}$ und $C \in \mathbb{R}^{n \times m}$
- $A = L \cdot R$ mit $l_{ii} = 1$ und $r_{ii} > 0$ für alle $i = 1, 2, \dots, n$
- Diagonaldominante Matrizen

2.5.2 Herleitung des Verfahrens

Man geht davon aus, daß L bekannt ist und berechnet formal $A = L \cdot L^T$

$$\begin{array}{c|c}
 & L^T = \begin{pmatrix} l_{11} & l_{21} & l_{31} & \dots & l_{n1} \\ & l_{22} & l_{32} & & l_{n2} \\ & & l_{33} & & l_{n3} \\ & & & \ddots & \vdots \\ & & & & l_{nn} \end{pmatrix} \\
 \hline
 L = \begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix} & A = \begin{pmatrix} a_{11} & & & & \star \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ \vdots & & & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}
 \end{array}$$

Erste Spalte:

$$\begin{aligned}
 a_{11} = l_{11}^2 &\Rightarrow l_{11} = \sqrt{a_{11}} \\
 a_{i2} = l_{11}l_{i2} &\Rightarrow l_{i2} = \frac{a_{i2}}{l_{11}}
 \end{aligned}$$

Zweite Spalte:

$$\begin{aligned}
 a_{22} = l_{21}^2 + l_{22}^2 &\Rightarrow l_{22} = \sqrt{a_{22} - l_{21}^2} \\
 a_{i2} = l_{21}l_{i1} + l_{22}l_{i2} &\Rightarrow l_{i2} = \frac{a_{i2} - l_{21}l_{i1}}{l_{22}}
 \end{aligned}$$

Die Hauptdiagonalelemente berechnen sich dann allgemein wie folgt:

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \quad i = 1, 2, \dots, n$$

Und die Elemente der unteren Dreiecksmatrix folgendermaßen:

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk} \right), \quad i = 1, 2, \dots, n \quad \text{und} \quad j = 1, 2, \dots, i-1$$

2.5.3 Der Algorithmus

```

# Parameter
n      : Integer                # Anzahl der Zeilen und Spalten
a[n,n] : Real                  # Symmetrische, positiv definite
                                      # Koeffizientenmatrix

# Cholesky Verfahren
for i := 1 to n:
    sum := 0                    # Diagonalelemente berechnen
    for k := 1 to n-1:

```

```

    sum += l[i,k] * l[i,k]
    l[i,i] := sqrt(a[i,i] - sum)

    for j := 1 to i-1:           # Untere Dreiecksmatrix berechnen
        sum := 0
        for k := 1 to j-1:
            sum += l[i,k] * l[j,k]
        l[i,j] := (a[i,j] - sum) / l[j,j]

```

Der Rechenaufwand beträgt lediglich $\frac{1}{6}(n^3 + 3n^2 - 4n) + n$. Das ist beinahe doppelt so gut wie bei der Gauß-Elimination, zusätzlich ist keine Pivotstrategie nötig.

Ein guter Informatiker kann außerdem die Platzkomplexität reduzieren, indem er lediglich die Hauptdiagonalelemente der Unteren Dreiecksmatrix außerhalb speichert und den Rest der Unteren Dreiecksmatrix innerhalb der Koeffizientenmatrix unterbringt. Dies ist aufgrund der Symmetrieeigenschaft möglich.

2.5.4 Vorgehensweise

Das Lösen eines linearen Gleichungssystems $Ax = b$ mit $A = A^T$ und A positiv definit erfolgt in drei Schritten:

1. Berechnen der Unteren Dreiecksmatrix durch das Verfahren von Cholesky:

$$L \cdot \underbrace{L^T \cdot x}_v = b$$

2. Lösen der Gleichung $Lv = b$ durch Vorwärtseinsetzen
3. Berechnen von x durch Rückwärtseinsetzen aus $L^T x = v$

2.6 Iterationsverfahren

Man kann die k -te Gleichung eines beliebigen Gleichungssystem $Ax = b$ mit $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ folgendermaßen darstellen:

$$a_{k1}x_1 + a_{k2}x_2 + \dots + a_{kk}x_k + \dots + a_{kn}x_n = b_k$$

Damit dieser Verfahren funktionieren, muß die zugrundeliegende Matrix diagonaldominant sein.

2.6.1 Gesamtschrittverfahren nach Jacobi

Man beginnt mit einem beliebig gewählten *Startvektor* $x^{(0)}$, setzt diese zufälligen Werte in die nach dem jeweiligen x_k aufgelöste Gleichung ein und erhält so einen "besseren" Startvektor:

$$x_i^{\text{neu}} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{\text{alt}} - \sum_{j=i+1}^n a_{ij}x_j^{\text{alt}} \right), \quad i = 0, 1, \dots, n$$

Das Verfahren wird solange durchlaufen, bis ein *Abbruchkriterium* erfüllt wird, zum Beispiel, wenn der Residuenvektor ($\vec{r} = \vec{b} - A\vec{x}$) im Rahmen der Maschinengenauigkeit Null beträgt: $\|\vec{r}\| \leq \varepsilon_M$. Alternativ kann auch die Differenz zwischen zwei aufeinanderfolgenden Lösungsvektoren als Kriterium verwendet werden, dann wird abgebrochen, falls gilt:

$$\frac{\|\vec{x}_{\text{neu}}\| - \|\vec{x}_{\text{alt}}\|}{\|\vec{x}_{\text{neu}}\|} \leq \varepsilon_M \quad \text{oder besser:} \quad \|\vec{x}_{\text{neu}}\| - \|\vec{x}_{\text{alt}}\| \leq \|\vec{x}_{\text{neu}}\| \cdot \varepsilon_M$$

Der Rechenaufwand beträgt pro Iterationsschritt $2n$ Operationen. Falls die Koeffizientenmatrix nur schwach besetzt ist (die meisten Elemente sind Null), werden auch nur die ursprünglichen Elemente verwendet, die Null-Elemente werden im Gegensatz zum Gauß-Jordan Verfahren *nicht* überschrieben.

Beispiel 2.1 (SS2001/1. Aufgabe) Gegeben ist das lineare Gleichungssystem $A\vec{x} = \vec{b}$ mit

$$\begin{pmatrix} 8 & 0 & 2 \\ 0 & 14 & 1 \\ 2 & 1 & 11 \end{pmatrix} \cdot \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad ; \quad \text{beginne bei} \quad \vec{x}_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\vec{x}_1 = \begin{pmatrix} \frac{1}{8} & (1 & -(0 \cdot 1 + 2 \cdot 1)) \\ \frac{1}{8} & (2 & -(0 \cdot 1) & -(1 \cdot 1)) \\ \frac{1}{8} & (3 & -(2 \cdot 1 + 1 \cdot 1)) &) \end{pmatrix} = \begin{pmatrix} \frac{-1}{8} \\ \frac{1}{14} \\ 0 \end{pmatrix}$$

2.6.2 Einzelschrittverfahren nach Seidel

Analog zum Gesamtschrittverfahren wählt man einen beliebigen Startvektor $\vec{x}^{(0)}$, setzt jedoch die neu berechneten, besseren Werte so früh wie möglich ein:

$$x_i^{\text{neu}} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{\text{neu}} - \sum_{j=i+1}^n a_{ij} x_j^{\text{alt}} \right), \quad i = 0, 1, \dots, n$$

Programmierungstechnisch sind, je nach Abbruchkriterium, höchstens zwei aufeinanderfolgende iterierte Vektoren x^{alt} und x^{neu} erforderlich. Die Konvergenzbedingungen sind günstiger als im Gesamtschrittverfahren.

Kapitel 3

Interpolation

Die Aufgabenstellung der Interpolation ist das Finden von Zwischenwerten zu einer Menge an gegebenen Punkten.

Definition 3.1 (Interpolation) *Bei der Interpolation mit Polynomen geht die errechnete Funktion durch die Gegebenen Punkte hindurch. Man nennt dies auch die Interpolationsbedingung.*

Definition 3.2 (Approximation) *Bei der Approximation nähert sich die errechnete Funktion möglichst gut an die gegebenen Punkte an.*

Zur Interpolation und Approximation periodischer Funktionen eignen sich Trigonometrische Funktionen (Sinus und Cosinus), das Verfahren heißt *Fourier-Analyse*.

3.1 Lagrange-Interpolation

3.1.1 Aufgabenstellung

Gegeben sind $n + 1$ Stützstellen $(x_0, y_0), \dots, (x_n, y_n)$ mit $x_i, y_i \in \mathbb{R}$, Gesucht wird das Interpolationspolynom $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$, oder genauer gesagt die Koeffizienten des Polynoms n -ten Grades $P_n(x)$, so daß gilt:

$$P_n(x_i) = y_i, i = 0, 1, \dots, n$$

Satz 3.1 (Fundamentalsatz der Algebra) *Ein Polynom vom Grad n hat genau n Nullstellen.*

3.1.2 Eindeutigkeit

Satz 3.2 (Eindeutigkeit) *Es gibt nur ein Polynom vom Grad n , daß die oben genannte Aufgabenstellung erfüllt.*

Beweis 3.1 (Beweis durch Widerspruch) *Angenommen, es gibt zwei Polynome $P_n(x)$ und $P_n^*(x)$, die beide die Aufgabe lösen. Man bildet das Polynom $Q_n(x) = P_n(x) - P_n^*(x)$. Dann gilt:*

$$Q_n(x_i) = P_n(x_i) - P_n^*(x_i) = y_i - y_i^* = 0, i = 0, 1, \dots, n$$

Im Klartext: das Polynom $Q_n(x)$ hätte $n + 1$ Nullstellen, was dem Fundamentalsatz der Algebra widerspricht.

3.1.3 Existenz

Die Aufgabe, ein Polynom vom Grad n durch $n + 1$ Punkte zu legen wird gelöst durch:

$$P_n(x) = \sum_{i=0}^n y_i \cdot L_i(x) \quad \text{mit}$$

$$L_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \frac{x - x_1}{x_i - x_1} \dots \frac{x - x_{j-1}}{x_i - x_{j-1}} \frac{x - x_{j+1}}{x_i - x_{j+1}} \dots \frac{x - x_n}{x_i - x_n}$$

Die Lagrange-Polynome $L_i(x)$ sind alle vom Grad n und es gilt:

$$L_i(x_k) = \begin{cases} 0 & \text{für } i \neq k \\ 1 & \text{für } i = k \end{cases} = \delta_{i,k}$$

Dann ist an der Stelle k :

$$P_n(x) = \sum_{i=0}^n y_i \cdot L_i(x_k) = \sum_{i=0}^n y_i \cdot \delta_{i,k} = y_k$$

Beispiel 3.1 ($n = 1$: Lineare Interpolation) Gegeben sind die Stützpunkte (x_0, y_0) mit $x_0 = 1, y_0 = 2.7183$ und (x_1, y_1) mit $x_1 = 2, y_1 = 7.3891$ aus der e^x -Funktion, gesucht ist der Wert an der Stelle $x = 1.5$. (der exakte Wert wäre: 4.4817)

$$\begin{aligned} P_1(x) &= y_0 L_0(x) + y_1 L_1(x) \\ &= y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \\ &= 2.7183 \frac{x - 2}{1 - 2} + 7.3891 \frac{x - 1}{2 - 1} \\ P_1(1.5) &= 5.0537 \end{aligned}$$

Beispiel 3.2 ($n = 2$: Quadratische Interpolation) In diesem Beispiel gilt derselbe Versuchsaufbau wie oben, jedoch mit einer zusätzlichen Stützstelle (x_0, y_0) mit $x_0 = 0, y_0 = 1$.

$$\begin{aligned} P_2(x) &= y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) \\ &= y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \\ &= 1 \frac{(x - 1)(x - 2)}{(0 - 1)(0 - 2)} + 2.7183 \frac{(x - 0)(x - 2)}{(1 - 0)(1 - 2)} + 7.3891 \frac{(x - 0)(x - 1)}{(2 - 0)(2 - 1)} \\ P_2(1.5) &= 4.6846 \end{aligned}$$

3.1.4 Äquidistante Stützstellen

Sei

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

dann ist

$$x_i = x_0 + i \cdot h; \quad i = 0, 1, \dots, n$$

Wir setzen

$$x = x_0 + t \cdot h; \quad t \in [0, n]$$

dann:

$$\begin{aligned} x - x_j &= x_0 + t \cdot h - x_0 - j \cdot h = h \cdot (t - j) \\ x_i - x_j &= h \cdot (i - j) \end{aligned}$$

damit

$$L_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j} = \prod_{j=0, j \neq i}^n \frac{h \cdot (t - j)}{h \cdot (i - j)}$$

Die Interpolationsformel für *äquidistante Stützstellen* lautet:

$$P_n(x) = \sum_{i=0}^n y_i L_i(x) = \sum_{i=0}^n y_i \prod_{j=0, j \neq i}^n \frac{t - j}{i - j}$$

3.2 Newton-Interpolation

Im Ansatz wird bereits die Hinzunahme weiterer Stützstellen berücksichtigt.

3.2.1 Aufgabenstellung

Gegeben sind $n+1$ paarweise voneinander verschiedene Stützstellen $(x_0, y_0), \dots, (x_n, y_n)$.
Gesucht werden die Koeffizienten b_0, b_1, \dots, b_n im Ansatz:

$$\begin{aligned} P_n(x) &= b_0 \\ &+ b_1(x - x_0) \\ &+ b_2(x - x_0)(x - x_1) \\ &\vdots \\ &+ b_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) \end{aligned}$$

Berechnung der Koeffizienten b_0, b_1, \dots, b_n mit Hilfe der Interpolationsbedingung:

$$\begin{aligned} i = 0 : P_n(x_0) = y_0 = b_0 &\Rightarrow b_0 = y_0 \\ i = 1 : P_n(x_1) = y_1 = b_0 + b_1(x - x_0) &\Rightarrow b_1 = \frac{y_1 - y_0}{x_1 - x_0} \\ i = 2 : P_n(x_2) = y_2 = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) &\Rightarrow b_2 = \frac{y_2 - b_0 - b_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \end{aligned}$$

Durch Umformung der letzten Gleichung erhält man:

$$b_2 = \frac{\frac{y_2 - y_1}{x_2 - x_1} - \overbrace{\frac{y_1 - y_0}{x_1 - x_0}}^{b_1}}{x_2 - x_0}$$

Wie man sieht (und durch aufwendige Rechnung auch für weitere Stützstellen zeigen kann), tauchen hier immer wieder bereits berechnete Brüche auf. Man führt eine neue Schreibweise ein:

$$\begin{aligned} b_0 &= f[x_0] = y_0 \\ b_1 &= f[x_1x_0] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\ b_2 &= f[x_2x_1x_0] = \frac{f[x_2x_1] - f[x_1x_0]}{x_2 - x_0} \end{aligned}$$

3.2.2 Bildungsgesetz der dividierten Differenzen

Das Bildungsgesetz für die *dividierten Differenzen* lautet

$$f[x_{i+k}x_{i+k-1} \dots x_{i+1}x_i] := \frac{f[x_{i+k} \dots x_{i+1}] - f[x_{i+k-1} \dots x_i]}{x_{i+k} - x_i}$$

mit den Startwerten $f[x_i] = y_i, i = 0, 1, \dots, n$. Damit es noch klarer wird, folgen hier noch ein paar Werte:

$$\begin{aligned} f[x_1x_0] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\ f[x_2x_1] &= \frac{f[x_2] - f[x_1]}{x_2 - x_1} \\ f[x_2x_1x_0] &= \frac{f[x_2x_1] - f[x_1x_0]}{x_2 - x_0} \\ &\vdots \end{aligned}$$

3.2.3 Rechenschema

Bei der Berechnung der Koeffizienten b_0, b_1, \dots, b_n hilft das folgende Rechenschema. Die Koeffizienten selbst lassen sich aus der obersten Schrägzeile ablesen.

x_i	b_0	b_1	b_2	b_3	b_4
x_0	$y_0 = f[x_0]$				
x_1	$y_1 = f[x_1]$	$f[x_1x_0]$			
x_2	$y_2 = f[x_2]$	$f[x_2x_1]$	$f[x_2x_1x_0]$		
x_3	$y_3 = f[x_3]$	$f[x_3x_2]$	$f[x_3x_2x_1]$	$f[x_3x_2x_1x_0]$	
x_4	$y_4 = f[x_4]$	$f[x_4x_3]$	$f[x_4x_3x_2]$	$f[x_4x_3x_2x_1]$	$f[x_4x_3x_2x_1x_0]$

Die Koeffizienten b_i sind dann die Werte der obersten Schrägzeile. ($f[\dots x_0]$)
Durch Hinzunahme eines weiteren Punktes entsteht eine neue Schrägzeile und eine neue Spalte mit dem nächsten Koeffizienten.

3.2.4 Algorithmus

Zum Abschluß der Newton-Interpolation folgt der prgrammtechnisch leicht umsetzbare Algorithmus zur Berechnung der Polynomkoeffizienten b_i :

```
for i = 0 to n:
  b[i] = y[i]
for k = 0 to n:
  for i = n downto k
    b[i] = (b[i] - b[i-1]) / (x[i] - x[i-k])
```

3.2.5 Berechnung eines Zwischenwertes

Die Berechnung des Polynomwertes $P_n(x)$ an der Stelle x erfolgt analog zum *HORNER-Schema*:

$$\begin{aligned}
 P_n(x) &= b_0 \\
 &+ b_1(x - x_0) \\
 &+ b_2(x - x_0)(x - x_1) \\
 &+ b_3(x - x_0)(x - x_1)(x - x_3) \\
 &+ \dots \\
 &+ b_n(x - x_0)(x - x_1) \cdot \dots \cdot (x - x_{n-1}) \\
 P_n(x) &= b_0 + (x - x_0)(b_1 + (x - x_1) \cdot (b_2 + (x - x_2)
 \end{aligned}$$

Hiermit ergibt sich folgender Algorithmus:

```
P = b[n];
for k = n-1 downto 0:
  P = P * (x - x[k]) + b[k]
```

3.2.6 Sonderfall: Äquidistante Stützstellen

FIXME

3.3 Interpolation mit rationalen Funktionen

Die Interpolation mit rationalen Funktionen eignet sich vor allem für Funktionen mit *Singularitäten* und/oder Funktionen, deren Graph eine horizontale beziehungsweise eine *schiefe Asymptote* aufweist.

3.3.1 Aufgabenstellung

Gegeben sind $n + 1$ Stützstellen $(x_0, y_0), \dots, (x_n, y_n)$ einer rationalen Funktion $R(x)$ mit

$$R(x) = \frac{p_0 + p_1x + \dots + p_sx^s}{q_0 + q_1x + \dots + q_tx^t}, \quad p, q \in \mathbb{R}$$

Gesucht werden die Koeffizienten $p_0, p_1, \dots, p_s, q_0, q_1, \dots, q_t$.

3.3.2 Herleitung

Im Polynom $R(x)$ kann durch einen der Koeffizienten gekürzt werden, so daß die Bedingung, daß das Polynom maximal so viele Koeffizienten enthalten darf, wie Stützstellen gegeben sind auch erfüllt bleibt, wenn ein zusätzlicher Koeffizient hinzugenommen wird.

Benutzt man die Interpolationsbedingung zur Berechnung der Koeffizienten, ergibt sich folgendes Bild:

$$R(x_i) = y_i = \frac{p_0 + p_1x_i + \dots + p_sx_i^s}{q_0 + q_1x_i + \dots + q_tx_i^t}, \quad i = 0, 1, \dots, n$$

$$\Leftrightarrow p_0 + p_1x_i + \dots + p_sx_i^s = y_i(q_0 + q_1x_i + \dots + q_tx_i^t), \quad i = 0, 1, \dots, n$$

Ausgeschrieben ist dies ein homogenes, lineares Gleichungssystem:

$$\begin{aligned} p_0 + (x_0)p_1 + \dots + (x_0^s)p_s &= y_0q_0 + (y_0x_0)q_1 + \dots + (y_0x_0^t)q_t \\ p_0 + (x_1)p_1 + \dots + (x_1^s)p_s &= y_1q_0 + (y_1x_1)q_1 + \dots + (y_1x_1^t)q_t \\ &\vdots \\ p_0 + (x_n)p_1 + \dots + (x_n^s)p_s &= y_nq_0 + (y_nx_n)q_1 + \dots + (y_nx_n^t)q_t \end{aligned}$$

In diesem Gleichungssystem kann einer der Koeffizienten beliebig gewählt werden, sinnvollerweise setzt man ihn gleich Eins. Die gesuchten Koeffizienten erhält man nun durch Lösen dieses Gleichungssystems.

3.4 Spline-Interpolation

Die herkömmliche Polynominterpolation bei vielen Stützstellen und damit einem hohen Polynomgrad hat den Nachteil, daß die "Ausschläge" zwischen den Stützstellen mit der Größe des Polynomgrades unverhältnismäßig wachsen.

Die *Spline-Interpolation* stammt ursprünglich aus dem Schiffsbau (sie ist auch als "Latteninterpolation" bekannt. Es wird für *jedes* Stützstellenintervall ein Polynom geringen Grades gelegt.

3.4.1 Kubische Splines

Man legt zwischen je zwei Stützstellen $(x_i, y_i), (x_{i+1}, y_{i+1})$ ein Polynom vom Grad drei. Es wird gefordert, daß an den Stützstellen ein stetiger Übergang in der Funktion sowie in der ersten und zweiten Ableitung besteht. Zusätzlich sind die Werte der Ableitung an den Randpunkten gegeben. Für jedes Intervall wird also eine Splinefunktion $\varphi(x)$ gesucht, das folgende Eigenschaften erfüllt:

1. $\varphi(x)$, $\varphi'(x)$ und $\varphi''(x)$ sind stetig im Intervall $[x_0, x_n]$, dies gilt auch an den Stützstellen x_i selbst!
2. $\varphi(x)$ ist in jedem Intervall $[x_i, x_{i+1}]$ durch ein Polynom dritten Grades gegeben:

$$\varphi(x) = P_i(x) := a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 0, 1, \dots, n-1$$

3. Die Interpolationsbedingung ist erfüllt:

$$\varphi(x_i) = y_i, i = 0, 1, \dots, n$$

4. $\varphi(x)$ erfüllt die beiden Randbedingungen:

$$\varphi'(x_0) = \alpha \quad \text{und} \quad \varphi'(x_n) = \beta$$

5. Der Bequemlichkeit halber legt man ein Polynom vom Grad zwei durch die Randpunkte:

$$\begin{aligned} P_0(x) &= a_0 + b_0(x - x_0) + c_0(x - x_0)^2 \\ P_n(x) &= a_n + b_n(x - x_n) + c_n(x - x_n)^2 \end{aligned}$$

Aus diesen Forderungen mit $4n + 3$ Unbekannten lassen sich nun $4n + 3$ Bestimmungsgleichungen zur Berechnung der Unbekannten ableiten:

$$\begin{aligned} (A) \quad P_i(x_i) &= y_i; & i = 0, 1, \dots, n & \quad (= \text{Bedingung 3}) \\ (B) \quad P_i(x_{i+1}) &= y_{i+1}; & i = 0, 1, \dots, n-1 & \\ (C) \quad P'_i(x_i) &= P'_{i-1}(x_i); & i = 1, 2, \dots, n & \\ (D) \quad P''_i(x_i) &= P''_{i-1}(x_i); & i = 1, 2, \dots, n & \\ (E) \quad P'_0(x_0) &= \alpha; & & \\ & P'_n(x_n) &= \beta; & \\ & P'_{n-1}(x_n) &= \beta & \end{aligned}$$

Aus (A) und 2. erhält man: $a_i = y_i, \quad i = 0, 1, \dots, n$

Mittels sehr umfangreicher, aber elementarer Rechnung lassen sich nun b_i und d_i durch c_i ausdrücken, und dann mit:

$$(h_i := x_i - x_{i+1}, \quad i = 0, \dots, n-2)$$

$$\begin{aligned} 2 \cdot h_0 \cdot c_0 + h_0 \cdot c_1 &= 3 \cdot \left(\frac{y_1 - y_0}{h_0} - \alpha \right) \\ h_i + c_i + 2 \cdot (h_i + h_{i+1}) \cdot c_{i+1} + h_{i+1} \cdot c_{i+2} &= 3 \cdot \left(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i} \right) \\ h_{n-1} \cdot c_{n-1} + 2 \cdot h_{n-1} \cdot c_n &= 3 \cdot \left(\beta - \frac{y_n - y_{n-1}}{h_{n-1}} \right) \end{aligned}$$

Aus diesen $n + 1$ Gleichungen für c_0, c_1, \dots, c_n läßt sich ein lineares Gleichungssystem aufstellen (Koeffizientenmatrix), aus dem man die Koeffizienten c_i errechnen kann. Anschließend erhält man b_i und d_i aus folgenden Formeln:

$$\left. \begin{aligned} b_i &= \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i) \\ d_i &= \frac{c_{i+1} - c_i}{3h_i} \end{aligned} \right\} \quad \text{mit} \quad i = 0, 1, \dots, n-1$$

Damit sind in jedem Intervall $[x_i, x_{i+1}]$ die Koeffizienten des gesuchten Polynoms bekannt:

$$P_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Spezialfall: Äquidistante Stützstellen

$$h \begin{pmatrix} 2 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \frac{3}{h^2} \begin{pmatrix} y_1 - y_0 - h\alpha \\ y_2 - 2y_1 + y_0 \\ y_3 - 2y_2 + y_1 \\ \vdots \\ y_n - 2y_{n-1} + y_{n-2} \\ \beta h - y_n + y_{n-1} \end{pmatrix}$$

3.4.2 Natürliche Splines

An den Rändern x_0 und x_n sind nicht wie bei den kubischen Splines die ersten Ableitungen gegeben sondern die Zweiten, das heißt, man fordert eine konstante *Krümmung*. Die Krümmung wird übrigens wie folgt berechnet:

$$\text{Krümmung} = \frac{y''}{\sqrt{1 + y'^2}}$$

Die Aufgabenstellung ist analog zu den Splines mit der ersten Randableitung, mit folgenden Unterschieden:

- $\varphi(x)$ erfüllt die Randbedingungen $\varphi''(x_0) = 0$ und $\varphi''(x_n) = 0$

Sonderfall: Äquidistante Stützstellen

$$\begin{pmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \frac{3}{h^2} \begin{pmatrix} y_2 - 2y_1 + y_0 \\ y_3 - 2y_2 + y_1 \\ y_4 - 2y_3 + y_2 \\ \vdots \\ y_n - 2y_{n-1} + y_{n-2} \end{pmatrix}$$

3.5 Parameter-Splines**3.5.1 Ebene Kurven**

Bisher haben wir gelernt, einfache Funktionswerte zu interpolieren, doch wie kann man auch mehrdeutige Funktionen (*Kurven*) zeichnen? Die Lösung ist ein zusätzlich eingeführter, monoton wachsender Parameter t , der die *Bogenlänge* repräsentiert. Man berechnet $x(t)$ und $y(t)$ und zeichnet y über x .

Numerische Berechnung der Bogenlänge

In Fällen, wo die Parameterdarstellung nicht von vorneherein gegeben ist, ist es zwingend erforderlich, die Bogenlängen der Teilstücke mit Hilfe des Satzen von *Pythagoras* numerisch zu berechnen (sprich: anzunähern).

$$\begin{aligned} t_0 &:= 0 \\ t_{i+1} &:= t_i + \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \end{aligned}$$

Allgemeine Vorgehensweise

1. Falls keine Bogenlängen gegeben sind, sind diese durch numerische Berechnung – wie oben angegeben – nachzutragen
2. Spline-Interpolation bestimmen für $x(t)$ und $y(t)$
3. Zwischenwerte für x und y an den gleichen Stützstellen berechnen
4. $z(x(t_i), y(t_i))$ bei t_0 beginnend mit monoton wachsendem t zeichnen

Beispiele**Beispiel 3.3** *FIXME***3.5.2** **Raumkurven**

1. Falls keine Bogenlängen gegeben sind, müssen diese analog zum zweidimensionalen Fall durch numerische Berechnung nachgetragen werden:

$$\begin{aligned}t_0 &:= 0 \\t_{i+1} &:= t_i + \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}\end{aligned}$$

2. Spline-Interpolation bestimmen für $x(t)$, $y(t)$ und $z(t)$
3. Zwischenwerte für x , y und z an den gleichen Stützstellen berechnen
4. $z(x(t_i), y(t_i))$ bei t_0 beginnend mit monoton wachsendem t zeichnen

Kapitel 4

Approximation

Der Unterschied zwischen Interpolation und Approximation besteht darin, daß bei der Approximation gegebene Stützstellen nur möglichst gut angenähert werden sollen, während ein Interpolationspolynom immer durch die Stützstellen hindurchgehen muß (Interpolationsbedingung).

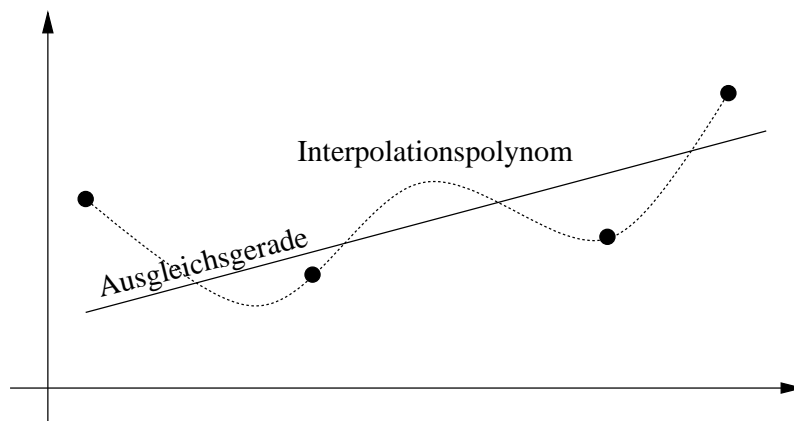


Abbildung 4.1: Motivation

4.1 Die Gauß'sche Fehlerquadratmethode

4.1.1 Aufgabenstellung

Gegeben sind m Stützstellen $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ mit $x, y \in \mathbb{R}$ und $x_1 < x_2 < \dots < x_m$ sowie n reelle Funktionen $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$ gegeben als Funktionensystem. Dabei müssen mehr Stützstellen gegeben sein als Funktionen. (Bei gleicher Anzahl handelte es sich um ein Interpolationsproblem!)

$$\left. \begin{array}{l} (x_i, y_i) \quad i = 1, 2, \dots, m \\ \{\varphi_k(x)\} \quad k = 1, 2, \dots, n \end{array} \right\} m > n$$

Gesucht werden die Koeffizienten $\alpha_1, \alpha_2, \dots, \alpha_n$, so daß gilt:

$$\sum_{i=1}^m \underbrace{(y_i - \varphi(x))}_{r_i}^2 = \min, \quad \varphi(x) = \sum_{k=1}^n \alpha_k \varphi_k(x_i)$$

Bei der Wahl der Funktionen des Funktionensystems besitzt man jede Freiheit, Potenzen von x eignen sich jedoch am besten für einfache Berechnungen. Wählte man stattdessen zum Beispiel die trigonometrischen Funktionen $\sin(x)$, $\cos(x)$, $\sin^2(x)$, $\cos^2(x)$, ... dann wäre dies die diskrete *Fourier-Transformation*.

4.1.2 Aufstellen der Fehlergleichungen

$$\begin{aligned} r_1 &= y_1 - (\varphi_1(x_1)\alpha_1 + \dots + \varphi_n(x_1)\alpha_n) \\ r_2 &= y_2 - (\varphi_1(x_2)\alpha_1 + \dots + \varphi_n(x_2)\alpha_n) \\ &\vdots \\ r_m &= y_m - (\varphi_1(x_m)\alpha_m + \dots + \varphi_n(x_m)\alpha_n) \end{aligned}$$

Sowohl die y_i wie auch die $\varphi_k(x_i)$ sind bekannt. Faßt man die r_i und y_i als Komponenten eines Vektors \vec{r} respektive \vec{y} auf und die $\varphi_k(x_i)$ als Koeffizienten einer Matrix C , die mit dem gesuchten Vektor $\vec{\alpha}$ multipliziert wird, so kann man schreiben:

$$\vec{r} = \vec{y} - C\vec{\alpha}$$

Die Summe der Fehlerquadrate ist dann:

$$\begin{aligned} \sum_{i=1}^m r_i^2 &= \vec{r}^T \vec{r} \\ &= (\vec{y} - C\vec{\alpha})^T (\vec{y} - C\vec{\alpha}) \\ &= (\vec{y}^T - \vec{\alpha}^T C^T) (\vec{y} - C\vec{\alpha}) \\ &= \vec{y}^T \vec{y} - \vec{\alpha}^T C^T \vec{y} - \vec{y}^T C \vec{\alpha} + \vec{\alpha}^T C^T C \vec{\alpha} \\ &= \vec{y}^T \vec{y} - 2\vec{y}^T C \vec{\alpha} + \vec{\alpha}^T C^T C \vec{\alpha} \end{aligned}$$

Beispiel 4.1 (Ausgleichsgerade) *FIXME*

Beispiel 4.2 (Ausgleichsparabel) *FIXME*

4.2 Lösung über Normalgleichungen

4.2.1 Summe der Fehlerquadrate als Quadratische Form

FIXME

4.2.2 Aufstellen der Normalgleichungen

Um die Forderung zu erfüllen, daß die Summe der Fehlerquadrate $Q(\alpha_1, \alpha_2, \dots, \alpha_n)$ minimal sein soll, Müssen folgende Bedingungen erfüllt sein:

Notwendige Bedingungen

Alle partiellen Ableitungen (der *Gradient*) müssen gleich Null sein:

$$\left. \begin{array}{l} \frac{\delta Q}{\delta \alpha_1} = 0 \\ \frac{\delta Q}{\delta \alpha_2} = 0 \\ \vdots \\ \frac{\delta Q}{\delta \alpha_n} = 0 \end{array} \right\} \nabla(Q) = \vec{0}$$

Hinreichende Bedingungen

Die Matrix aller partiellen Ableitungen muß positiv definit sein:

$$A := \left(\frac{\delta^2 Q}{\delta \alpha_i \alpha_k} \right) = \begin{pmatrix} \frac{\delta^2 Q}{\delta \alpha_1 \alpha_1} & \frac{\delta^2 Q}{\delta \alpha_1 \alpha_2} & \cdots & \frac{\delta^2 Q}{\delta \alpha_1 \alpha_n} \\ \frac{\delta^2 Q}{\delta \alpha_2 \alpha_1} & \frac{\delta^2 Q}{\delta \alpha_2 \alpha_2} & \cdots & \frac{\delta^2 Q}{\delta \alpha_2 \alpha_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta^2 Q}{\delta \alpha_n \alpha_1} & \frac{\delta^2 Q}{\delta \alpha_n \alpha_2} & \cdots & \frac{\delta^2 Q}{\delta \alpha_n \alpha_n} \end{pmatrix}, \quad i, k = 1, 2, \dots, n$$

Diese Matrix ist jedoch schon positiv definit nach Konstruktion.

Aufstellen eines linearen Gleichungssystems

$$A\vec{\alpha} - \vec{b} = 0 \Rightarrow A\vec{\alpha} = \vec{b}$$

Es ergibt sich ein lineares Gleichungssystem zur Berechnung von $\alpha_1, \alpha_2, \dots, \alpha_n$ mit einer symmetrischen Koeffizientenmatrix $A = C^T C$ und einem Lösungsvektor $\vec{b} = C^T \vec{y}$.

4.2.3 Nachiteration**Mathematisch**

FIXME

Algorithmisch

1. Bilde:

$$\vec{r} = \vec{y} - C\vec{\alpha}$$

2. Berechne $\Delta\vec{\alpha}$ durch Minimierung von

$$\Delta\vec{r}^T \Delta\vec{r} = \sum_{i=1}^m \Delta r_i^2 = |\vec{r} - C\Delta\vec{\alpha}|^2$$

das heißt, ersetze im Hauptverfahren \vec{y} durch \vec{r} und $\vec{\alpha}$ durch $\Delta\vec{\alpha}$. Die Cholesky-Zerlegung liegt bereits vor.

3. Falls die Korrektur $\Delta\vec{\alpha}$ brauchbar ist (vergleiche Abbrechkriterium):
 - (a) Bilde $\vec{\alpha} = \vec{\alpha} + \Delta\vec{\alpha}$
 - (b) Wiederhole ab 1.

Abbrechkriterium

1. Breche das Verfahren ab, falls

$$\|\Delta \vec{\alpha}\| \geq \frac{1}{4} \|\vec{\alpha}\|$$

2. Andernfalls:

- (a) $\vec{\alpha} = \vec{\alpha} + \Delta \vec{\alpha}$
- (b) $\vec{\alpha}_{\text{alt}} = \Delta \vec{\alpha}$
- (c) Wiederhole ab 2., falls

$$\|\Delta \vec{\alpha}\| < \frac{1}{4} \|\Delta \vec{\alpha}_{\text{alt}}\|$$

- (d) Andernfalls fertig

4.2.4 Fehlerbetrachtung

Löst man das Gleichungssystem $A\vec{\alpha} = \vec{b}$ mit der symmetrischen Matrix $A = C^T C$ und $\vec{b} = C^T \vec{y}$ dann liefert die Rechnung einen mehr oder weniger genauen Vektor $\vec{\alpha}_{\text{num}}$, der sich vom exakten Wert genau um $\Delta \vec{\alpha}$ unterscheidet. Der relative Fehler der erhaltenen Lösung ist damit:

$$\frac{\|\Delta \vec{\alpha}\|}{\|\vec{\alpha}_{\text{num}}\|} \leq \text{cond}(A) \cdot \varepsilon_{\text{masch}}$$

Bei der Berechnung von $\vec{\alpha}$ aus $\vec{r} = \vec{y} - C\vec{\alpha}$ über $C^T C\vec{\alpha} = C^T \vec{y}$ quadriert sich die Konditionszahl der Matrix.

Die *Kondition einer Matrix* $M \in \mathbb{R}^{m \times n}$, $m > n$ wird mit Hilfe der Eigenwerte, genauer gesagt, dem betragsgrößten und dem betragskleinsten Eigenwert berechnet:

$$\text{cond}(M) = \sqrt{\frac{\lambda_{\max}(M^T M)}{\lambda_{\min}(M^T M)}} \geq 1$$

Für eine quadratische und symmetrische Matrix $M \in \mathbb{R}^{n \times n}$, $M = M^T$ gilt:

$$\text{cond}(M) = \sqrt{\frac{\lambda_{\max}(M^2)}{\lambda_{\min}(M^2)}} = \sqrt{\frac{\lambda_{\max}^2(M)}{\lambda_{\min}^2(M)}} = \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)}$$

Hier:

$$\text{cond}(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} = \frac{\lambda_{\max}(C^T C)}{\lambda_{\min}(C^T C)} = \text{cond}^2(C)$$

4.3 Orthogonalisierungsverfahren

Einen alternativen Lösungsweg bietet das *Householder-Verfahren* zur Lösung der Ausgangsmatrix C .

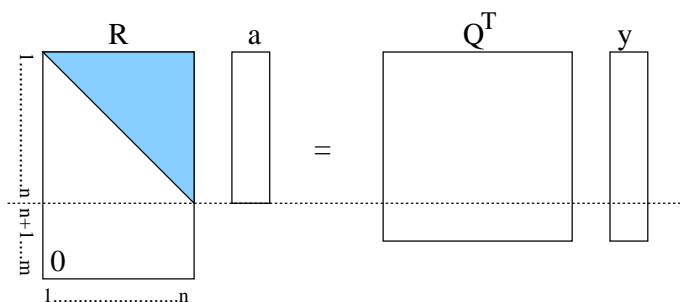
4.3.1 Prinzip des Verfahrens

Man zerlegt die Matrix C in eine orthogonale Matrix Q und eine obere Dreiecksmatrix R mit Hilfe des später noch zu beschreibenden Verfahrens der QR -Zerlegung. Die Fehlergleichungen $\vec{r} = \vec{y} - C\vec{\alpha}$ nach der Zerlegung $C = QR$ errechnen sich wie folgt:

$$\begin{aligned} \vec{r} &= \vec{y} - C\vec{\alpha} & | & C = QR \\ \Rightarrow \vec{r} &= \vec{y} - QR\vec{\alpha} & | & \cdot Q^T \\ \Rightarrow Q^T\vec{r} &= Q^T\vec{y} - Q^TQR\vec{\alpha} & | & Q^TQ = E \\ \Rightarrow Q^T\vec{r} &= Q^T\vec{y} - R\vec{\alpha} \end{aligned}$$

Ziel ist nach wie vor die Minimierung der Summe der Fehlerquadrate. Die Komponenten $n + 1$ bis m des Vektors $Q^T\vec{r}$ sind zwingenderweise gleich den entsprechenden Komponenten des Vektors $Q^T\vec{y}$, liegen also zahlenmäßig fest. $(Q^T\vec{r})^T \cdot Q^T\vec{r}$ wird genau dann Minimal, wenn die ersten n Komponenten des Vektors $Q^T\vec{r}$ gleich Null sind. Für diese Komponenten muß dann gelten:

$$\vec{0} = Q^T\vec{y} - R\vec{\alpha} \Rightarrow R\vec{\alpha} = Q^T\vec{y}$$



4.3.2 Das Householder-Verfahren

Zu lösen ist das Gleichungssystem $\vec{r} = \vec{y} - C\vec{\alpha}$, wobei $\vec{\alpha}$ so zu wählen ist, das die Summe der Quadrate der Komponenten von \vec{r} Minimal wird ($\vec{r}^T\vec{r} = \min$).

1. Orthogonalisierung (QR -Zerlegung) von C durchführen:
 $C = QR$
2. Bilde $\vec{f} = Q^T\vec{y}$ aus den Ersten n Komponenten
3. Löse $R\vec{\alpha} = \vec{f}$ durch Rückwärtseinsetzen. R ist die obere Dreiecksmatrix
4. Falls erforderlich, berechne den Residuenvektor:
 $\vec{r} = \vec{y} - C\vec{\alpha}$
5. Gegebenenfalls nachiterieren

4.3.3 Fehlerbetrachtung

Der relative Fehler berechnet sich wie folgt:

$$\frac{\|\Delta \vec{\alpha}\|}{\|\vec{\alpha}\|} \leq \varepsilon_{\text{masch}} (\text{cond}(C) + \text{cond}^2(C) \frac{\|\vec{r}\|}{\|\vec{y}\|})$$

Ähnlich wie bei der Fehlerberechnung über Normalgleichungen kommt auch hier einmal das Quadrat der Kondition der Matrix C zum tragen, allerdings mit einem Faktor versehen. Falls dieser Summand wesentlich kleiner ist als der Erste – was meistens der Fall ist – dann ist das Orthogonalisierungsverfahren weit überlegen. Es muß also gelten:

$$\begin{aligned} \text{cond}^2(C) \frac{\|\vec{r}\|}{\|\vec{y}\|} &<< \text{cond}(C) \\ \text{bzw. } \text{cond}(C) \frac{\|\vec{r}\|}{\|\vec{y}\|} &<< 1 \end{aligned}$$

4.3.4 QR Zerlegung einer Matrix

Gegeben ist die Matrix $A \in \mathbb{R}^{m \times n}$ mit $m > n$, Gesucht werden eine orthogonale Matrix $Q \in \mathbb{R}^{m \times m}$, $Q^T = Q^{-1}$ und eine obere Dreiecksmatrix $R \in \mathbb{R}^{m \times n}$, so daß die Dreieckszerlegung $QR = A$ gilt.

Man führt n Transformationen mit einer später noch bestimmenden, orthogonalen und symmetrischen Matrix $H_i, i = 1, 2, \dots, n$ durch. Es wird bei einem Startwert $A_1 = A$ begonnen, alle weiteren berechnen sich wie folgt:

$$A_{i+1} = H_i A_i, i = 1, 2, \dots, n$$

Ausgeschrieben ergibt sich:

$$\begin{aligned} A_1 &= A \\ A_2 &= H_1 A_1 = H_1 A \\ A_3 &= H_2 A_2 = H_2 H_1 A \\ &\vdots \\ A_n &= H_{n-1} A_{n-1} = H_{n-1} \cdots H_2 H_1 A \\ A_{n+1} &= H_n A_n = \underbrace{H_n H_{n-1} \cdots H_2 H_1 A}_{Q^T} = R \end{aligned}$$

Formt man dies um, ergibt sich folgendes Bild:

$$\begin{array}{l} R = H_n H_{n-1} \cdots H_2 H_1 A \quad | \cdot H_n^{-1} \\ H_n^{-1} R = H_{n-1} \cdots H_2 H_1 A \quad | \cdot H_{n-1}^{-1} \\ \vdots \\ \underbrace{H_1 H_2 \cdots H_{n-1} H_n R}_Q = A \end{array}$$

Aufbau der Matrix H_i :

$$H_i = E - 2\vec{w}\vec{w}^T \quad \text{mit} \quad \vec{w} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ w_i \\ \vdots \\ w_m \end{pmatrix} \quad \text{und} \quad \vec{w}^T \vec{w} = 1$$

Wir haben also gesehen, daß folgendes gilt:

$$\begin{aligned} R &= Q^T A \\ QR &= A \\ \text{mit } Q &= H_1 H_2 \cdots H_{n-1} H_n \end{aligned}$$

Es folgt noch kurz ein Nachweis der Symmetrie und der Orthogonalität von H_i :

- Symmetrie (zu zeigen: $H_i = H_i^T$):

$$H_i^T = (E - 2\vec{w}\vec{w}^T)^T = E^T - (2\vec{w}\vec{w}^T)^T = E - 2\vec{w}\vec{w}^T = H_i$$

- Orthogonalität (zu zeigen ist: $H_i^T = H_i^{-1} \Rightarrow H_i H_i^T = E$):

$$H_i H_i^T = (E - 2\vec{w}\vec{w}^T)(E - 2\vec{w}\vec{w}^T) = E + 4\vec{w}\vec{w}^T \cdot \vec{w}\vec{w}^T - 4\vec{w}\vec{w}^T = E$$

Kapitel 5

Fourier-Analyse

5.1 Fourier-Analyse analytisch gegebener Funktionen

5.1.1 Aufgabenstellung

Gegeben ist eine periodische Funktion $f(t) : \mathbb{R} \rightarrow \mathbb{R}$, die obendrein stückweise stetig ist und an den Unstetigkeitsstellen endliche Funktionswerte besitzt:

$$\lim_{\varepsilon \rightarrow 0} f(\tau - \varepsilon) = y^-, \lim_{\varepsilon \rightarrow 0} f(\tau + \varepsilon) = y^+$$

Gesucht werden die Fourier-Koeffizienten $a_0, a_1, \dots, a_n, b_1, \dots, b_n$ der Fourier-Reihe

$$\varphi_n(t) = a_0 + 2 \sum_{k=1}^n (a_k \cos(k\omega t) + b_k \sin(k\omega t))$$

so daß gilt:

$$\lim_{n \rightarrow \infty} \varphi_n(t) = f(t)$$

das heißt:

$$f(t) = a_0 + 2 \sum_{k=1}^{\infty} (a_k \cos(k\omega t) + b_k \sin(k\omega t)), \quad a_n, b_n \in \mathbb{R}^1$$

5.2 Verfahren von Goertzl

Gegeben:

$$f_i = f(t_i), \quad i = 0, 1, \dots, n-1 \text{ Periode } T : f(t_i + T) = f_i$$

Mit $\omega = \frac{2\pi}{T}$ und $t_k = \frac{T}{n} \cdot k = k \cdot t_1; t_1 = \frac{T}{n}$ wird $\omega t_k = \omega k t_1 (= \frac{2\pi}{T} \frac{T}{n} k = \frac{2\pi}{n} k)$; $\omega t_1 = \frac{2\pi}{n}$

Formeln zur Berechnung der Fourier-Koeffizienten:

$$\left. \begin{aligned} a_l &= \frac{2}{n} \sum_{k=0}^{n-1} f_k \cdot \cos(l\omega t_k) \\ &= \frac{2}{n} \sum_{k=0}^{n-1} f_k \cdot \cos(l\omega k t_1) \\ b_l &= \frac{2}{n} \sum_{k=0}^{n-1} f_k \cdot \sin(l\omega t_k) \\ &= \frac{2}{n} \sum_{k=0}^{n-1} f_k \cdot \sin(l\omega k t_1) \end{aligned} \right\} l = 1, 2, \dots, q; 2q + 1 \leq n$$

$$a_0 = \dots; a_{\frac{n}{2}} = \dots; b_{\frac{n}{2}} = 0$$

Ziel: Effiziente Berechnung der beiden Summen

$$S_1 = \sum_{k=0}^{n-1} f_k \cdot \cos(l\omega t_k)$$

$$S_2 = \sum_{k=1}^{n-1} f_k \cdot \sin(l\omega t_k)$$

Satz 5.1 Für alle $l \neq \frac{n}{2} \cdot r, r = 0, 1, 2, \dots$ (hier: $l \neq 0, l \neq \frac{n}{2}$) gelten für

$$U_i = \frac{1}{\sin(l\omega t_1)} \cdot \sum_{k=j}^{n-1} f_k \sin((k-j+1)\omega t_1), j = 0, 1, \dots, n-1$$

$$U_n = U_{n+1} = 0$$

die Rekursionsformeln

$$1. U_j = f_j + w \cos(l\omega t_1) \cdot U_{j+1} - U_{j+2}; j = n-1, n-2, \dots, 1, 0$$

$$2. \underbrace{\sum_{k=1}^{n-1} f_k \cdot \sin(l\omega t_k)}_{S_2} = U_1 \cdot \sin(l\omega t_1)$$

$$3. \underbrace{\sum_{k=0}^{n-1} f_k \cdot \cos(l\omega t_k)}_{S_1} = f_0 + \cos(l\omega t_1) \cdot U_1 - U_2$$

Beweis 5.1 FIXME

Daraus ergibt sich:

5.2.1 Algorithmus von Goertzel

$$U_n = U_{n+1} = 0;$$

$$c = \cos(l\omega t_1) (= \cos(l \frac{2\pi}{n})); cc := 2c$$

für $i = n-1, n-2, \dots, 1$:

$$U_i := f_i + cc \cdot U_{i+1} - U_{i+2};$$

$$S_1 := f_0 + c \cdot U_1 - U_2;$$

$$S_2 := U_1 \cdot \sin(l\omega t_1) (= \sin(l \frac{2\pi}{n}))$$

5.2.2 Rechenaufwand

$$\left. \begin{array}{l} 2 \quad \text{trig. Funktionsaufrufe} \\ +(n+2) \quad \text{Mult.} \\ +(2n) \quad \text{Add./Subtr.} \end{array} \right\} l = 1, 2, \dots, q; 2q+1 \leq n \quad \cdot (n-1)$$

ergibt:

$(2n-2) <\text{trig. Fkt.-Aufrufe}> + (n^2+n-2) <\text{Mult.}> + (2n^2-2n) <\text{Add./Subtr.}>$
zum Vergleich: herkömmliche Summenformeln:

$$\begin{array}{l} (2n^2 - 3n + 1) <\text{trig. Fkt.-Aufrufe}> \\ (2n^2 - 3n + 1) <\text{Mult.}> + (2n^2 - 3n + 1) <\text{Add.}> \end{array}$$

Das ist eine erhebliche Einsparung!

5.3 Fast Fourier Transformation (FFT)

Wer sich für diese Thematik interessiert kann eine Ausarbeitung zum Thema "Seminar Numerische Mathematik: Diskrete Fourier-Analyse und -Synthese mittels der Schnellen Fouriertransformation" im World Wide Web unter der Adresse <http://www.informatik.fh-muenchen.de/~ifw99001/> herunterladen.

5.3.1 Aufgabenstellung

Gegeben sind n Stützstellen f_i mit Periode T :

$$f_i = f(t_i) = f\left(\frac{iT}{n}\right); \quad i = 0, 1, \dots, n-1$$

Gesucht werden die Koeffizienten $a_0, a_1, \dots, a_{n/2}, b_1, b_2, \dots, b_{n/2-1}$ im Ansatz

$$\varphi_n(t) = a_0 + 2 \sum_{k=1}^{n/2-1} (a_k \cos(k\omega t) + b_k \sin(k\omega t)) + a_{n/2} \cos\left(\frac{n\omega t}{2}\right)$$

so daß gilt:

$$\varphi_n(t_j) = f_i; \quad i = 0, 1, \dots, n-1$$

Man nennt dies die *Trigonometrische Interpolation*

5.3.2 Herleitung

$$\left. \begin{array}{l} a_k = \frac{1}{n} \sum_{j=0}^{n-1} f_j \cos(k\omega t_j) \\ b_k = \frac{1}{n} \sum_{j=0}^{n-1} f_j \sin(k\omega t_j) \end{array} \right\} k = 0, 1, \dots, \frac{n}{2}$$

darin enthalten: $b_0 = 0; b_{\frac{n}{2}} = 0; a_{\frac{n}{2}} = \frac{1}{n} \sum_{j=0}^{n-1} (-1)^j f_j$

Mit $t_j = j\frac{T}{n}$ und $\omega = \frac{2\pi}{T}$ ist $\omega t_j = \frac{2\pi}{T} \cdot j\frac{T}{n} = \frac{j2\pi}{n}$

$$\begin{aligned}
k = \frac{n}{2} : \quad & \sin(k\omega t_j) = \sin\left(\frac{n}{2}j\frac{2\pi}{n}\right) \equiv \sin(j\pi) = 0 \\
& \Rightarrow b_{\frac{n}{2}} = 0 \\
a_{\frac{n}{2}} = \frac{1}{n} \sum_{j=0}^{n-1} f_j \cdot \underbrace{\cos\left(\frac{n}{2}j\frac{2\pi}{n}\right)}_{(-1)^j} &= \frac{1}{n} \sum_{j=0}^{n-1} f_j \cdot (-1)^j
\end{aligned}$$

Ziel

Berechnung der a_k, b_k mit möglichst wenig Rechenaufwand. Für $n = 2^p, p \in \mathbb{Z}$ im Ansatz:

$$\varphi_n(t_j) = a_0 + 2 \sum_{k=0}^{\frac{n}{2}-1} (a_k \cdot \cos(k\omega t_j) + b_k \cdot \sin(k\omega t_j)) + b_{\frac{n}{2}} \cos\left(\frac{n}{2}\omega t_j\right)$$

5.3.3 Komplexe Darstellung

Fourier-Analyse (A):

$$\begin{aligned}
c_k &= a_k - ib_k \\
&= \frac{1}{n} \sum_{j=0}^{n-1} f_j \cos\left(kj\frac{2\pi}{n}\right) - i \frac{1}{n} \sum_{j=0}^{n-1} f_j \sin\left(kj\frac{2\pi}{n}\right) \\
&= \frac{1}{n} \sum_{j=0}^{n-1} f_j \left(\cos\left(kj\frac{2\pi}{n}\right) - i \sin\left(kj\frac{2\pi}{n}\right) \right) \\
&= \frac{1}{n} \sum_{j=0}^{n-1} f_j \cdot w^{kj} \quad \text{für } k = 0 \dots n-1
\end{aligned}$$

Fourier-Synthese (S):

$$\varphi_n(b_j) = \sum_{k=0}^{n-1} c_k \cdot w^{-kj} \quad \text{für } j = 0 \dots n-1$$

Anmerkung: Der formale Unterschied zwischen den Formeln (S) und (A) ist lediglich das Vorzeichen im Exponenten. Beide Summen können mit demselben mathematischen Verfahren berechnet werden. FFT-Programme berechnen wahlweise – über einen Steuerparameter – (S) oder (A).

5.3.4 Rechenzeitbetrachtung

Ziel: drastische Reduzierung des Rechenaufwandes von $O(n^2)$ auf $O(n \log(n))$.

Menge	Operation
n^2	Komplexe Multiplikationen für $j = 0 \dots n - 1$
n^2	Additionen für $j = 0 \dots n - 1$
$2n$	Funktionsaufrufe für $\sin()$ und $\cos()$
$n^2 + n$	Komplexe Multiplikationen
n^2	Additionen

Tabelle 5.1: Rechenaufwand

Kapitel 6

Numerische Integration

Ziel der Numerischen Integration ist die näherungsweise Berechnung bestimmter Integrale, entweder, weil die Stammfunktion nicht bekannt ist oder die Funktionswerte nur als Tabelle gegeben sind.

6.1 Interpolatorische Quadraturformeln

Früher nannte man Integration auch "Quadratur".

Die Integrationsformeln basieren auf der Polynom-Interpolation. Gegeben sind $n + 1$ Punkte $(x_0, y_0), \dots, (x_n, y_n)$. Nun legt man ein Polynom vom Grad n durch die gegebenen Punkte und integriert über dieses Polynom.

6.1.1 Über Lagrange-Interpolation

Interpolationsformel:

$$f(x) = \sum_{i=0}^n f_i(x) \cdot L_i(x)$$

dann:

$$\begin{aligned} \int_a^b f(x) dx &= \int_a^b \sum_{i=0}^n f_i(x) \cdot L_i(x) dx \\ \int_a^b f(x) dx &= \sum_{i=0}^n f_i(x) \cdot \int_a^b L_i(x) dx \\ &= \sum_{i=0}^n f_i(x) \cdot \underbrace{\int_a^b L_i(x) dx}_{\tilde{\omega}_i} \\ &\quad \text{Gewichtskoeffizienten } \int_a^b L_i(x) dx \end{aligned}$$

Allgemeiner Ansatz:

$$\int_a^b f(x) dx = (b-a) \sum_{i=0}^n \omega_i \cdot f(x_i) + \underbrace{R_{Int.}}_{\text{Restglied}}$$

Das Restglied R_{Int} verschwindet, falls $f(x)$ ein Polynom vom Grad $\leq n$ ist: Polynome bis Höchstgrad n werden exakt integriert.

Berechnung der ω_i statt über Lagrange-Polynome einfacher:

$$f(x) = x^m$$

6.1.2 Berechnung der Gewichtskoeffizienten ω_i

für $f(x) = x^m$, $m \leq n$ ist $R_{Int.} = 0$

Dann;

$$1. \int_a^b x^m dx = \frac{1}{m+1} \cdot x^{m+1} \Big|_a^b = \frac{1}{m+1} (b^{m+1} - a^{m+1})$$

$$2. (b-a) \sum_{i=0}^n \omega_i \cdot f(x_i) = (b-a) \sum_{i=0}^n \omega_i \cdot x_i^m$$

$$(1) = (2) \Rightarrow \frac{1}{m+1} \cdot (b^{m+1} - a^{m+1}) = (b-a) \cdot \sum_{i=0}^n \omega_i \cdot x_i^m$$

bzw.:

$$\sum_{i=0}^n \omega_i \cdot x_i^m = \frac{1}{b-a} \cdot \frac{1}{m+1} \cdot (b^{m+1} - a^{m+1})$$

mit $m = 0, 1, 2, \dots, n$

$$\begin{aligned} m=0: & \quad \omega_0 \cdot 1 + \omega_1 \cdot 1 + \dots + \omega_n \cdot 1 = \frac{1}{b-a} (b-a) \\ m=1: & \quad \omega_0 \cdot x_0^1 + \omega_1 \cdot x_1^1 + \dots + \omega_n \cdot x_n^1 = \frac{1}{b-a} \frac{1}{2} (b^2 - a^2) \\ m=2: & \quad \omega_0 \cdot x_0^2 + \omega_1 \cdot x_1^2 + \dots + \omega_n \cdot x_n^2 = \frac{1}{b-a} \frac{1}{3} (b^3 - a^3) \\ & \quad \vdots \\ m=n: & \quad \omega_0 \cdot x_0^n + \omega_1 \cdot x_1^n + \dots + \omega_n \cdot x_n^n = \frac{1}{b-a} \frac{1}{n+1} (b^{n+1} - a^{n+1}) \end{aligned}$$

\Rightarrow Lineares Gleichungssystem für $\omega_0, \dots, \omega_n$

Koeffizienten Determinante (=VANDERMONDE)

$$\begin{vmatrix} 1 & 1 & \dots & 1 \\ x_0^1 & x_1^1 & \dots & x_n^1 \\ x_0^2 & x_1^2 & \dots & x_n^2 \\ \vdots & \vdots & \dots & \vdots \\ x_0^n & x_1^n & \dots & x_n^n \end{vmatrix} \neq 0 (\Rightarrow \text{LGS ist lösbar})$$

1. für $n=0$: Rechteckregel $x_0 = a, x_1 = b$

$$\int_a^b f(x) dx = (b-a) \cdot \omega_0 \cdot f(x_0) + R_{Int.}$$

Ergebnis:

$$\int_a^b f(x) dx = (b-a) \cdot f(x_0) + R$$

bzw.:

$$\int_a^b f(x)dx = (b-a) \cdot f(a) + R_{\text{Rechteck}}$$

$$R_{\text{Rechteck}} = \frac{1}{2} \cdot h^2 f'(\zeta)$$

mit $a \leq \zeta \leq b$; $h = b - a$

2. für $n = 1$: Trapezregel

$$\int_a^b f(x)dx = (b-a) (\omega_0 f(x_0) + \omega_1 f(x_1)) + R_{\text{Int.}}$$

LGS für ω_0 und ω_1 :

$$\left. \begin{array}{l} 1\omega_0 + 1\omega_1 = 1 \\ 0\omega_0 + h\omega_1 = \frac{1}{2}h \end{array} \right\} \Rightarrow \omega_0 = \omega_1 = \frac{1}{2}$$

$$\int_a^b f(x)dx = h \cdot \frac{1}{2} (f(0) + f(h)) + R_{\text{Trapez}}$$

mit $R_{\text{Trapez}} = -\frac{h^2}{12} f''(\zeta)$ mit $\zeta \in [0, h]$

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

Mit $h := \frac{b-a}{n}$ und $n = \text{Anzahl der Elemente}$, $f(x_i) = f(a + ih)$; $i = 0, 1, \dots, n$ und $x_{i+1} = x_i + h$ gilt:

$$\begin{aligned} \int_a^b f(x)dx &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \dots + \int_{x_{n-1}}^{x_n} f(x)dx \\ &= \frac{h}{2} (f(x_0) + f(x_1)) + \frac{h}{2} (f(x_1) + f(x_2)) + \dots + \frac{h}{2} (f(x_{n-1}) + f(x_n)) + R \\ &= \int_a^b f(x)dx = \frac{h}{2} (f(x_0) + f(x_n)) + h \cdot \sum_{i=1}^{n-1} f(x_i) + R_{\text{Trapez}}(a, b) \end{aligned}$$

6.2 Romberg-Integration

Hohes Ziel der *Romberg-Integration* ist die Erhöhung der Rechengenauigkeit bei einer Verminderung des Rechenaufwandes. Die Ausgangsformel zur Berechnung des Integrals besteht aus der Summe der *Sehnentrapezregel* T und einem Restglied R , die Schrittweite ist h :

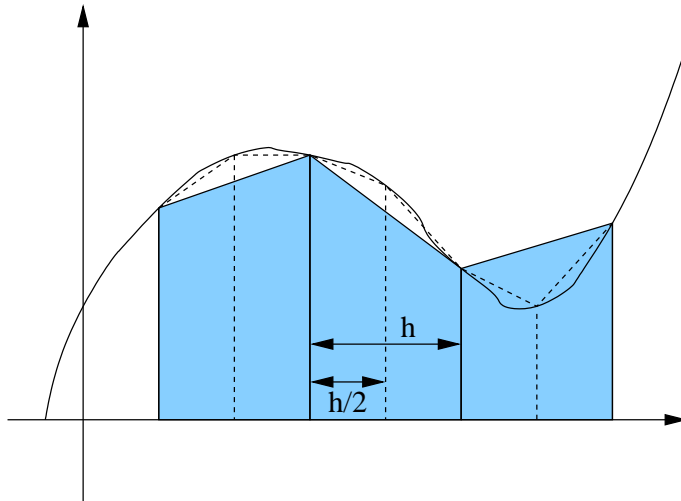
$$I = \int_a^b f(x)dx = T_0(h) + R$$

$$T_0(h) = \frac{h}{2}(f(a) + f(b)) + \sum_{k=1}^{n-1} f(a + kh)$$

$$R = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$$

6.2.1 Herleitung des Romberg-Schemas

Das Ziel ist es, die unbekanntenen Konstanten c_1, c_2, \dots eine nach der anderen zu eliminieren, bis der Fehler im Rahmen einer vorgegebenen Rechengenauigkeit klein genug ist.



c_1 eliminieren

1. Anwenden der Sehnentrapezregel für die halbe Schrittweite:

$$I = T_0\left(\frac{h}{2}\right) + c_1 \frac{h^2}{4} + c_2 \frac{h^4}{16} + c_3 \frac{h^6}{64} + \dots$$

2. Nun kann man c_1 eliminieren, indem man vom Vierfachen der neuen Formel die Alte subtrahiert:

$$4I - I = 4T_0\left(\frac{h}{2}\right) - T_0(h) + \underbrace{\left(4c_1 \frac{1}{4} - c_1\right)}_0 h^2 + \underbrace{\left(4c_2 \frac{1}{16} - c_2\right)}_{c'_2} h^4 + \underbrace{\left(4c_3 \frac{1}{64} - c_3\right)}_{c'_3} h^6 + \dots$$

$$\Rightarrow I = \underbrace{\frac{4T_0\left(\frac{h}{2}\right) - T_0(h)}{3}}_{T_1(h)} + c'_2 h^4 + c'_3 h^6 + \dots$$

 c_2 eliminieren

1. Anwenden der im ersten Schritt erzeugten Formel wiederum für die halbe Schrittweite:

$$I = T_1\left(\frac{h}{2}\right) + c_2 \frac{h^4}{16} + c_3 \frac{h^6}{64} + c_4 \frac{h^8}{256} + \dots$$

2. Nun kann man c_2 eliminieren, indem man vom Sechzehnfachen der neuen Formel die Alte subtrahiert:

$$\begin{aligned} 16I - I &= 16T_1\left(\frac{h}{2}\right) - T_1(h) + \underbrace{(16c_2 \frac{1}{16} - c_2)}_0 h^4 + \underbrace{(16c_3 \frac{1}{64} - c_3)}_{c'_3} h^6 + \underbrace{(16c_4 \frac{1}{256} - c_4)}_{c'_4} h^8 + \dots \\ \Rightarrow I &= \underbrace{\frac{16T_1\left(\frac{h}{2}\right) - T_1(h)}{15}}_{T_2(h)} + c'_3 h^6 + c'_4 h^8 + \dots \end{aligned}$$

6.2.2 Das Romberg-Schema

Führt man auf diese Weise nur lange genug mit dem Eliminieren der Konstanten c_i fort, kann man irgendwann folgendes Schema erkennen:

Das Romberg-Verfahren ist wie folgt rekursiv definiert:

$$\begin{aligned} T_0(h) &= \frac{h}{2}(f(a) + f(b)) + h \sum_{k=1}^{n-1} f(a + kh) \\ T_i(h) &= \frac{2^{2i} T_{i-1}(h/2) - T_{i-1}(h)}{2^{2i} - 1}, \quad i = 1, 2, \dots \end{aligned}$$

6.2.3 Abbrechkriterium

Die jeweils besten Näherungswerte stehen in der obersten Schrägzeile des Rombergschemas. Es wird abgebrochen, wenn die Differenz der letzten beiden "besten" Werte kleiner als eine vorgegebene Schranke ε ist.

$$\frac{|T_i(h) - T_{i-1}(h)|}{|T_i(h)|} \leq \varepsilon \quad \text{oder besser:} \quad |T_i(h) - T_{i-1}(h)| \leq \varepsilon \cdot |T_i(h)|$$

6.2.4 Rechenaufwand

Der Rechenaufwand läßt sich relativ gut anhand der Anzahl der Funktionsaufrufe an den Integranden bestimmen:

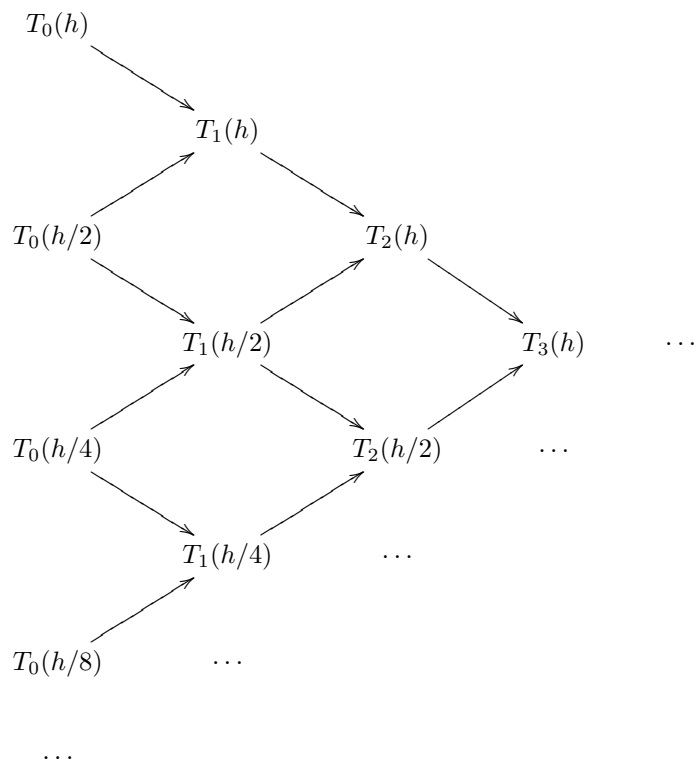


Abbildung 6.1: Das Romberg-Schema

i	Funktionsaufrufe
1	$n + 1$
2	$+n$
3	$+2n$
4	$+4n$

Tabelle 6.1: Anzahl der Funktionsaufrufe bei der Romberg-Integration

6.3 Gauß'sche Quadraturformel

Die bisherigen Formeln basierten alle auf der Polynominterpolation. Zu fest vorgegebenen Stützstellen werden Polynome vom Grad n exakt integriert. Die Gewichtskoeffizienten werden aus einem linearen Gleichungssystem berechnet. Gauß hingegen verzichtet auf die feste Vergabe der Stützstellen. Sein Ansatz lautet:

$$\int_{-1}^1 f(x) dx = \sum_{j=1}^n H_j f(a_j) + R$$

Die Unbekannten a_j und H_j werden so bestimmt, daß Polynome bis zum Grad $2n - 1$ exakt berechnet werden, eine Verbesserung um beinahe das Doppelte.

Beispiel 6.1 (n=2) Bekanntlich werden Polynome bis zum Grad $2n + 1$ exakt integriert, in diesem Fall also bis zum Grad 3. Im folgenden bedienen wir uns aus den Funktionen $f(x) = x^0, x^1, x^2, x^3$:

$$\begin{aligned} x^0 : \int_{-1}^1 x^0 dx &= x|_{-1}^1 = 2 &= H_1 \cdot 1 + H_2 \cdot 1 \\ x^1 : \int_{-1}^1 x^1 dx &= x|_{-1}^1 = 0 &= H_1 \cdot a_1 + H_2 \cdot a_2 \\ x^2 : \int_{-1}^1 x^2 dx &= x|_{-1}^1 = \frac{2}{3} x^2 &= H_1 \cdot a_1^2 + H_2 \cdot a_2^2 \\ x^3 : \int_{-1}^1 x^3 dx &= x|_{-1}^1 = 0 &= H_1 \cdot a_1^3 + H_2 \cdot a_2^3 \end{aligned}$$

Es entstehen 4 nicht-lineare Gleichungssysteme für die 4 Unbekannten H_1, H_2, a_1 und a_2 . Diese Gleichungssysteme zu lösen ist sehr schwierig, man liest die Koeffizienten besser aus einer Tabelle ab.

6.3.1 Transformation auf beliebige Intervallgrenzen

In den seltensten Fällen wird man Integrale in den Grenzen $[-1, 1]$ berechnen wollen, damit man aber die Gauß'schen Quadraturformel anwenden kann, müssen die tatsächlichen Intervallgrenzen transformiert werden.

Aus dem Ansatz $t = \alpha x + \beta$ folgt:

$$\left. \begin{aligned} t = a : x = -1; a = -\alpha + \beta \\ t = b : x = +1; b = \alpha + \beta \end{aligned} \right\} \begin{aligned} \alpha &= \frac{b-a}{2} \\ \beta &= \frac{b+a}{2} \end{aligned}$$

$$\begin{aligned} \Rightarrow \int_a^b f(t) dt &= \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2} + x + \frac{b+a}{2}\right) dx \\ &= \frac{b-a}{2} \sum_{j=1}^n H_j f\left(\frac{b-a}{2} + a_j + \frac{b+a}{2}\right) \end{aligned}$$

Beispiel 6.2

$$\int_0^1 f(t) dt = \frac{1-0}{2} \sum_{j=1}^n H_j f\left(\frac{1-0}{2} a_j + \frac{1+0}{2}\right)$$

Beispiel 6.3 (Qualitätsvergleich) Zum Vergleich der verschiedenen Methoden bietet sich ein Beispiel an, bei dem bereits vorab das exakte Ergebnis bekannt ist, hier etwa die Berechnen der Zahl π .

- Gauß'sche Quadraturformel mit 2 Stützstellen

$$\begin{aligned}\pi &= \int_0^1 \frac{4}{1+x^2} dx = 4 \cdot \frac{1}{2} \left(1 \cdot f\left(\frac{1}{2}a_1 + \frac{1}{2}\right) + 1 \cdot f\left(\frac{1}{2}a_2 + \frac{1}{2}\right) \right) \\ &= 3.14750\dots\end{aligned}$$

Der relative Fehler beträgt 0.18%.

- Führt man dieselbe Rechnung für 3 Stützstellen durch ($n = 3$), ergibt sich ein relativer Fehler von 0.0176%, das ist circa eine Zehnerpotenz besser.
- Bei Anwendung der Simpson-Regel

$$\begin{aligned}\int_0^1 f(x) dx &= \frac{1}{2} \int_{-1}^1 \frac{4}{1 + \left(\frac{1}{2}x + \frac{1}{2}\right)} dx \\ &= 8 \frac{1}{3} \left(\frac{1}{4} + 4 \cdot \frac{1}{5} + \frac{1}{8} \right) \\ &= 3.1333\dots\end{aligned}$$

Der relative Fehler beträgt 0.2629%.

6.4 Doppelintegrale

Aufgaben mit Doppelintegralen können gelöst werden durch Zurückführung auf 2 Einfachintegrale. Die größten Probleme liegen jedoch meist woanders:

FIXME: Finden der Intervallgrenzen

Kapitel 7

Numerische Lösung gewöhnlicher Differentialgleichungen

Gewöhnliche *Differentialgleichungen* beinhalten nur eine unabhängige Variable – im Gegensatz zu partiellen Differentialgleichungen mit mehreren unabhängigen Variablen. Im folgenden werden nur in expliziter Form vorliegende Differentialgleichungen 1. Ordnung behandelt.

7.1 Aufgabenstellung

Ausgehend von einem Startwert (x_0, y_0) werden Näherungswerte für die gesuchte Funktion $y(x)$ an den Stützstellen x_1, x_2, \dots, x_n gesucht. Die Stützstellen müssen nicht äquidistant sein.

7.1.1 Einteilung der Verfahren

- *Einschrittverfahren*: von x_i nach x_{i+1} unter Verwendung von Informationen vom Stützwert x_i
- *Mehrschrittverfahren*: von x_i nach x_{i+1} unter Verwendung von Informationen der Stützwerte x_i, x_{i-1}, \dots
- *Extrapolationsverfahren* basieren auf Verfahren mit Schrittweitenverkleinerung und dann $h \rightarrow 0$

Wir werden uns allerdings auf die Einschrittverfahren beschränken.

7.2 Verfahren von Euler/Cochy

FIXME

7.3 Verbessertes Polygonzugverfahren

FIXME

7.4 Verfahren von Runge-Kutta

$$K1 := h \cdot f(x, y_1, y_2, \dots, y_n) \quad ; \quad i = 1, 2, \dots, n$$

$$K2 := h \cdot f\left(x + \frac{h}{2}, y_1 + \frac{K1_1}{2}, y_2 + \frac{K1_2}{2}, \dots, y_n + \frac{K1_n}{2}\right) \quad ; \quad i = 1, 2, \dots, n$$

$$K3 := h \cdot f\left(x + \frac{h}{2}, y_1 + \frac{K2_1}{2}, y_2 + \frac{K2_2}{2}, \dots, y_n + \frac{K2_n}{2}\right) \quad ; \quad i = 1, 2, \dots, n$$

$$K4 := h \cdot f(x + h, y_1 + K3_1, y_2 + K3_2, \dots, y_n + K3_n) \quad ; \quad i = 1, 2, \dots, n$$

Daraus kann man berechnen:

$$K := \frac{1}{6}(K1 + 2K2 + 2K3 + K4) \quad ; \quad i = 1, 2, \dots, n$$

Die neuen Werte für x und y berechnen sich dann wie folgt:

$$y = y + K \quad ; \quad x = x + h$$

7.5 Systeme von Differentialgleichungen 1. Ordnung

7.5.1 Differentialgleichungen höherer Ordnung

Alle Differentialgleichungen der Ordnung n lassen sich zu einem System von n Differentialgleichungen 1. Ordnung umschreiben, vorausgesetzt, $y^{(n)}$ liegt in expliziter Form vor:

$$y^{(n)} = f(x, y, y', \dots, y^{(n-1)})$$

mit den n Anfangsbedingungen

$$y(x_0) = y_0; y'(x_0) = y'_0; \dots; y^{(n-1)}(x_0) = y_1^{(n-1)}$$

Man setzt

$$\begin{aligned} \varphi_1 = y &\Rightarrow \varphi'_1 = y' = \varphi_2 \\ \varphi_2 = y' &\Rightarrow \varphi'_2 = y'' = \varphi_3 \\ &\vdots \\ \varphi_n = y^{(n-1)} &\Rightarrow \varphi'_n = y^{(n)} = f(x, y, y', \dots, y^{(n-1)}) = f(x, \varphi_1, \varphi_2, \dots, \varphi_n) \end{aligned}$$

Es ergeben sich n Differentialgleichungen 1. Ordnung:

$$\begin{aligned}
 \varphi_1' &= \varphi_2 & \text{mit} & \quad \varphi_1(x_0) = y_0 \\
 \varphi_2' &= \varphi_3 & \text{mit} & \quad \varphi_2(x_0) = y_0' \\
 & \vdots & & \\
 \varphi_n' &= f(x, \varphi_1, \varphi_2, \dots, \varphi_n) & \text{mit} & \quad \varphi_n(x_0) = y_0^{(n-1)}
 \end{aligned}$$

7.5.2 Polygonzugverfahren

Im Wesentlichen führt man das Polygonzugverfahren für ein ganzes System von Differentialgleichungen auf die parallele Berechnung einzelner Differentialgleichungen zurück. Versieht man diese mit einem Index, kann man schreiben:

$$y_i' = f_i(x, \overline{y}^T) = f_i(x, y_1, y_2, \dots, y_n) \quad \text{mit} \quad y_i(x_0) = y_{i0}$$

Ausgeschrieben wäre das:

$$\begin{aligned}
 y_1' &= f(x, y_1, y_2, \dots, y_n) & \text{mit} & \quad y_1(x_0) = y_{10} \\
 y_2' &= f(x, y_1, y_2, \dots, y_n) & \text{mit} & \quad y_2(x_0) = y_{20} \\
 & \vdots & & \\
 y_n' &= f(x, y_1, y_2, \dots, y_n) & \text{mit} & \quad y_n(x_0) = y_{n0}
 \end{aligned}$$

7.5.3 Verfahren von Runge-Kutta

Beim Verfahren von Runge-Kutta für ein ganzes System von Differentialgleichungen müssen in jedem Schritt nacheinander folgende Berechnungen durchgeführt werden:

$$\begin{aligned}
 K1_i &:= h \cdot f_i(x, y_1, y_2, \dots, y_n) & ; & \quad i = 1, 2, \dots, n \\
 K2_i &:= h \cdot f_i\left(x + \frac{h}{2}, y_1 + \frac{K1_1}{2}, y_2 + \frac{K1_2}{2}, \dots, y_n + \frac{K1_n}{2}\right) & ; & \quad i = 1, 2, \dots, n \\
 K3_i &:= h \cdot f_i\left(x + \frac{h}{2}, y_1 + \frac{K2_1}{2}, y_2 + \frac{K2_2}{2}, \dots, y_n + \frac{K2_n}{2}\right) & ; & \quad i = 1, 2, \dots, n \\
 K4_i &:= h \cdot f_i(x + h, y_1 + K3_1, y_2 + K3_2, \dots, y_n + K3_n) & ; & \quad i = 1, 2, \dots, n
 \end{aligned}$$

Daraus kann man berechnen:

$$K_i := \frac{1}{6}(K1_i + 2K2_i + 2K3_i + K4_i) \quad ; \quad i = 1, 2, \dots, n$$

Die neuen Werte für x und y berechnen sich dann wie folgt:

$$y_i = y_i + K_i \quad ; \quad x = x + h$$

Kapitel 8

Lösung nichtlinearer Gleichungen

Aufgabenstellung und Allgemeines

Gegeben: $x \in \mathbb{R}; f(x) : \mathbb{R} \rightarrow \mathbb{R}$

Gesucht: $\xi \in \mathbb{R}$ mit $f(\xi) = 0$

Prinzip der numerischen Verfahren

Berechne Näherungswerte $x^{(0)}, x^{(1)}, \dots, x^{(n)}$ mit dem Ziel:

$$\lim_{n \rightarrow \infty} x^{(n)} = \xi$$

Konvergenzordnung

Definition 8.1 Man nennt man p die Ordnung des Verfahrens, falls gilt:

$$\lim_{n \rightarrow \infty} \frac{x^{(n-1)} - \xi}{(x^{(n-1)} - \xi)^p} = M$$

M sei eine unbekannte Konstante. Dann ist die Aussage dieser Definition:

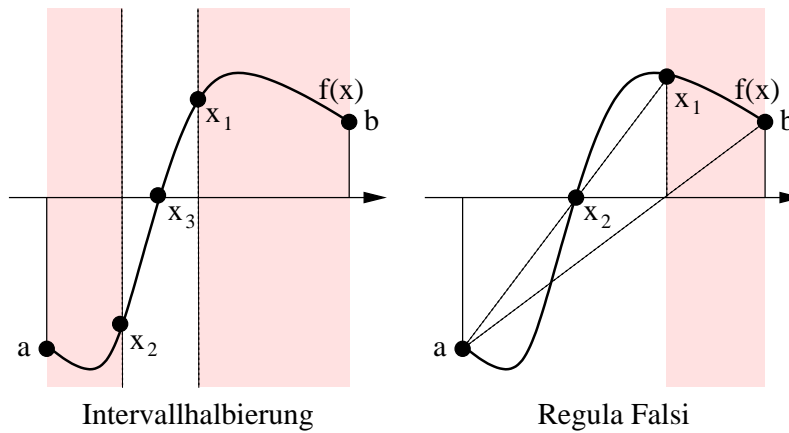
$$\lim_{n \rightarrow \infty} \left\{ \left| x^{(n+1)} - \xi \right| = M \left| x^{(n)} - \xi \right|^p \right\}$$

8.1 Interallhalbierung

Gegeben: $f(x)$ stetig in $[a, b]$, $\text{sign}(a) \neq \text{sign}(b)$

Algorithmus:

1. $SA := \text{sign}(f(a)); SB := \text{sign}(f(b))$
2. $k := 0$
3. Wiederhole:
 - (a) $x^{(k)} := \frac{a+b}{2}$



- (b) Berechne: $F_0 := f(x^{(k)})$
- (c) Abbruchkriterium:
Falls $|F_0|$ oder das Intervall $[a, b]$ klein genug ist, ist $x^{(k)}$ die Nullstelle
- (d) Verfahren fortsetzen:
 - Falls $\text{sign}(F_0) = SA$: $a := x^{(k)}$; $SA := \text{sign}(F_0)$
 - Andernfalls: $b := x^{(k)}$; $SB := \text{sign}(F_0)$
- (e) $k := k + 1$

Konvergenzordnung: $p=1$

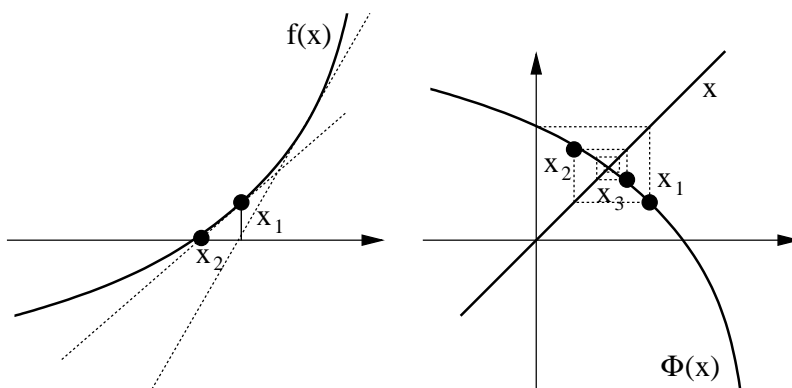
8.2 Regula Falsi

Gegeben: $f(x)$ stetig in $[a, b]$, $\text{sign}(a) \neq \text{sign}(b)$

Algorithmus:

1. $FA := f(a)$; $FB := f(b)$
2. $k := 0$
3. Wiederhole:
 - (a) $x^{(k)} := b - FB \frac{b-a}{FB-FA}$
 - (b) Berechne: $F_0 := f(x^{(k)})$
 - (c) Falls Abbruchkriterium erfüllt: $x^{(k)}$ ist Nullstelle.
 - (d) Andernfalls:
 - Falls $\text{sign}(F_0) = \text{sign}(FA)$: $A := x^{(k)}$; $FA := F_0$
 - Andernfalls: $B := x^{(k)}$; $FB := F_0$

Konvergenzordnung: $p=1$



Newton-Verfahren

Picard-Verfahren

x_2

8.3 Newton-Verfahren

Gegeben: $x^{(k)}, f(x^{(k)})$

Vorgehensweise: Man berechnet den Schnittpunkt der Tangente in $(x^{(k)}, f(x^{(k)}))$ mit der x-Achse und benutzt diesen Wert als neues $x^{(k)}$:

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

Voraussetzung: Der Startwert $x^{(0)}$ stellt bereits eine einigermaßen gute Näherung dar:

$$|f(x)f''(x)| < (f'(x)^2) \forall x \in \{x_0, x_1, \dots\}$$

Kpnvergenzordnung: $p=2$

8.4 Picard-Iterationsverfahren

Gesucht werden die Nullstellen der Funktion $f(x)$, das Verfahren geht folgendermaßen:

1. Suche Näherungswert $x^{(0)}$ für die Nullstellen (Startwert für die Iteration)
2. Bringe $f(x) = 0$ auf die neue Form $x = \varphi(x)$
3. Untersuche Konvergenzeigenschaften in der Nähe von $x^{(0)}$
4. Iteriere:

$$x^{(k+1)} := \varphi(x^{(k)}) \quad , \quad k = 0, 1, \dots$$

5. Falls Iteration konvergent: letztes $x^{(k+1)}$ ist Näherungswert für die Nullstelle, Andernfalls versuche ab Punkt 2. mit neuer Form $x = \varphi(x)$

Voraussetzung: $|\varphi'(x)| < 1$ im betrachteten Intervall

Konvergenzordnung: $p = 2$

Beispiel 8.1 Gegeben ist die Funktion $f(x) = x - \cos(x)$, deren Nullstellen sich nicht analytisch berechnen lassen. Man forme wie folgt um:

$$x - \cos(x) = 0 \Rightarrow x = \underbrace{\cos(x)}_{\varphi(x)}$$

Nun kann man der Reihe nach berechnen:

$$\begin{aligned} x_0 = 0 : \quad \cos(0) &= 0.54 \\ x_1 = 0.52 : \quad \cos(0.54) &= \dots \end{aligned}$$

Aufgabentypen

1. Fehleranalyse
 - Relative Fehler ausrechnen
2. Lineare Gleichungssysteme
 - Gauß-Jordan Eliminationsverfahren
 - Gesamtschrittverfahren nach Jacobi
 - Einzelschrittverfahren nach Seidel
3. Interpolation
 - Lagrange-Interpolation
 - Newton Verfahren
 - Spline-Interpolation mit kubischen Splines
4. Approximation
 - Gauß'sche Fehlerquadratmethode
 - Fehlergleichungen aufstellen
5. Fourier-Analyse
 - FIXME
6. Integration
 - Simpson-Regel
 - Romberg-Integration
 - Gauß-Integration
7. Differentialgleichungen
 - Euler-Cauchy
 - Verbessertes Polygonzuverfahren
 - Runge-Kutta Verfahren
8. Nullstellen
 - Intervallhalbierung
 - Regula Falsi
 - Newton-Verfahren
 - Picard-Verfahren

Taylor-Reihen

1. $f(x) = R^1 \rightarrow R^1 : y := f(x); y \in R^1$

$$y + \Delta y := f(x + \Delta x) = f(x) + f'(x) \cdot \Delta x + f''(x) \cdot \frac{\Delta x^2}{2!} + f'''(x) \cdot \frac{\Delta x^3}{3!} \dots$$

$$f(x + \Delta x) - f(x) = y + \Delta y - y = f'(x) \cdot \Delta x + f''(x) \cdot \frac{\Delta x^2}{2!} + \dots$$

$$\Delta y = f'(x) \cdot \Delta x + f''(x) \cdot \frac{\Delta x^2}{2!} \dots$$

Daraus folgt für den *absoluten Fehler*:

$$|\Delta x| \approx |f'(x) \cdot \Delta x|$$

und für den *relativen Fehler*:

$$\left| \frac{\Delta y}{y} \right| \approx \left| f'(x) \cdot \frac{x}{f(x)} \cdot \frac{\Delta x}{x} \right| = \left| f'(x) \cdot \frac{x}{f(x)} \right| \cdot \left| \frac{\Delta x}{x} \right|$$

$$|\varepsilon_y| \approx \left| f'(x) \cdot \frac{x}{f(x)} \right| \cdot |\varepsilon_x|$$

2. $f(x_1, x_2) : R^2 \rightarrow R^1, y := f(x_1, x_2); y \in R^1$

$$y + \Delta y := f(x_1 + \Delta x_1, x_2 + \Delta x_2) = f(x_1, x_2) + \frac{\partial f}{\partial x_1} \cdot \Delta x_1 + \frac{\partial f}{\partial x_2} \cdot \Delta x_2 + \frac{\partial^2 f}{\partial x_1^2} \cdot \frac{\Delta x_1^2}{2!} + \frac{\partial^2 f}{\partial x_2^2} \cdot \frac{\Delta x_2^2}{2!} + \dots$$

$$y + \Delta y - y = f(x_1, x_2) + \frac{\partial f}{\partial x_1} \cdot \Delta x_1 + \frac{\partial f}{\partial x_2} \cdot \Delta x_2 - f(x_1, x_2)$$

$$\Delta y = \frac{\partial f}{\partial x_1} \cdot \Delta x_1 + \frac{\partial f}{\partial x_2} \cdot \Delta x_2$$

Daraus folgt für den *absoluten Fehler*:

$$|\Delta y| \leq \left| \frac{\partial f}{\partial x_1} \cdot \Delta x_1 \right| + \left| \frac{\partial f}{\partial x_2} \cdot \Delta x_2 \right|$$

und für den *relativen Fehler*:

$$\underbrace{\frac{\Delta y}{y}}_{\varepsilon_y} \leq \underbrace{\left| \frac{\partial f}{\partial x_1} \cdot \frac{x_1 \cdot \Delta x_1}{f(x_1, x_2) \cdot x_1} \right|}_{\varepsilon_{x_1}} + \underbrace{\left| \frac{\partial f}{\partial x_2} \cdot \frac{x_2 \cdot \Delta x_2}{f(x_1, x_2) \cdot x_2} \right|}_{\varepsilon_{x_2}}$$

$$|\varepsilon_y| \leq \left| \frac{\partial f}{\partial x_1} \cdot \frac{x_1}{f(x_1, x_2)} \right| \cdot |\varepsilon_{x_1}| + \left| \frac{\partial f}{\partial x_2} \cdot \frac{x_2}{f(x_1, x_2)} \right| \cdot |\varepsilon_{x_2}|$$

3. $f(x_1, x_2, \dots, x_n) : R^n \rightarrow R^1; y := f(x_1, \dots, x_n); y \in R^1$

$$y + \Delta y := f(x_1 + \Delta x_1, \dots, x_n + \Delta x_n) = f(x_1, \dots, x_n) + \sum_{i=1}^n \frac{\partial f}{\partial x_i} \cdot \Delta x_i + \dots$$

$$\Delta y = f(x_1 + \Delta x_1, \dots, x_n + \Delta x_n) - f(x_1, \dots, x_n) = \sum_{i=1}^n \frac{\partial f}{\partial x_i} \cdot \Delta x_i$$

Daraus folgt für den *absoluten Fehler*:

$$|\Delta y| \leq \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \right| \cdot |\Delta x_i|$$

und für den *relativen Fehler*:

$$\underbrace{\frac{\Delta y}{y}}_{|\varepsilon_y|} \leq \sum_{i=1}^n \left| \frac{\partial f}{\partial x_i} \cdot \frac{x_i}{f(x_1, \dots, x_n)} \right| \cdot \underbrace{\left| \frac{\Delta x_i}{x_i} \right|}_{|\varepsilon_{x_i}|}$$

$$|\varepsilon_y| \leq \sum_{i=1}^n \underbrace{\left| \frac{\partial f}{\partial x_i} \cdot \frac{x_i}{f(x_1, \dots, x_n)} \right|}_{\chi_i} \cdot |\varepsilon_{x_i}|$$

Konditionszahlen χ_i

Die **Konditionszahl** entspricht der Verstärkung der *relativen Eingangsfehler*.
Je größer die Konditionszahl umso unbrauchbarer das Ergebnis. Man spricht
von *gut* und *schlecht konditioniert*.

Index

- Abbrechfehler, 1
- Abbruchkriterium, 15
- Approximation, 16

- Cramer'sche Regel, 4

- Determinante, 5
- Diagonaldominant, 7
- diagonaldominant, 12
- Differentialgleichungen, 45
- direkte Verfahren, 4
- Dividierte Differenz, 19
- Doppelintegrale, 44
- Dreieckszerlegung, 6, 30

- Einschrittverfahren, 45
- Einzelschrittverfahren, 15
- Euler/Cochy, 45
- Exponent, 1
- Extrapolationsverfahren, 45

- Fourier-Analyse, 35
- Fourier-Koeffizienten, 32
- Fourier-Reihe, 32
- Fourier-Synthese, 35
- Fourier-Transformation, 26

- Gesamtschrittverfahren, 14
- Gleitpunktzahlen, 1
- Gradient, 27

- HORNER-Schema, 20
- Householder-Verfahren, 28

- indirekte Verfahren, 4
- Integerzahlen, 1
- Interallhalbierung, 48
- Interpolation, 16
- Interpolationsbedingung, 16
- Inverse Matrix, 8

- Jacobi, 14

- Kondition einer Matrix, 28
- Konditionszahlen, 2, 54
- Konvergenzordnung, 48
- Kurve, 23

- Lagrange-Polynome, 17
- Lineare Interpolation, 17

- Mantisse, 1
- Matrixinversion, 8
- Matrixnorm, 10
- Mehrschrittverfahren, 45

- Nachiteration, 8
- Newton-Verfahren, 50

- Obere Dreiecksform, 4

- periodische Funktion, 32
- Picard-Iterationsverfahren, 50
- Polygonzugverfahren, 46
- positiv definit, 12
- Pythagoras, 23

- Quadratische Interpolation, 17

- Regula Falsi, 49
- Residuenvektor, 8, 15
- Romberg-Integration, 40
- Romberg-Schema, 41
- Rundungsfehler, 1
- Runge-Kutta, 46

- Schiefe Asymptote, 20
- Seidel, 15
- Startvektor, 14
- Symmetrie, 31
- symmetrisch, 12

- Trigonometrische Interpolation, 34

- Untere Dreiecksform, 5

VANDERMONDE, 38
Vektornorm, 9
Verfahrensfehler, 1