

Numerik 1

Martin Reißel
Matthias Grajewski

4. März 2025

Organisatorisches

- Prof. Dr. Martin Reißel
Fachbereich 9
FH Aachen, Campus Jülich, Raum 01C13
Mail: reissel@fh-aachen.de
- Übung:
 - Ausgabe, Abgabe, Korrektur im Wochenrhythmus (ILIAS)
 - aktive Teilnahme wird kontrolliert
- Materialien in ILIAS
- Klausur in Prüfungsperiode Juli

Inhaltsverzeichnis

1 Einleitung	1
2 Lineare Gleichungssysteme – Direkte Löser I	10
2.1 Einleitung	10
2.2 Gaußelimination	15
2.3 LU-Zerlegung	18
2.4 LU-Zerlegung für beliebige reguläre Matrizen	23
2.5 Cholesky-Zerlegung	30
3 Fehlerbetrachtung	36
3.1 Gut gestellte Probleme	36
3.2 Fehlergrößen	37
3.3 Fehlerquellen	40
3.3.1 Floating-Point-Darstellung von Zahlen	41
3.3.2 Eingabefehler, Kondition	45
3.3.3 Verfahrensfehler	51
3.3.4 Akkumulierter Rundungsfehler, Floating-Point-Arithmetik	52
3.4 Zusammenfassung	55
4 Lineare Gleichungssysteme – Direkte Löser II	57
4.1 Fehlerbetrachtung für lineare Gleichungssysteme	57
4.2 LU-Zerlegung mit Pivot-Strategie	63
4.3 Orthogonale Transformationen	69
4.3.1 Grundlagen	69
4.3.2 Verfahren von Givens	71
4.3.3 Verfahren von Householder	79
5 Iterative Löser für lineare Gleichungssysteme	85
5.1 Grundlagen	85
5.2 Lineare stationäre Verfahren	87
5.2.1 Grundlagen, Konvergenz	87
5.2.2 Jacobi-Verfahren (Gesamtschritt-Verfahren)	96
5.2.3 Gauß-Seidel-Verfahren (Einzelschritt-Verfahren)	100
5.2.4 Nachiteration	103
5.2.5 Relaxationsverfahren	103
5.2.6 Symmetrische Varianten	107
5.3 Nichtstationäre Iterationen	108
5.3.1 Steepest-Descent-Verfahren (Methode des steilsten Abstiegs)	108
5.3.2 Konjugierte Gradienten (CG)	112
5.4 Vergleich verschiedener Verfahren	113
5.5 Dünn besetzte Matrizen	114

6 Eigenwerte	117
6.1 Motivation	117
6.2 Theorie	120
6.3 Vektoriteration	123
6.4 Jacobi-Verfahren	126
6.5 QR-Verfahren, Hessenberg-Transformation, Shift-Technik	129
7 Nichtlineare Gleichungen	135
7.1 Einleitung	135
7.2 Einfache Verfahren und ihre geometrische Interpretation	135
7.2.1 Bisektions-Verfahren	135
7.2.2 Regula-Falsi-Verfahren	137
7.2.3 Sekanten-Verfahren	140
7.2.4 Taylor-Reihen-basierte Verfahren, Newton-Verfahren	142
7.3 Stationäre Iterationen, Banachscher Fixpunktsatz	145
7.4 Newton-Verfahren	158
8 Interpolation	163
8.1 Einleitung	163
8.2 Polynominterpolation	163
8.3 Splines	170
8.4 B-Splines	178
8.5 Interpolierende Kurven	183
9 Approximation	189
9.1 Lineare Ausgleichsprobleme	189
9.2 QR Zerlegung für lineare Ausgleichsprobleme	193
9.3 CGLS	194
9.4 Pseudoinverse	196
9.5 Singulärwertzerlegung	198
9.6 Regularisierung schlecht konditionierter Probleme	201
9.6.1 Grundlagen	201
9.6.2 Abgeschnittene Singulärwertzerlegung (TSVD)	206
9.6.3 Tikhonov Regularisierung	208
9.6.4 Parameterwahl	211
10 Numerische Integration	213
10.1 Einleitung	213
10.2 Newton-Cotes Quadraturformeln	213
10.3 Gauß-Integration	222
10.4 Extrapolation	227
10.5 Adaptive Verfahren	237
11 Optimierung ohne Nebenbedingungen	242
11.1 Grundlagen	242
11.2 Abstiegsverfahren	245
11.2.1 Grundlagen	245
11.2.2 Liniensuche	247
11.2.3 Methode des steilsten Abstiegs (Steepest-Descent)	249
11.3 CG-Verfahren	252
11.4 Newton-Verfahren	256
11.4.1 Grundlagen	256
11.4.2 Trust-Region-Verfahren	259
11.4.3 Quasi-Newton-Verfahren	263
11.5 Nichtlineare Ausgleichsprobleme	267

12 Diskrete Fourier- und Wavelettransformation	273
12.1 Motivation	273
12.2 Fourierreihen	274
12.3 Diskrete Fouriertransformation	280
12.4 Schnelle Fouriertransformation (FFT)	285
12.5 Diskrete Wavelettransformation	288

Kapitel 1

Einleitung

Was ist Numerik?

Numerical analysis is the study of algorithms for the problems of continuous mathematics
(Lloyd Trefethen 1992)

- das heißt konkret:
 - “continuous mathematics”: reelle oder komplexe Zahlen sind beteiligt (in Abgrenzung zur diskreten Mathematik). Dies umfasst die Analysis und Teile der linearen Algebra
 - Entwicklung von (effizienten) Berechnungsmethoden
 - Bei Berechnung mit einem Computer: **Floating-Point**-Zahlen und -Arithmetik
 - Konsequenz: Studium von Rundungsfehlern
- Hintergrund: Viele mathematische Objekte sind zwar wohldefiniert, aber nicht endlich berechenbar

Beispiel 1 (*Auswertung der Exponentialfunktion*).

$$\blacktriangleright e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

- Ziel: Man berechne $e^{\frac{1}{2}}$
- Problem: unendliche Summation i. A. nicht direkt berechenbar \Rightarrow *abgebrochene* Summation
- Wir lösen statt des ursprünglichen Problems ein **Ersatzproblem**:

$$e^x \approx \sum_{k=0}^n \frac{x^k}{k!}$$

- n ist ein neuer freier Parameter, den wir geeignet wählen müssen
- Was bedeutet “geeignet”?

- ▶ n klein \Rightarrow eventuell zu viel vernachlässigt, Wert ungenau approximiert
- ▶ n groß \Rightarrow Berechnung aufwändig, Gewinn an Genauigkeit unter Umständen gering
- ▶ allgemein: stimme Parameter so ab, dass Genauigkeit im Verhältnis zu übrigen Fehlerquellen steht
- ▶ Ersatzproblem wird gelöst, nicht Originalproblem
- ▶ jetzt: Implementierung des Ersatzproblems in ein Computerprogramm
- ▶ zusätzliche Fehlerquelle: Zahldarstellung, inexakte Arithmetik
- ▶ verschiedene Summationsreihenfolgen denkbar:

$$s_1(x) = 1 + x + \frac{x^2}{2} + \dots + \frac{x^n}{n!}$$

$$s_2(x) = \frac{x^n}{n!} + \dots + \frac{x^2}{2} + x + 1$$

- ▶ bei exakter Arithmetik: Summationsreihenfolge ohne Bedeutung
- ▶ für $x = \frac{1}{2}$ ist der exakte Wert $e^{\frac{1}{2}} \approx 1.6487$
- ▶ Floating-Point-Rechnung mit 3 Stellen Genauigkeit liefert

n	1	2	3	4	5
$s_1(0.5)$	1.5	1.62	1.64	1.64	1.64
$s_2(0.5)$	1.5	1.62	1.65	1.65	1.65

- ▶ Die Summationsreihenfolge beeinflusst das Ergebnis!
- ▶ Ein Abbruch bei $n = 4$ ist für $x = 1/2$ angemessen
- ▶ aber: für $x = 2$ liefert erst $n = 8$ eine Genauigkeit von 3 Stellen: $e^2 \approx 7.385$

n	2	4	6	7	8	9
$s_1(2)$	5.00	7.00	7.36	7.39	7.40	7.40

Die Parameterwahl im Ersatzproblem kann von den Eingabedaten abhängen!

- **bisher gesehen:** innermathematische Anwendung numerischer Verfahren
- mehr Beispiele hierzu:
 - analog zu oben: Man berechne $\sin \alpha$
 - In der Analysis: Definition von $\sin \alpha$ üblicherweise durch unendliche Reihe (Taylorreihe):

$$\sin \alpha = \sum_{n=0}^{\infty} (-1)^n \frac{\alpha^{2n+1}}{(2n+1)!}$$

- Aus der Analysis bekannt: Diese Reihe konvergiert für jedes $\alpha \Rightarrow \sin$ wohldefiniert
- aber: Grenzwert i. A. nicht endlich berechenbar \Rightarrow numerische Verfahren zur näherungsweisen Berechnung

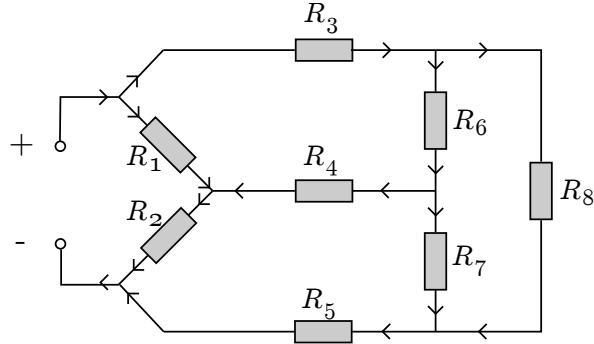
- analog: $\cos \alpha$
- $\int_a^b f(x) dx$ existiert immer für stetiges f , ist aber i. A. nicht unmittelbar berechenbar, wenn man keine Stammfunktion von f findet \Rightarrow numerische Verfahren zur näherungsweisen Berechnung
- Berechnung von \sqrt{x} (siehe Übung) i. A. nicht exakt durchführbar \Rightarrow numerische Verfahren zur näherungsweisen Berechnung
- Verallgemeinerung: Nullstellen von Polynomen höheren Grades i. A. nicht direkt berechenbar \Rightarrow numerische Verfahren zur näherungsweisen Berechnung
- ...
- jetzt: **außermathematische** Anwendungen von numerischen Verfahren: **wissenschaftliches Rechnen / Numerische Simulation**

Bemerkung 2.

- ◊ **Numerische Simulation:** *Verhaltensvorhersage eines physikalischen Systems durch mathematische Modelle und numerische Verfahren*
- ◊ physikalisches System: z. B. beliebiges Bauteil, Auto, Flugzeug, Atom, Planet, ...
- ◊ wichtiges Einsatzgebiet der numerischen Simulation: Produktentwicklung in der Industrie
- ◊ **klassisches Vorgehen in der Produktentwicklung (vereinfacht):**
 - ◊ grobe Vorauslegung der Konstruktion
 - ◊ Prototyp bauen und testen
 - ◊ wenn Test erfolgreich: Prototyp kann in Serie gehen
 - ◊ wenn Test fehlgeschlagen: Ein veränderter Prototyp wird getestet
 - ◊ Iteration bis zum Erfolg
- ◊ **Alternative:**
 - ◊ virtueller Prototyp: vollständige Analyse durch **Numerische Simulation**
 - ◊ davon ausgehend: endgültige Auslegung der Konstruktion
 - ◊ am Ende: **Validierung** durch *einen* realen Prototypen
- ◊ Vorteil: spart i. d. R. Geld und Zeit (Extrembeispiel: Konstruktion eines Verkehrflugzeugs)
- ◊ *Numerische Simulation ist Big Business:* Die Ansys Corp, einer der führenden Anbieter von Software für Simulation, wies für 2012 einen Umsatz von 807,7 Mio \$ aus (2013 Investor Day Executive Summary, <http://investors.ansys.com>)
- ◊ Die numerische Mathematik stellt mathematische Grundlagen und effiziente Algorithmen zur Numerischen Simulation bereit.
- ◊ **wissenschaftliches Rechnen:** ähnlich, aber eher durch wissenschaftliches Interesse motiviert (z. B. Simulationen, um die Entstehung von Planeten zu verstehen, Simulation des menschlichen Gehirns; siehe z. B. <https://www.humanbrainproject.eu>)
- ◊ Wissenschaftliches Rechnen und Numerische Simulation können extreme Anforderungen an die Leistungsfähigkeit von Computern stellen; siehe z. B. <http://www.deep-project.eu>
- ◊ Entwicklung von effizienten Algorithmen für Supercomputer: Grenzgebiet zwischen Numerik und Informatik

Beispiel 3 (Berechnung von Schaltungsnetzwerken).

- ▶ Szenario: Entwurf eines elektrischen Gerätes
- ▶ **Frage:** Wie müssen die Bauteile in der elektronischen Schaltung dimensioniert werden?
- ▶ konkretes Beispiel: einfaches Widerstandsnetzwerk (s.u.)

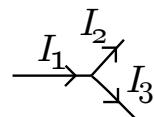


- ▶ einige Vereinfachungen:
 - ▶ statische Betrachtung
 - ▶ nur **ohmsche Widerstände**
- ▶ relevante Größen: Strom I , Spannung U , Leistung $P = UI$ (wichtig zur Dimensionierung der Bauteile)
- ▶ Ziel der Berechnungen: Leistung P_i in jedem Widerstand R_i
- ▶ ohmscher Widerstand: konstantes Verhältnis von U zu I :

$$U/I = R \quad (1.1)$$

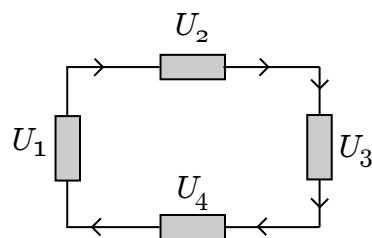
- ▶ physikalisches Prinzip der **Ladungserhaltung**: In einem "Knoten" (Zusammentreffen mehrerer Leitungen in einer Schaltung) ist die (gerichtete) Summe der Ströme 0 ("Knotenregel"):

$$\sum_{i=1}^k I_i = 0 \quad (1.2)$$

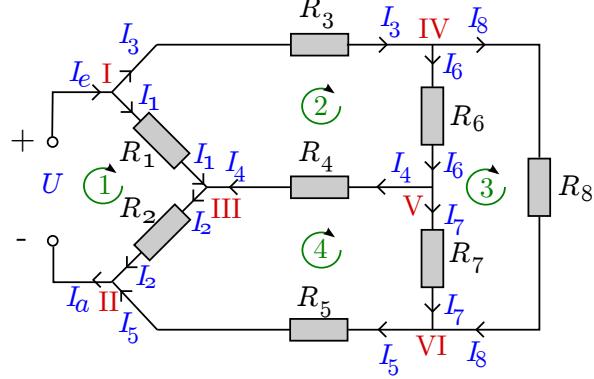


- ▶ Energieerhaltung: In einer "Masche" (geschlossener Leitungsverlauf in einer Schaltung) ist die gerichtete Summe der Spannungen 0 ("Maschenregel"):

$$\sum_{j=1}^l U_j = 0 \quad (1.3)$$



- wichtig auch hier: die per Konvention zuvor festgelegte Stromrichtung bestimmt das Vorzeichen der Spannungen in der Summe
- (1.2) und (1.3): **Kirchhoffsche Regeln** (nach Gustav Robert Kirchhoff, 1824-1887)
- Bezeichnung der Ströme und Spannungen im Widerstandsnetzwerk



- Anwendung der Knotenregel (1.2) auf das Widerstandnetzwerk oben

$$I : \quad I_e - I_1 - I_3 = 0$$

$$II : \quad -I_a + I_2 + I_5 = 0$$

$$III : \quad I_1 - I_2 + I_4 = 0$$

$$IV : \quad I_3 - I_6 - I_8 = 0$$

$$V : \quad -I_4 + I_6 - I_7 = 0$$

$$VI : \quad -I_5 + I_7 + I_8 = 0$$

- Anwendung der Maschenregel (1.3) und des Ohmschen Gesetzes (1.1):

$$1 : \quad U + R_1 I_1 + R_2 I_2 = 0$$

$$2 : \quad -R_1 I_1 + R_3 I_3 + R_4 I_4 + R_6 I_6 = 0$$

$$3 : \quad -R_6 I_6 - R_7 I_7 + R_8 I_8 = 0$$

$$4 : \quad -R_2 I_2 - R_4 I_4 + R_5 I_5 + R_7 I_7 = 0$$

- Ergebnis: lineares Gleichungssystem (kurz: LGS) in I_i

$$\begin{pmatrix} 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & R_1 & R_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -R_1 & 0 & R_3 & R_4 & 0 & R_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_6 & -R_7 & R_8 \\ 0 & 0 & 0 & -R_2 & 0 & -R_4 & R_5 & 0 & R_7 & 0 \end{pmatrix} \cdot \begin{pmatrix} I_e \\ I_a \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \\ I_8 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ U \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (1.4)$$

- bis hier: keine Numerik, sondern Physik und mathematische Modellbildung

- Die Lösung von (1.4) ist endlich berechenbar
- aber: Handrechnung aufwändig \Rightarrow Berechnungen mithilfe eines Computers
- Effiziente Algorithmen zum Lösen von LGS? Einfluss der Rechner(un)genauigkeiten auf das Ergebnis? \Rightarrow Numerik!
- konkrete Werte der Widerstände (in Ω):

i	1	2	3	4	5	6	7	8
R_i	20	50	100	75	150	200	10	100

- aus I_i : Leistung $P_i = U_i I_i = R_i I_i^2$ am Widerstand R_i
- Ergebnisse für $U = 10V$ (Leistungswerte auf 3 geltenden Ziffern gerundet):

i	I_i [A]	P_i [W]
e	0.1774	-
a	0.1774	-
1	0.0251	0.0126
2	0.1523	1.160
3	0.1391	1.935
4	-0.0133	0.0133
5	0.0077	0.00889
6	0.0209	0.0874
7	0.0383	0.0147
8	0.0174	0.0303

- erforderliche Baugröße der Widerstände kann jetzt bestimmt werden

Beispiel 4 (Strukturmechanik).

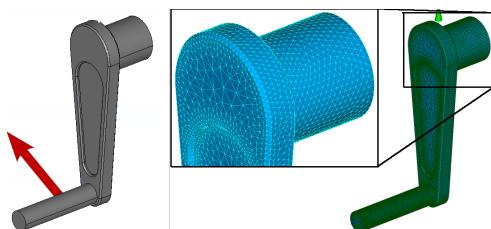
- Szenario: Auf ein Bauteil in einer Konstruktion wirken bestimmte Kräfte ein
- **Fragen:** Hält das Bauteil? Wie stark verformt es sich?
- Standardsituation im Maschinenbau
- man geht im Prinzip so vor wie beim Beispiel zuvor:
- mathematische Modellbildung: Das Zusammenspiel der für die Fragestellung relevanten physikalischen Größen wird in Form von Gleichungen beschrieben
 - relevante Größen:
 - Verschiebung $u(x, t)$: beschreibt die Formänderung des Bauteils aufgrund der angreifenden Kräfte; Funktion im Ort x und Zeit t
 - Spannungszustand $T(x, t)$: Maß für die lokale Beanspruchung des Materials; matrixwertige Funktion x und t
 - Modellbildung: Zusammenspiel zwischen Physik, Mathematik, Maschinenbau, Materialwissenschaften, ...
 - Modellbildung ist ungleich komplizierter als beim ersten Beispiel und wird deswegen hier nicht betrachtet

- am Ende: System von partiellen Differentialgleichungen (Lamé-Naviersche Differentialgleichungen)

$$\begin{aligned}
 E &= \frac{1}{2}(\nabla u + \nabla u^T) \text{ in } \Omega \\
 T &= 2\mu E + \lambda \operatorname{tr}(E) \cdot I \\
 -\operatorname{div} T &= f \\
 u &= 0 \text{ auf } \Gamma_0 \subseteq \partial\Omega \\
 T \cdot \mathbf{n} &= g \text{ auf } \partial\Omega \setminus \Gamma_0
 \end{aligned}$$

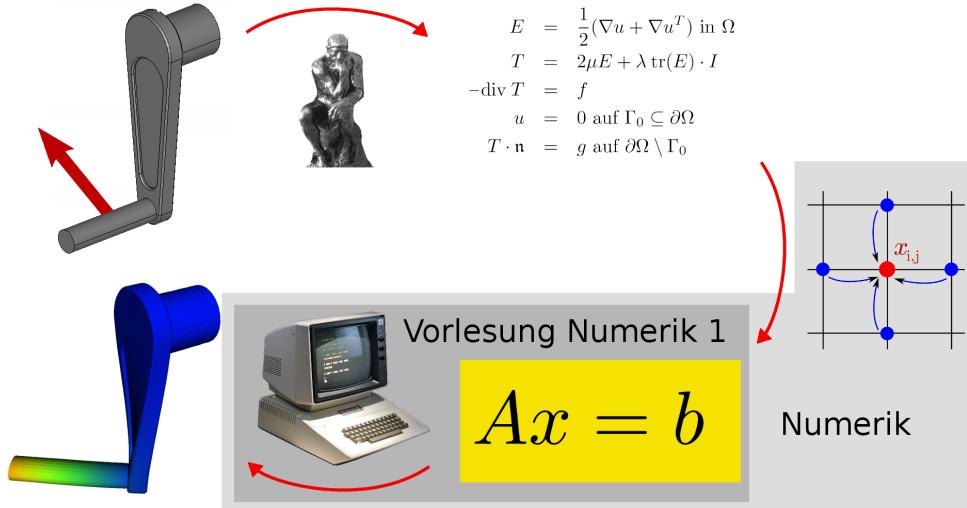
- Unter nicht sehr restriktiven Annahmen existiert genau eine Lösung der Lamé-Naviersches Differentialgleichungen (Ergebnis aus der mathematischen Analyse)

- **Problem:** Diese Lösung ist bis auf wenige Spezialfälle nicht berechenbar
- Aufgabe: Man finde ein **Ersatzproblem** ⇒ Numerik!
- Standardverfahren in der Strukturmechanik: Finite Elemente
- theoretische Hintergründe hier nicht zugänglich
- kurzer Abriss der Vorgehensweise:
 - Bauteil liegt als CAD-Datensatz vor (CAD: Computer Aided Design)
 - Bauteil wird "vernetzt", d. h. das Volumen wird mit Würfeln, Quadern, Tetraedern o. ä. ausgefüllt: "Elemente"
 - man erhält ein lineares Gleichungssystem, dessen Lösung eine Approximation an die gesuchte Verschiebung $u(x, t)$ repräsentiert
 - Anzahl der Gleichungen \sim Anzahl der Gitterpunkte
 - Faustregel: Je feiner die Vernetzung, umso besser die Approximation an $u(x, t)$, aber umso größer das LGS
- konkretes Beispiel: Kurbel (siehe unten)
 - Bauteil an Welle fest eingespannt
 - auf den Griff wirkt eine Kraft (roter Pfeil)
- hier: Vernetzung mit Tetraedern



- man erhält für die gezeigte nicht allzu feine Vernetzung ein LGS mit **163.233** Gleichungen und Unbekannten
- bei komplizierteren Bauteilen/komplexeren Simulationen: LGS mit $\mathcal{O}(10^6) - \mathcal{O}(10^8)$ Unbekannten durchaus üblich
- im Prinzip alle endlich berechenbar
- in der Praxis: Berechnung mit dem Computer einzige Möglichkeit der Lösung

- hier viel wichtiger als beim ersten Beispiel: effiziente Algorithmen zum Lösen von LGS
- am Ende: Aufbereitung der gewonnene Daten, um aus den Simulationsergebnisse die richtigen Schlüsse ziehen zu können \Rightarrow Visualisierung / Informatik
- Schema der numerischen Simulation:



The next big thing: automatische Optimierung

- bisher: Konstruktion gegeben, numerische Simulation erlaubt virtuelle Tests
- besser: Man lasse den Computer die optimale Konstruktion finden
- dazu:
 - man hält gewisse Abmessungen im CAD-Modell variabel ("Parametrisierung der Geometrie")
 - man finde iterativ den optimalen Parametersatz und damit die optimale Geometrie
 - pro Iteration: eine numerische Simulation

Bemerkung 5.

- ◊ beide Beispiele zeigen: effizientes Lösen von LGS in der Praxis wichtig
- ◊ Konsequenz: numerische lineare Algebra ist ein Schwerpunkt der Vorlesung (Kapitel 2, 4, 5)
- ◊ Einsatz von Computern impliziert Rundungsfehler
- ◊ Konsequenz: Studium von Rundungsfehlern und Fehlerfortpflanzung (Kapitel 3)
- ◊ Voraussetzung für Numerische Simulation: CAD-Darstellung von Bauteilen
- ◊ Konsequenz: effizientes Repräsentieren von Kurven im Computer, Interpolation (Kapitel 8)
- ◊ The next big thing: Optimierung (Kapitel 11)
- ◊ Numerik 1: Studium wesentlicher numerischer Komponenten für praxisrelevante Berechnungen und Simulationen

Zusammenfassung.

- *viele Objekte der Mathematik nicht endlich berechenbar*
- *Aufgabenstellungen müssen oft durch vereinfachte Ersatzprobleme approximiert werden*
- *mathematisch äquivalente Algorithmen können in Floating-Point-Arithmetik unterschiedliche Ergebnisse liefern*
- *ein großes Anwendungsgebiet der numerischen Mathematik: Numerische Simulation / wissenschaftliches Rechnen*

Kapitel 2

Lineare Gleichungssysteme – Direkte Löser I

2.1 Einleitung

- betrachte mehrere Gleichungen für eine bestimmte Anzahl von Unbekannten
- Unbekannte dürfen nur linear auftreten, wie in den folgenden Beispielen:

$$(a) \ 5x = 7$$

$$(b) \ x + 2y = 5$$

$$(c) \ 5x + 3y = 8$$

$$(d) \ 5x = 7$$

$$2x + y = 4$$

$$10x = 3$$

- das System

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

$$x + 2y = 3$$

ist nichtlinear

- betrachten wir allgemein ein System von m Gleichungen für n Unbekannte, so erhalten wir:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots &\quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

mit

$a_{ij} \in \mathbb{R}$ gegebene Koeffizienten,

$b_i \in \mathbb{R}$ gegebene rechte Seite,

$x_j \in \mathbb{R}$ gesuchte Unbekannte

- zur Abkürzung benutzt man die Matrix-Vektor-Schreibweise

$$Ax = b$$

mit Systemmatrix

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m \times n},$$

rechter Seite

$$b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{R}^m,$$

Vektor der Unbekannten

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

und Matrix-Vektor Produkt

$$Ax = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + \dots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + \dots + a_{mn}x_n \end{pmatrix}$$

- lineare Algebra:
 - ist $m \neq n$ (# Gleichungen \neq # Unbekannte) dann gibt es in der Regel keine *eindeutige* Lösung sondern entweder keine Lösung oder unendlich viele
 - $m = n \Rightarrow Ax = b$ eindeutig lösbar falls A regulär ist

Definition 6. $A \in \mathbb{R}^{n \times n}$ heißt *regulär*, falls $\det(A) \neq 0$, *singulär*, falls $\det(A) = 0$.

- alternative Kriterien sind:

- $\det(A) \neq 0 \Leftrightarrow$ Spalten $s_j = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{nj} \end{pmatrix}, j = 1, \dots, n$ linear unabhängig

- $\det(A) \neq 0 \Leftrightarrow$ Zeilen $z_i = (a_{i1}, \dots, a_{in}), i = 1, \dots, n$ linear unabhängig
- die Anzahl unabhängiger Spalten und unabhängiger Zeilen ist gleich und wird als Rang der Matrix $\text{rang}(A)$ bezeichnet
- $A \in \mathbb{R}^{n \times n}$ regulär $\Leftrightarrow \text{rang}(A) = n$, $A \in \mathbb{R}^{n \times n}$ singulär $\Leftrightarrow \text{rang}(A) < n$
- eine Matrix $A \in \mathbb{R}^{m \times n}$ definiert eine lineare Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^m$ durch $x \mapsto Ax$
- jede lineare Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^m$ ist als Matrix darstellbar
- $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{p \times m}$ nacheinander ausgeführt ergibt eine neue lineare Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^p$:

$$\mathbb{R}^n \ni x \mapsto Ax \mapsto B(Ax) \in \mathbb{R}^p$$

- die Matrix C der neuen Abbildung ergibt sich durch Matrizenmultiplikation:

$$B = \begin{pmatrix} b_{11} & \dots & b_{1m} \\ \vdots & & \vdots \\ b_{p1} & \dots & b_{pm} \end{pmatrix} \in \mathbb{R}^{p \times m}, \quad A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m \times n},$$

$$BA = \begin{pmatrix} (b_{11}a_{11} + \dots + b_{1m}a_{m1}) & \dots & (b_{11}a_{1n} + \dots + b_{1m}a_{mn}) \\ \vdots & & \vdots \\ (b_{p1}a_{11} + \dots + b_{pm}a_{m1}) & \dots & (b_{p1}a_{1n} + \dots + b_{pm}a_{mn}) \end{pmatrix} \in \mathbb{R}^{p \times n}$$

- Matrizenmultiplikation ist nur bei passender Dimension möglich
- in der Regel ist $AB \neq BA$ (auch bei $m = n$)

- die zu $A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m \times n}$ **transponierte Matrix** ist

$$A^T = \begin{pmatrix} a_{11} & \dots & a_{m1} \\ \vdots & & \vdots \\ a_{1n} & \dots & a_{mn} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

- A^T entsteht durch "Vertauschen von Zeilen mit Spalten", d.h. "Spiegelung" an der Diagonalen
- $(AB)^T = B^T A^T$
- ist $m = n$, so nennt man $A \in \mathbb{R}^{n \times n}$ quadratisch
- $A, B \in \mathbb{R}^{n \times n} \Rightarrow \det(AB) = \det(A)\det(B) \Rightarrow$ sind A, B regulär, dann ist auch AB regulär
- ist $A \in \mathbb{R}^{n \times n}$ regulär, so existiert die Inverse $A^{-1} \in \mathbb{R}^{n \times n}$ mit

$$A^{-1}A = AA^{-1} = I = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & & 1 \end{pmatrix}$$

- für A, B regulär ist $(AB)^{-1} = B^{-1}A^{-1}$
- auf \mathbb{R}^n verwenden wir das euklidische **Skalarprodukt**

$$\langle x, y \rangle = \sum_{i=1}^n x_i y_i, \quad x, y \in \mathbb{R}^n$$

- betrachten wir $x, y \in \mathbb{R}^n$ als Matrizen

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^{n \times 1}, \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

dann ist $x^T = (x_1, \dots, x_n) \in \mathbb{R}^{1 \times n}$ und

$$x^T y = (x_1, \dots, x_n) \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = x_1 y_1 + \dots + x_n y_n = \langle x, y \rangle \in \mathbb{R}^{1 \times 1} = \mathbb{R}$$

- wir betrachten in folgendem Kapitel nur Systeme mit regulärem A , d.h. quadratische Systeme, die eindeutig lösbar sind
 - um den “Betrag” eines Vektors $x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ zu beschreiben benutzt man gewöhnlich die **euklidische Länge**
- $$\sqrt{x_1^2 + \dots + x_n^2}$$
- dies ist ein Spezialfall einer Vektornorm

Definition 7. $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ heißt Norm auf \mathbb{R}^n falls

- (1) $\|x\| = 0 \Leftrightarrow x = 0$
- (2) $\|\alpha x\| = |\alpha| \|x\| \quad \forall x \in \mathbb{R}^n \quad \forall \alpha \in \mathbb{R}$
- (3) $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in \mathbb{R}^n$

Bemerkung 8. aus (1)–(3) folgt $\|x\| > 0 \quad \forall x \neq 0$

- häufig benutzte Normen sind:
 - $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$, $1 \leq p < \infty$, zum Beispiel

$$p = 1 \quad : \quad \|x\|_1 = |x_1| + \dots + |x_n|$$

$$p = 2 \quad : \quad \|x\|_2 = \sqrt{x_1^2 + \dots + x_n^2}$$
 - $\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$
- die verschiedenen Normen auf \mathbb{R}^n sind im folgenden Sinn äquivalent:
 - sind $\|\cdot\|, \|\cdot\|'$ zwei Normen auf \mathbb{R}^n dann gibt es Konstanten α, β unabhängig von x mit
$$\alpha \|x\|' \leq \|x\| \leq \beta \|x\|' \quad \forall x \in \mathbb{R}^n$$
- wir verwenden Normen um Fehler zu messen:
 - betrachte $Ax = b, Ax' = b'$
 - vergleiche $\|b - b'\|$ mit $\|x - x'\|$
- benutze Norm, die einfach zu handhaben ist
- Norm beschreibt “Größe” von Vektoren
- “Größe” von Matrizen:
 - $A \in \mathbb{R}^{n \times n}$ definiert lineare Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^n$ durch $x \mapsto Ax$
 - “Größe” von A :
 - wie stark wird ein Vektor maximal verlängert?
 - vergleiche $\|Ax\|$ mit $\|x\|$

Satz 9. Ist $\|\cdot\|$ eine Norm auf \mathbb{R}^n , $A \in \mathbb{R}^{n \times n}$, dann definiert

$$\|A\| := \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

eine **Matrixnorm** auf $\mathbb{R}^{n \times n}$

Bemerkung 10.

- ◊ die Matrixnorm erfüllt analog zur Vektornorm die Bedingungen (1)–(3)
- ◊ die Matrixnorm hängt von der benutzten Vektornorm ab und heißt deshalb die von der Vektornorm **induzierte Matrixnorm**
- ◊ Vektornorm und induzierte Matrixnorm sind kompatibel, d.h.

$$\begin{array}{ccc} \|Ax\| & \leq & \|A\| \\ \text{Vektor} & & \text{Matrix} \end{array} \quad \begin{array}{c} \|x\| \\ \text{Vektor} \end{array}$$

- ◊ für induzierte Matrixnormen gilt

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

- ◊ auf $\mathbb{R}^{n \times n}$ sind alle Matrixnormen äquivalent

Beispiel 11.

- für $\|\cdot\|_2$ erhält man als induzierte Matrixnorm die **Spektralnorm**

$$\|A\|_2 = \sqrt{\rho(A^T A)},$$

wobei der **Spektralradius** $\rho(B)$ der betragsgrößte Eigenwert von B ist

- für $\|\cdot\|_\infty$ erhält man die **Zeilensummennorm**

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

- für $\|\cdot\|_1$ erhält man die **Spaltensummennorm**

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

- die **Frobeniusnorm**

$$\|A\|_F = \sqrt{\text{spur}(A^T A)}, \quad \text{spur}(C) = \sum_{i=1}^n c_{ii}$$

ist *nicht* induziert, erfüllt aber

$$\|AB\|_F \leq \|A\|_F \cdot \|B\|_F$$

und ist wegen

$$\|Ax\|_2 \leq \|A\|_F \cdot \|x\|_2 \quad \forall x$$

mit der Vektornorm $\|\cdot\|_2$ verträglich; $\|A\|_F$ ist eine obere Schranke für $\|A\|_2$, aber deutlich einfacher zu berechnen

2.2 Gaußelimination

- erstes systematisches Verfahren für lineare Gleichungssysteme
- von Gauß in seinen astronomischen Arbeiten benutzt
- war vorher schon in China und im arabischen Kulturkreis bekannt
- wir betrachten das folgende Beispiel:

$$x_1 + x_2 + 3x_3 = 6 \quad (2.1a)$$

$$3x_1 + 6x_2 + 10x_3 = 25 \quad (2.1b)$$

$$x_1 + 4x_2 + 6x_3 = 15 \quad (2.1c)$$

- versuche durch Skalieren/Subtrahieren die Gleichungen zu vereinfachen, d.h. Unbekannte zu eliminieren:

$$x_1 + x_2 + 3x_3 = 6 \quad (2.2a)$$

$$(2.1b) - 3 \cdot (2.1a) \quad 3x_2 + x_3 = 7 \quad (2.2b)$$

$$(2.1c) - (2.1a) \quad 3x_2 + 3x_3 = 9 \quad (2.2c)$$

- eliminieren wir durch (2.2c) – (2.2b) aus (2.2c) x_2 , so erhalten wir

$$x_1 + x_2 + 3x_3 = 6 \quad (2.3a)$$

$$3x_2 + x_3 = 7 \quad (2.3b)$$

$$2x_3 = 2 \quad (2.3c)$$

- Gleichung (2.3c) enthält nur eine Unbekannte und ist direkt lösbar:

$$x_3 = 1$$

- setzt man x_3 in Gleichung (2.3b) ein so erhalten wir

$$3x_2 + 1 = 7$$

und damit $x_2 = 2$

- analog folgt für die erste Gleichung $x_1 + 2 + 3 = 6$ und

$$x_1 = 1$$

- damit ist $x = (1, 2, 1)^T$ Lösung des Systems
- wie sehen die Umformungen in Matrixschreibweise aus?
- wir fassen dazu A, b im erweiterten System $(A|b)$ zusammen:

$$\left(\begin{array}{ccc|c} 1 & 1 & 3 & 6 \\ 3 & 6 & 10 & 25 \\ 1 & 4 & 6 & 15 \end{array} \right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 3 & 6 \\ 0 & 3 & 1 & 7 \\ 0 & 3 & 3 & 9 \end{array} \right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 3 & 6 \\ 0 & 3 & 1 & 7 \\ 0 & 0 & 2 & 2 \end{array} \right)$$

- *eliminiere* durch Skalierung/Subtraktion von Zeilen die *Matrixspalten* unterhalb der Diagonalen (schrittweises Entfernen von Unbekannten aus Gleichungen)
- erhalte **Dreiecksgestalt**, d.h. letzte Gleichung enthält eine Unbekannte, vorletzte 2, etc.
- aus Dreiecksgestalt kann wie oben durch **rückwärts einsetzen** die Lösung bestimmt werden

- zusammengefasst erhalten wir folgendes Schema:

erweitertes System $\xrightarrow{\text{Spaltenelimination}}$ Dreiecksgestalt $\xrightarrow{\text{Rückwärtseinsetzen}}$ Lösung

- funktioniert dieses Verfahren für alle regulären Systeme mit $A \in \mathbb{R}^{n \times n}$?
- betrachten wir als Beispiel das folgende erweiterte System

$$(A|b) = \left(\begin{array}{ccc|c} 1 & 2 & 3 & 8 \\ 3 & 6 & 10 & 25 \\ 1 & 4 & 6 & 15 \end{array} \right)$$

- für A gilt $\det(A) = -2$, d.h. $Ax = b$ ist immer eindeutig lösbar
- wenden wir den Eliminationsprozess an so erhalten wir nach dem ersten Schritt

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 8 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 3 & 7 \end{array} \right)$$

- der zweite Schritt ist so nicht direkt durchführbar, da das zweite Diagonalelement verschwindet
- deshalb tauschen wir die Reihenfolge der zweiten und dritten Gleichung (d.h. der entsprechenden Zeilen in $(A|b)$) und erhalten

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 8 \\ 0 & 2 & 3 & 7 \\ 0 & 0 & 1 & 1 \end{array} \right)$$

wobei die Lösung wieder mit Rückwärtseinsetzen berechenbar ist

- es gilt also:
 - bei 0 auf der Diagonalen kann zunächst nicht eliminiert werden
 - die aktuelle Zeile muss mit einer darunter liegenden Zeile vertauscht werden, so dass anschließend das Diagonalelement ungleich 0 ist
- wir nehmen zunächst an, dass alle Diagonalelemente $\neq 0$ sind
- allgemeiner Fall kommt später
- dort sehen wir dann auch, dass für A regulär immer ein geeigneter Zeilentausch möglich ist
- Verallgemeinerung auf $n \times n$ Systeme:
 - sei A, b gegeben, d.h.

$$\left(\begin{array}{ccc|c} a_{11} & \dots & a_{1n} & b_1 \\ a_{21} & \dots & a_{2n} & b_2 \\ \vdots & & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} & b_n \end{array} \right)$$

- ist $a_{11} \neq 0$ so eliminiert man die erste Spalte durch

$$\begin{array}{rcl} \text{Zeile } 2 & - & \frac{a_{21}}{a_{11}} \text{ Zeile } 1 \\ \vdots & & \vdots \\ \text{Zeile } n & - & \frac{a_{n1}}{a_{11}} \text{ Zeile } 1 \end{array}$$

so dass als neues System

$$\left(\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} & b_n^{(1)} \end{array} \right)$$

entsteht mit unveränderter erster Zeile

$$a_{1j}^{(1)} = a_{1j}, \quad j = 1, \dots, n, \quad b_1^{(1)} = b_1,$$

und

$$a_{ij}^{(1)} = a_{ij} - l_{i1}a_{1j}, \quad b_i^{(1)} = b_i - l_{i1}b_1, \quad i = 2, \dots, n, \quad j = 1, \dots, n,$$

d.h. die mit

$$l_{i1} = \frac{a_{i1}}{a_{11}}$$

skalierte erste Zeile wird von der i -ten Zeile subtrahiert

- damit haben wir $Ax = b$ in ein äquivalentes System $A^{(1)}x = b^{(1)}$ transformiert mit erster Spalte

$$s_1^{(1)} = \begin{pmatrix} a_{11}^{(1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- ist $a_{22}^{(1)} \neq 0$ kann analog die zweite Spalte eliminiert werden, etc.
- insgesamt erhalten wir nach $n - 1$ Schritten

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

mit

$$A^{(n-1)} = U = \begin{pmatrix} u_{11} & \dots & \dots & u_{1n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & u_{nn} \end{pmatrix}$$

- statt $Ax = b$ lösen wir dann $Ux = b^{(n-1)}$ durch Rückwärtseinsetzen:

$$\begin{aligned} u_{11}x_1 + \dots + u_{1n-1}x_{n-1} + u_{1n}x_n &= b_1^{(n-1)} \\ &\quad \ddots \quad \vdots \\ u_{n-1n-1}x_{n-1} + u_{n-1n}x_n &= b_{n-1}^{(n-1)} \\ u_{nn}x_n &= b_n^{(n-1)} \end{aligned}$$

$$\Rightarrow \begin{aligned} x_n &= \frac{b_n^{(n-1)}}{u_{nn}} \\ x_{n-1} &= \frac{b_{n-1}^{(n-1)} - u_{n-1n}x_n}{u_{n-1n-1}} \\ x_{n-2} &= \frac{b_{n-2}^{(n-1)} - u_{n-2n}x_n - u_{n-2n-1}x_{n-1}}{u_{n-2n-2}} \\ &\vdots \end{aligned}$$

Zusammenfassung.

- eliminiere nacheinander Spalten, bis Dreiecksgestalt erreicht ist
- eventuell Problem bei Diagonale = 0 ⇒ Zeilentausch
- Lösung des Dreieckssystems durch Rückwärtseinsetzen

2.3 LU-Zerlegung

- untersuchen den Eliminationsprozess von oben

$$\begin{aligned} A &\rightarrow A^{(1)} \rightarrow \dots \rightarrow A^{(n-1)} = U \\ b &\rightarrow b^{(1)} \rightarrow \dots \rightarrow b^{(n-1)} \end{aligned}$$

und beschreiben ihn mathematisch präziser

- betrachten dann $Ax = b$, $Ax' = b'$ für identisches A :
 - Elimination liefert in beiden Fällen gleiches U
 - wie kann man sich doppelte Arbeit sparen?
- nehmen nach wie vor an, dass A regulär ist und wir keine Zeilen tauschen müssen

- sei zunächst $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $a_{11} \neq 0$:

- eliminiere a_{21} , indem von der zweiten Zeile $\frac{a_{21}}{a_{11}}$ mal Zeile 1 abgezogen wird
- das kann als Matrixmultiplikation geschrieben werden
- mit

$$F = \begin{pmatrix} 1 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -l_{21} & 1 \end{pmatrix}$$

folgt

$$\begin{aligned} FA &= \begin{pmatrix} 1 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \\ &= \begin{pmatrix} a_{11} & a_{12} \\ -\frac{a_{21}}{a_{11}}a_{11} + a_{21} & -\frac{a_{21}}{a_{11}}a_{12} + a_{22} \end{pmatrix} \\ &= \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} - \frac{a_{21}}{a_{11}}a_{12} \end{pmatrix} \end{aligned}$$

- analog folgt für die Elimination der ersten Spalte $A \rightarrow A^{(1)}$ im allgemeinen Fall

$$A^{(1)} = F_1 A, \quad F_1 = \begin{pmatrix} 1 & & & \\ -l_{21} & 1 & & \\ -l_{31} & & 1 & \\ \vdots & & & \ddots \\ -l_{n1} & & & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

bzw. für die Elimination der k -ten Spalte $A^{(k-1)} \rightarrow A^{(k)}$

$$A^{(k)} = F_k A^{(k-1)}, \quad F_k = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & -l_{k+1k} & 1 \\ & & \vdots & & \ddots \\ & & -l_{nk} & & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}, \quad (2.4)$$

$$\text{mit } l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$$

- die F_k heißen **Frobeniusmatrizen**
- mit den F_k erhalten wir

$$U = A^{(n-1)} = F_{n-1} \cdot \dots \cdot F_1 A$$

und

$$y = b^{(n-1)} = F_{n-1} \cdot \dots \cdot F_1 b$$

da auf b dieselben Zeilenoperationen angewandt werden

- alle F_k sind regulär ($\det F_k = 1$) so dass $Ax = b$ und $Ux = y$ wirklich äquivalent sind
- aus

$$U = F_{n-1} \cdot \dots \cdot F_1 A$$

folgt andererseits

$$A = F_1^{-1} \cdot \dots \cdot F_{n-1}^{-1} U$$

- für F_k rechnet man leicht nach, dass

$$F_k^{-1} = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & l_{k+1k} & 1 \\ & & \vdots & & \ddots \\ & & l_{nk} & & 1 \end{pmatrix} \quad (2.5)$$

- durch Ausmultiplizieren erhält man

$$L := F_1^{-1} \cdot \dots \cdot F_{n-1}^{-1} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \dots & l_{nn-1} & 1 \end{pmatrix}$$

und damit $A = LU$

Satz 12. Sind alle $a_{kk}^{(k-1)} \neq 0$ so erzeugt die Gaußelimination eine LU-Zerlegung

$$A = LU$$

von A , wobei L eine untere Dreiecksmatrix mit Diagonale 1 und U eine obere Dreiecksmatrix ist.

- hat man für A einmal eine Gaußelimination durchgeführt, so kennt man L, U
- $LUX = b$ ist sehr leicht zu lösen:
 - löse $Ly = b$, dann $Ux = y$
 - x ist die gesuchte Lösung, da $b = Ly = LUx = Ax$
 - $Ux = y$ kann durch Rückwärtseinsetzen gelöst werden, da U obere Dreiecksmatrix ist
 - L ist untere Dreiecksmatrix \Rightarrow löse $Ly = b$ durch **Vorwärtseinsetzen**:

$$Ly = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & l_{32} & 1 & \\ \vdots & \vdots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \end{pmatrix}$$

$$\begin{aligned} y_1 &= b_1 \\ \iff l_{21}y_1 + y_2 &= b_2 \quad \Leftrightarrow \quad y_2 = b_2 - l_{21}y_1 \\ &\vdots && \vdots \end{aligned}$$

- Aufwand (Anzahl der Floating-Point-Operationen):

- $Ax = b$ mit Gaußelimination $\sim \frac{2}{3}n^3$
- L, U bestimmen mit Gaußelimination $\sim \frac{2}{3}n^3$
- $LUX = b$ lösen $\sim 2n^2$

\Rightarrow ist L, U bekannt, so ist die Lösung für verschiedene b "billig" zu erhalten

- praktische Durchführung:

- Implementierung in 3 Routinen:
 - (1) LU-Zerlegung $A = LU$ mit Gaußelimination
 - (2) Lösen von $Ly = b$ mit Vorwärtseinsetzen
 - (3) Lösen von $Ux = y$ mit Rückwärtseinsetzen
- Algorithmen bestehen aus direkter Umsetzung der Formeln aus diesem Abschnitt
- die Speicherfrage (wohin mit L/U) ist noch offen
- starten mit Speicher für

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

- hören auf mit

$$L = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \dots & l_{n,n-1} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ u_{22} & \dots & u_{2n} \\ \ddots & & \vdots \\ u_{nn} \end{pmatrix}$$

- die 1 auf der Diagonalen von L braucht nicht explizit gespeichert zu werden
- der Rest von L, U "passt" in A rein
- A selbst brauchen wir dann nicht mehr, da $Ax = b \Leftrightarrow LUx = b$
- L, U können direkt (ohne Zwischenspeicherung) im Laufe der Zerlegung in A eingetragen werden

Beispiel 13.

► wir betrachten

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 4 & 0 \\ 2 & 7 & 1 \end{pmatrix}$$

und eliminieren die erste Spalte:

$$\begin{aligned} z_2 - l_{21}z_1, \quad l_{21} &= \frac{4}{2} = 2 \\ z_3 - l_{31}z_1, \quad l_{31} &= \frac{2}{2} = 1 \end{aligned} \Rightarrow F_1 A = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 2 & -2 \\ 0 & 6 & 0 \end{pmatrix}$$

- zwei Elemente wurden eliminiert, zwei neue Einträge für L berechnet (l_{21} eliminiert a_{21} , l_{31} eliminiert a_{31})
- trage l_{21}, l_{31} an der Position der eliminierten Elemente ein:

$$\left[\begin{array}{ccc|cc} 2 & 1 & 1 & & \\ 2 & 2 & -2 & & \\ 1 & 6 & 0 & & \end{array} \right]$$

- eliminieren wir analog die zweite Spalte durch $z_3 - l_{32}z_2$, $l_{32} = \frac{6}{2} = 3$ so erhalten wir

$$\begin{array}{ccc|c} & 2 & 1 & 1 \\ & 2 & 2 & -2 \\ 1 & 3 & | & 6 \end{array},$$

d.h.

$$L = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ 1 & 3 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 6 \end{pmatrix}$$

- dies gilt auch allgemein:
 - l_{ij} dient zum Eliminieren des Elements an Position i, j
 - l_{ij} wird an Stelle des eliminierten Elements gespeichert
- im nächsten Beispiel verwenden wir die LU-Zerlegung um den Fall mehrerer rechte Seiten zu behandeln

Beispiel 14.

- wir betrachten das letzte Beispiel

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 4 & 0 \\ 2 & 7 & 1 \end{pmatrix}$$

und wollen $Ax = b$ bzw. $Ax' = b'$ für

$$b = \begin{pmatrix} 7 \\ 12 \\ 19 \end{pmatrix}, \quad b' = \begin{pmatrix} 3 \\ 4 \\ 9 \end{pmatrix}$$

effizient lösen

- wir kennen $A = LU$ mit

$$L = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ 1 & 3 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 6 \end{pmatrix}$$

- wir lösen für b zunächst $Ly = b$, d.h.

$$\begin{pmatrix} 1 & & \\ 2 & 1 & \\ 1 & 3 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 12 \\ 19 \end{pmatrix}$$

durch Vorwärtseinsetzen

$$\begin{aligned} y_1 &= 7 \\ y_2 &= 12 - 2 \cdot y_1 = -2 \\ y_3 &= 19 - y_1 - 3 \cdot y_2 = 18 \end{aligned} \Rightarrow y = \begin{pmatrix} 7 \\ -2 \\ 18 \end{pmatrix}$$

und dann $Ux = y$, d.h.

$$\begin{pmatrix} 2 & 1 & 1 \\ & 2 & -2 \\ & & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ -2 \\ 18 \end{pmatrix}$$

$$\begin{aligned} x_1 &= \frac{7 - x_2 - x_3}{2} = 1 \\ \Leftrightarrow x_2 &= \frac{-2 + 2 \cdot x_3}{2} = 2 \quad \Leftrightarrow x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \\ x_3 &= 3 \end{aligned}$$

► für $b' = (3, 4, 9)^T$ folgt

$$Ly' = b' \quad \text{mit} \quad y' = \begin{pmatrix} 3 \\ -2 \\ 12 \end{pmatrix}, \quad Ux' = y' \quad \text{mit} \quad x' = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$$

Zusammenfassung.

- Gaußelimination zerlegt $A = LU$ unabhängig von rechter Seite von $Ax = b$
- $Ax = b \Leftrightarrow LUx = b \Leftrightarrow Ly = b, Ux = y$
- Aufwand für Zerlegung $A = LU \sim \frac{2}{3}n^3$
- Aufwand für Lösen von $LUx = b \sim 2n^2$
- L, U kann über A gespeichert werden
- für mehrere rechte Seiten wird einmal $A = LU$ bestimmt und dann wiederholt $LUx = b, LUx' = b'$

2.4 LU-Zerlegung für beliebige reguläre Matrizen

- bisher haben wir vorausgesetzt, dass während der Elimination kein Diagonalelement verschwinden darf, d.h. $a_{kk}^{(k-1)} \neq 0$
- das Beispiel oben hat gezeigt, dass es reguläre Matrizen A gibt, die dies nicht erfüllen
- als Abhilfe haben wir dort Zeilen getauscht
- das funktioniert immer

Lemma 15. Sei A regulär, $A^{(k-1)} = F_{k-1} \cdot \dots \cdot F_1 A$ mit Hilfe der Gaußelimination erzeugt und $a_{kk}^{(k-1)} = 0$. Dann gibt es in der k -ten Spalte unterhalb der Diagonalen immer mindestens ein Element $a_{ik}^{(k-1)} \neq 0, i > k$, d.h. ein erfolgreicher Zeilentausch ist immer möglich.

Beweis.

- Annahme: falsch
- sei k erster Index so dass $a_{ik}^{(k-1)} = 0, i \geq k$, d.h.

$$A^{(k-1)} = \begin{pmatrix} a_{11}^{(k-1)} & \dots & a_{1k-1}^{(k-1)} & a_{1k}^{(k-1)} & a_{1k+1}^{(k-1)} & \dots & a_{1n}^{(k-1)} \\ 0 & \ddots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & a_{k-1k-1}^{(k-1)} & a_{k-1k}^{(k-1)} & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & a_{nk+1}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{pmatrix}$$

- die k -te Spalte ist linear abhängig von den ersten $k-1$ (bzw. $\det A^{(k-1)} = 0$), weshalb $A^{(k-1)}$ singulär ist
- aber $A^{(k-1)} = F_{k-1} \cdot \dots \cdot F_1 A$, wobei alle F_i regulär sind, weshalb schon A singulär sein muss, was ein Widerspruch zur Voraussetzung ist

□

- wie baut man Zeilentauschen in LU-Matrixformalismus ein:

- bevor F_k die k -te Spalte eliminiert, muss eventuell getauscht werden
- Zeilentausch erfolgt durch Multiplikation mit einer **Permutationsmatrix**:

$$P = \begin{pmatrix} & & i & & j & & \\ & & \vdots & & \vdots & & \\ & & 1 & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & \dots & \dots & 0 & \dots & \dots & 1 & \dots & \dots & \\ & & & & \vdots & & & \vdots & & & \\ & & & & 1 & & & & \ddots & & \\ & & & & & 1 & & & & \ddots & \\ & & & & & & 1 & & & & \\ & & & & & & & 0 & \dots & \dots & \\ & & & & & & & & \vdots & & \\ & & & & & & & & & 1 & \\ & & & & & & & & & & \ddots \end{pmatrix} \quad (2.6)$$

- PA tauscht Zeilen i und j , AP tauscht Spalten i und j
- für Permutationen gilt $P = P^T$ und $PP = I$, d.h. $P = P^{-1}$

- damit erhalten wir

$$U = F_{n-1} P_{n-1} \cdot \dots \cdot F_1 P_1 A$$

wobei U eine obere Dreiecksmatrix ist und P_i Permutationsmatrizen oder $P_i = I$ (was auch eine Permutation ist)

- ohne Permutation war

$$\begin{aligned} U &= F_{n-1} \cdot \dots \cdot F_1 A &= L^{-1} A \\ L &= (F_{n-1} \cdot \dots \cdot F_1)^{-1} = F_1^{-1} \cdot \dots \cdot F_{n-1}^{-1} \end{aligned}$$

wobei L untere Dreiecksmatrix ist

- analoges Vorgehen mit Permutationen liefert

$$U = \tilde{L}^{-1} A, \quad \tilde{L} = (F_{n-1} P_{n-1} \cdot \dots \cdot F_1 P_1)^{-1}$$

- Problem:

- \tilde{L} ist in der Regel keine untere Dreiecksmatrix
- $\tilde{L}y = b$ ist nicht mit Vorwärtseinsetzen lösbar

- wie bekommt man Dreiecksgestalt zurück?

- benutze $P_i P_i = I \Rightarrow P_1 P_2 \cdot \dots \cdot P_{n-1} P_{n-1} \cdot \dots \cdot P_2 P_1 = I$

- damit erhalten wir:

$$\begin{aligned} U &= F_{n-1} P_{n-1} \cdot \dots \cdot F_2 P_2 F_1 P_1 A \\ &= (F_{n-1} P_{n-1} \cdot \dots \cdot F_2 P_2 F_1 P_1) (P_1 \cdot \dots \cdot P_{n-1} P_{n-1} \cdot \dots \cdot P_1) A \\ &= (F_{n-1} P_{n-1} \cdot \dots \cdot F_2 P_2 F_1 P_2 \cdot \dots \cdot P_{n-1}) (P_{n-1} \cdot \dots \cdot P_1) A, \end{aligned} \tag{2.7}$$

d.h. $U = L^{-1} PA$ bzw. $PA = LU$ mit Permutationsmatrix (nur Zeilentausch)

$$P = P_{n-1} \cdot \dots \cdot P_1 \tag{2.8}$$

und

$$\begin{aligned} L &= (F_{n-1} P_{n-1} \cdot \dots \cdot F_2 P_2 F_1 P_2 \cdot \dots \cdot P_{n-1})^{-1} \\ P_i^{-1} &= P_i \\ &= P_{n-1} \cdot \dots \cdot P_2 F_1^{-1} P_2 F_2^{-1} \cdot \dots \cdot P_{n-1} F_{n-1}^{-1} \end{aligned} \tag{2.9}$$

- L hat jetzt Dreiecksgestalt:

- setze

$$L_1 = F_1^{-1}, \quad L_k = P_k L_{k-1} P_k F_k^{-1}, \quad k = 2, \dots, n-1$$

d.h.

$$\begin{aligned} L_2 &= P_2 F_1^{-1} P_2 F_2^{-1} \\ L_3 &= P_3 L_2 P_3 F_3^{-1} = P_3 P_2 F_1^{-1} P_2 F_2^{-1} P_3 F_3^{-1} \\ &\vdots \\ L_{n-1} &= L \end{aligned}$$

- wir zeigen jetzt, dass alle L_k untere Dreiecksgestalt haben:

- $L_1 = F_1^{-1} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & & & 1 \end{pmatrix}$
- $L_2 = P_2 L_1 P_2 F_2^{-1}$:
 - in $P_2 L_1$ sind die Zeilen 2 und j vertauscht:

$$P_2 L_1 = \begin{pmatrix} 1 & & & & & & & \\ l_{j1} & 0 & & & & & & 1 \\ l_{31} & & 1 & & & & & \\ \vdots & & & \ddots & & & & \\ l_{j-11} & & & & 1 & & & \\ l_{21} & & 1 & & & 0 & & \\ l_{j+11} & & & & & & 1 & \\ \vdots & & & & & & & \ddots \\ l_{n1} & & & & & & & 1 \end{pmatrix}$$

- $(P_2 L_1) P_2$ tauscht Spalte 2 und j :

$$P_2 L_1 P_2 = \begin{pmatrix} 1 & & & & & & & \\ l_{j1} & 1 & & & & & & \\ l_{31} & & 1 & & & & & \\ \vdots & & & \ddots & & & & \\ l_{j-11} & & & & 1 & & & \\ l_{21} & & & & & 1 & & \\ l_{j+11} & & & & & & 1 & \\ \vdots & & & & & & & \ddots \\ l_{n1} & & & & & & & 1 \end{pmatrix}$$

- wegen

$$F_2^{-1} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & l_{32} & 1 & & & \\ & & \vdots & & \ddots & & \\ & & l_{n2} & & & 1 & \end{pmatrix}$$

erhalten wir schließlich

$$P_2 L_1 P_2 F_2^{-1} = \begin{pmatrix} 1 & & & & & & & \\ l_{j1} & 1 & & & & & & \\ l_{31} & l_{32} & 1 & & & & & \\ \vdots & \vdots & & \ddots & & & & \\ l_{j-11} & l_{j-12} & & & 1 & & & \\ l_{21} & l_{j2} & & & & 1 & & \\ l_{j+11} & l_{j+12} & & & & & 1 & \\ \vdots & \vdots & & & & & & \ddots \\ l_{n1} & l_{n2} & & & & & & 1 \end{pmatrix}$$

- allgemein gilt (per Induktion) für $L_k = P_k L_{k-1} P_k F_k^{-1}$:

- L_{k-1} ist gegeben durch

$$L_{k-1} = \begin{pmatrix} 1 & & & & \\ * & \ddots & & & \\ \vdots & \ddots & 1 & & \\ * & \dots & * & 1 & \\ \vdots & & \vdots & & \ddots \\ * & \dots & * & & 1 \end{pmatrix}_{k-1}$$

- $P_k L_{k-1} P_k$ tauscht in den ersten $k-1$ Spalten Zeile k mit Zeile $j > k$, d.h. im Block

$$\begin{matrix} * & & & \\ \vdots & \ddots & & \\ * & \dots & * & k \\ \vdots & & \vdots & \\ * & \dots & * & j > k \\ \vdots & & \vdots & \\ * & \dots & * & \end{matrix}$$

- Multiplikation mit F_k vergrößert diesen Block um die k -te Spalte von F_k^{-1} , d.h.

$$L_k = P_k L_{k-1} P_k F_k^{-1} = \begin{pmatrix} 1 & & & & \\ * & \ddots & & & \\ \vdots & \ddots & 1 & & \\ * & \dots & * & 1 & \\ \vdots & & \vdots & l_{k+1k} & 1 \\ \vdots & & \vdots & \vdots & \ddots \\ * & \dots & * & l_{nk} & & 1 \end{pmatrix}$$

- damit haben alle L_k und somit auch $L = L_{n-1}$ untere Dreiecksgestalt

Satz 16. Ist $A \in \mathbb{R}^{n \times n}$ regulär, so gibt es Permutationen $P_k, k = 1, \dots, n-1$ so dass mit $P = P_{n-1} \dots P_1$ gilt

$$PA = LU,$$

wobei L, U untere/obere Dreiecksmatrizen sind mit $\text{diag}(L) = 1$.

Bemerkung 17.

- ◊ tauscht man vorab die Zeilen in A entsprechend P , so sind alle $a_{kk}^{(k-1)} \neq 0$, d.h. $PA = LU$
- ◊ die Permutation P ist aber nicht vorher bekannt, sondern ergibt sich erst während der LU -Zerlegung

- Praktische Durchführung:

- $Ax = b \Leftrightarrow PAx = Pb \Leftrightarrow LUx = Pb$
- Implementierung in 3 Routinen:
 - (1) LU-Zerlegung mit Permutation durch Gaußelimination liefert L, U, P
 - (2) löse $Ly = Pb$ durch Vorwärtseinsetzen
 - (3) löse $Ux = y$ durch Rückwärtseinsetzen
- $P = P_{n-1} \cdots P_1$ wobei P_k die k -te Zeile mit einer Zeile $j_k > k$ vertauscht
- deshalb können alle P_k in einem *einzigem* ganzzahligen Vektor

$$p = (j_1, j_2, \dots, j_{n-1})^T$$

abgespeichert werden

- wie muss Zeilentausch bei Gaußelimination eingebaut werden?
- speichere wie oben L im freiwerdenden Teil von A
- im k -ten Schritt (Elimination von Spalte k) ergibt sich folgendes Bild:
 - bestimme P_k und berechne $P_k A^{(k-1)}$ d.h. tausche Zeile k von $A^{(k-1)}$ mit Zeile $j_k > k$
 - nach oben: tausche Zeile k mit Zeile j_k im wesentlichen Block von L_{k-1}
 - eliminiere Spalte k und erweitere L_{k-1} -Block um eine Spalte zu L_k
- insgesamt heißt das für einen Eliminationsschritt:
 - suche j_k
 - tausche *komplette* Zeile im 2D-Array, in dem ursprünglich A abgelegt ist
 - eliminiere k -te Spalte wie üblich

Beispiel 18.

► betrachte

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 2 & 1 & 2 & 0 \\ 4 & 4 & 0 & 0 \\ 2 & 3 & 1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 5 \\ 4 \\ 5 \end{pmatrix}, \quad b' = \begin{pmatrix} 4 \\ 10 \\ 12 \\ 11 \end{pmatrix}$$

► 1. Spalte: tausche z_1 mit $z_2 \rightarrow j_1 = 2$

$$\begin{array}{cccc} 2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 4 & 0 & 0 \\ 2 & 3 & 1 & 0 \end{array} \Rightarrow F_1 = \begin{pmatrix} 1 & & & \\ 0 & 1 & & \\ -2 & & 1 & \\ -1 & & & 1 \end{pmatrix} \Rightarrow \begin{array}{c|ccc} 2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 2 & -4 & 0 \\ 1 & 2 & -1 & 0 \end{array}$$

► 2. Spalte: tausche z_2 mit $z_3 \rightarrow j_2 = 3$

$$\begin{array}{cccc} 2 & 1 & 2 & 0 \\ 2 & 2 & -4 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & -1 & 0 \end{array} \Rightarrow F_2 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & 0 & 1 & \\ & -1 & & 1 \end{pmatrix} \Rightarrow \begin{array}{c|ccc} 2 & 1 & 2 & 0 \\ 2 & 2 & -4 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 3 & 0 \end{array}$$

► 3.Spalte: tausche z_3 mit $z_4 \rightarrow j_3 = 4$

$$\begin{array}{rccccc} \begin{array}{c|cc} 2 & 1 & 2 & 0 \\ \hline 2 & 2 & -4 & 0 \\ 1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{array} & \Rightarrow & F_3 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 0 & 1 \end{pmatrix} & \Rightarrow & \begin{array}{c|cc|c} 2 & 1 & 2 & 0 \\ \hline 2 & 2 & -4 & 0 \\ 1 & 1 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{array} \end{array}$$

► damit haben wir:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 2 & 0 \\ 0 & 2 & -4 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad p = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$$

► löse $Ax = b$ über $Ly = Pb$, $Ux = y$:

► $Pb : p = (2, 3, 4)^T$ bedeutet sukzessives Tauschen von Zeile 1 mit 2, 2 mit 3, 3 mit 4, also

$$b = \begin{pmatrix} 3 \\ 5 \\ 4 \\ 5 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 \\ 3 \\ 4 \\ 5 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 \\ 4 \\ 3 \\ 5 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 \\ 4 \\ 5 \\ 3 \end{pmatrix} = Pb$$

► $Ly = Pb$:

$$\begin{pmatrix} 1 & & & \\ 2 & 1 & & \\ 1 & 1 & 1 & \\ 0 & 0 & 0 & 1 \end{pmatrix} y = \begin{pmatrix} 5 \\ 4 \\ 5 \\ 3 \end{pmatrix} \Rightarrow \begin{array}{l} y_1 = 5 \\ y_2 = 4 - 2y_1 = -6 \\ y_3 = 5 - y_1 - y_2 = 6 \\ y_4 = 3 \end{array} \Rightarrow y = \begin{pmatrix} 5 \\ -6 \\ 6 \\ 3 \end{pmatrix}$$

► $Ux = y$:

$$\begin{pmatrix} 2 & 1 & 2 & 0 \\ 0 & 2 & -4 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} x = \begin{pmatrix} 5 \\ -6 \\ 6 \\ 3 \end{pmatrix} \Rightarrow \begin{array}{l} x_1 = \frac{5 - x_2 - 2x_3}{2} = 0 \\ x_2 = \frac{-6 + 4x_3}{2} = 1 \\ x_3 = 2 \\ x_4 = 3 \end{array} \Rightarrow x = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}$$

► analog folgt für $b' = (4, 10, 12, 11)^T$.

$$Pb' = \begin{pmatrix} 10 \\ 12 \\ 11 \\ 4 \end{pmatrix}, \quad y' = \begin{pmatrix} 10 \\ -8 \\ 9 \\ 4 \end{pmatrix}, \quad x' = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Zusammenfassung.

- jede reguläre Matrix $A \in \mathbb{R}^{n \times n}$ kann in $PA = LU$ zerlegt werden
- $Ax = b \Leftrightarrow Ly = Pb, Ux = y$
- P, L, U erhält man aus Gaußelimination mit Zeilentausch
- L, U werden über die alte Matrix gespeichert, P wird als ganzzahliger Vektor abgelegt
- Aufwand Elimination $\sim \frac{2}{3}n^3$, Aufwand Vorwärts/Rückwärtseinsetzen je $\sim n^2$
- Freiheiten beim Zeilentausch werden später noch ausgenutzt

2.5 Cholesky-Zerlegung

- haben bis jetzt beliebige reguläre $A \in \mathbb{R}^{n \times n}$ betrachtet
- in einem wesentlichen Anwendungsgebiet (Partielle Differentialgleichungen) sind häufig spezielle Gleichungssysteme zu lösen

Definition 19. $A \in \mathbb{R}^{n \times n}$ heißt symmetrisch positiv definit (**spd**) falls $A = A^T$ und

$$\langle x, Ax \rangle > 0 \quad \forall x \in \mathbb{R}^n, x \neq 0.$$

Satz 20. Ist $A \in \mathbb{R}^{n \times n}$ spd, so ist A regulär und $a_{ii} > 0, i = 1, \dots, n$.

Beweis.

- ist A nicht regulär, so existiert ein $x \neq 0$ mit $Ax = 0$ und damit

$$\langle x, Ax \rangle = \langle x, 0 \rangle = 0,$$

so dass A nicht spd sein kann

- ist $e_i = (0, \dots, 0, 1, 0, \dots, 0)^T$ der i -te Einheitsvektor so folgt

$$a_{ii} = e_i^T A e_i = \langle e_i, A e_i \rangle \stackrel{A \text{ spd}}{>} 0$$

□

- wir betrachten jetzt die Gaußelimination für A spd
- A ist symmetrisch, d.h.

$$A = \left(\begin{array}{c|cccc} a_{11} & a_{21} & \dots & a_{n1} \\ \hline a_{21} & & & & \\ \vdots & & & & B \\ a_{n1} & & & & \end{array} \right)$$

- A ist spd, weshalb $a_{11} > 0$ so dass kein Zeilentausch notwendig ist

- wir eliminieren die erste Spalte durch Multiplikation mit F_1 von links:

$$F_1 A = \left(\begin{array}{c|cccc} a_{11} & a_{21} & \dots & a_{n1} \\ \hline 0 & & & & \\ \vdots & & \tilde{B} & & \\ 0 & & & & \end{array} \right)$$

- multiplizieren wir mit F_1^T zusätzlich von rechts, so folgt aus der Symmetrie

$$\begin{aligned} A^{(1)} = F_1 A F_1^T &= \left(\begin{array}{c|cccc} a_{11} & a_{21} & \dots & a_{n1} \\ \hline 0 & & & & \\ \vdots & & \tilde{B} & & \\ 0 & & & & \end{array} \right) \left(\begin{array}{cccc} 1 & -l_{21} & \dots & -l_{n1} \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{array} \right) \\ &= \left(\begin{array}{c|cccc} a_{11} & 0 & \dots & 0 \\ \hline 0 & & & & \\ \vdots & & \tilde{B} & & \\ 0 & & & & \end{array} \right), \end{aligned}$$

wobei \tilde{B} nicht verändert wird

- $A^{(1)}$ ist symmetrisch, denn

$$(A^{(1)})^T = (F_1 A F_1^T)^T = F_1 A^T F_1^T \stackrel{A \text{ sym.}}{=} F_1 A F_1^T = A^{(1)}$$

- $A^{(1)}$ ist auch positiv definit:

- es gilt

$$\langle x, A^{(1)}x \rangle = x^T F_1 A F_1^T x = (F_1^T x)^T A (F_1^T x) = y^T A y = \langle y, A y \rangle, \quad y = F_1^T x$$

- da F_1^T regulär ist, folgt aus $x \neq 0$ auch $y = F_1^T x \neq 0$
- da A spd ist, erhalten wir schließlich

$$\langle x, A^{(1)}x \rangle = \langle y, A y \rangle > 0 \quad \forall x \neq 0$$

- damit kann in $A^{(1)}$ ganz analog die zweite Spalte/Zeile beseitigt werden:

$$A^{(2)} = F_2 A^{(1)} F_2^T = F_2 F_1 A F_1^T F_2^T = \left(\begin{array}{ccccc} * & & & & \\ & * & & & \\ & & * & \dots & * \\ & & \vdots & & \vdots \\ & & * & \dots & * \end{array} \right)$$

wobei $A^{(2)}$ wieder spd

- nach $n - 1$ Schritten folgt

$$F_{n-1} \cdot \dots \cdot F_1 A F_1^T \cdot \dots \cdot F_{n-1}^T = \begin{pmatrix} a_{11} & & & \\ & a_{22}^{(1)} & & \\ & & \ddots & \\ & & & a_{nn}^{(n-1)} \end{pmatrix} =: D$$

mit D spd, d.h. alle Diagonalelemente sind größer 0

- damit ist

$$A = F_1^{-1} \cdot \dots \cdot F_{n-1}^{-1} D (F_{n-1}^T)^{-1} \cdot \dots \cdot (F_1^T)^{-1}$$

- für alle $C \in \mathbb{R}^{n \times n}$ regulär gilt $(C^T)^{-1} = (C^{-1})^T$ so dass

$$(F_{n-1}^T)^{-1} \cdot \dots \cdot (F_1^T)^{-1} = (F_{n-1}^{-1})^T \cdot \dots \cdot (F_1^{-1})^T = (F_1^{-1} \cdot \dots \cdot F_{n-1}^{-1})^T = \tilde{L}^T,$$

also ist $A = \tilde{L} D \tilde{L}^T$, \tilde{L} untere Dreiecksmatrix mit 1 auf der Diagonalen

- da

$$D = \begin{pmatrix} d_1 & & & \\ & \ddots & & \\ & & & d_n \end{pmatrix}$$

mit $d_i > 0$ gilt

$$D = E E^T, \quad E = \begin{pmatrix} \sqrt{d_1} & & & \\ & \ddots & & \\ & & & \sqrt{d_n} \end{pmatrix}$$

und somit

$$A = \tilde{L} E E^T \tilde{L}^T = L L^T, \quad L = \tilde{L} E$$

Satz 21. $A \in \mathbb{R}^{n \times n}$ symmetrisch positiv definit, kann wie folgt zerlegt werden:

- (1) $A = \tilde{L} D \tilde{L}^T$, \tilde{L} untere Dreiecksmatrix mit 1 auf der Diagonalen, D Diagonalmatrix mit positiven Diagonalelementen (**rationale Cholesky-Zerlegung**)
- (2) $A = L L^T$, L untere Dreiecksmatrix (**Cholesky-Zerlegung**)

- Aufwand:

- die Cholesky-Zerlegung basiert auf der Gaußelimination mit zusätzlicher Multiplikation mit F_k^T , was zunächst einen höheren Aufwand erwarten lässt
- aber F_k^T eliminiert nur die entsprechende Zeile und hat sonst keinen Einfluss, d.h. es ist nichts weiter zu tun als 0 in die Matrix einzutragen
- alle $A^{(k)}$ sind symmetrisch (oberes und unteres Dreieck sind identisch), weshalb nur eine Hälfte der Koeffizienten berechnet werden muss
- Aufwand Cholesky $\sim \frac{1}{2}$ Aufwand Gauß $\sim \frac{1}{3}n^3$ Operationen

- für die praktische Durchführung wird in der Regel eine andere Form benutzt, die gar nicht an Gauß erinnert
- sei $A \in \mathbb{R}^{n \times n}$ spd mit $A = LL^T$,

$$L = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & & & \\ \hline & & & \\ l^{(2)} & & L^{(2)} & \end{pmatrix}$$

mit

$$l^{(2)} = \begin{pmatrix} l_{21} \\ \vdots \\ l_{n1} \end{pmatrix} \in \mathbb{R}^{n-1}, \quad L^{(2)} = \begin{pmatrix} l_{22} & & & \\ \vdots & \ddots & & \\ l_{n2} & \dots & l_{nn} \end{pmatrix} \in \mathbb{R}^{(n-1) \times (n-1)}$$

dann ist

$$A = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ \hline a_{21} & & & \\ \vdots & & B^{(2)} & \\ a_{n1} & & & \end{pmatrix} = LL^T,$$

also

$$\begin{aligned} A &= \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ \hline l_{21} & & & \\ \vdots & & L^{(2)} & \\ l_{n1} & & & \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \dots & l_{n1} \\ \hline 0 & & & \\ \vdots & & L^{(2)T} & \\ 0 & & & \end{pmatrix} \\ &= \begin{pmatrix} l_{11}^2 & l_{11}l_{21} & \dots & l_{11}l_{n1} \\ \hline l_{11}l_{21} & & & \\ \vdots & & L^{(2)T} + l^{(2)}l^{(2)T} & \\ l_{11}l_{n1} & & & \end{pmatrix} \end{aligned}$$

- ein Koeffizientenvergleich liefert

$$\begin{aligned} l_{11}^2 &= a_{11} \quad \Rightarrow \quad l_{11} = \sqrt{a_{11}} \\ l_{11}l_{i1} &= a_{i1} \quad \Rightarrow \quad l_{i1} = \frac{a_{i1}}{\sqrt{a_{11}}}, \quad i = 2, \dots, n \end{aligned}$$

d.h. die erste Spalte von L (und damit $l^{(2)}$) ist bekannt

- die Submatrix $L^{(2)}$ ist noch unbekannt aber da

$$L^{(2)}L^{(2)T} = B^{(2)} - l^{(2)}l^{(2)T} =: A^{(2)}$$

mit

$$a_{ij}^{(2)} = a_{ij} - l_{i1}l_{j1}, \quad i, j \geq 2$$

kann aus $A^{(2)}$ analog die erste Spalte von $L^{(2)}$ (d.h. zweite Spalte von L) bestimmt werden

- diese Prozedur wird wiederholt, bis alle Spalten von L berechnet sind
- der k -te Schritt des Cholesky-Verfahrens besteht also aus

$$\left. \begin{array}{l} l_{kk} = \sqrt{a_{kk}^{(k)}} \\ l_{ik} = \frac{a_{ik}^{(k)}}{l_{kk}} \quad i = k+1, \dots, n \end{array} \right\} \text{berechne } k\text{-te Spalte}$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}l_{jk} \quad \left. \begin{array}{l} i = k+1, \dots, n \\ j = k+1, \dots, i \end{array} \right\} \text{berechne } A^{(k+1)}$$

- ist $A = LL^T$ bekannt, so verfährt man zur Lösung von $Ax = b$ wie üblich:
 - $Ax = b \Leftrightarrow LL^T x = b \Leftrightarrow Ly = b, L^T x = y$
 - $Ly = b$ wird durch Vorwärtseinsetzen, $L^T x = y$ durch Rückwärtseinsetzen gelöst

Beispiel 22.

► wir untersuchen das System

$$A = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 10 & 5 \\ 4 & 5 & 21 \end{pmatrix}, \quad b = \begin{pmatrix} 10 \\ 14 \\ 13 \end{pmatrix}$$

und betrachten bei der Zerlegung nur die Diagonale und das untere Dreieck von A

► 1. Schritt:

$$\begin{array}{ccc|cc} 4 & & & 2 & 2 \\ 2 & 10 & & 1 & 10 - 1 \cdot 1 \\ 4 & 5 & 21 & 2 & 5 - 1 \cdot 2 \quad 21 - 2 \cdot 2 \end{array} = \begin{array}{cc|c} 1 & 9 \\ 2 & 3 & 17 \end{array}$$

► 2. Schritt:

$$\begin{array}{cc|c} 2 & & 9 \\ 1 & 3 & 17 \\ 2 & 1 & 16 \end{array} \Rightarrow \begin{array}{cc|c} 2 & & 3 \\ 1 & 3 & 17 - 1 \cdot 1 \\ 2 & 1 & 16 \end{array} = \begin{array}{cc|c} 1 & 3 \\ 2 & 1 & 16 \end{array}$$

► 3.Schritt:

$$\begin{array}{cc|c} 2 & & 3 \\ 1 & 3 & 16 \\ 2 & 1 & 16 \end{array} \Rightarrow \begin{array}{cc|c} 2 & & 3 \\ 1 & 3 & 4 \\ 2 & 1 & 4 \end{array} \Rightarrow L$$

► somit erhalten wir für $Ly = b$

$$\begin{pmatrix} 2 & 0 & 0 \\ 1 & 3 & 0 \\ 2 & 1 & 4 \end{pmatrix} y = \begin{pmatrix} 10 \\ 14 \\ 13 \end{pmatrix} \Leftrightarrow \begin{array}{l} y_1 = 5 \\ y_2 = \frac{14 - y_1}{3} = 3 \\ y_3 = \frac{13 - 2y_1 - y_2}{4} = 0 \end{array} \Leftrightarrow y = \begin{pmatrix} 5 \\ 3 \\ 0 \end{pmatrix}$$

und aus $L^T x = y$

$$\begin{pmatrix} 2 & 1 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 4 \end{pmatrix} x = \begin{pmatrix} 5 \\ 3 \\ 0 \end{pmatrix} \Leftrightarrow \begin{aligned} x_1 &= \frac{5 - x_2 - 2x_3}{2} = 2 \\ x_2 &= \frac{3 - x_3}{3} = 1 \\ x_3 &= 0 \end{aligned} \Leftrightarrow x = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$$

Zusammenfassung.

- Cholesky zerlegt $A \in \mathbb{R}^{n \times n}$ spd in LL^T , L untere Dreiecksmatrix
- verwendet mit Gaußelimination
- arbeitet immer auf symmetrischen Matrizen \Rightarrow halber Aufwand
- verschiedene Implementierungen möglich

Kapitel 3

Fehlerbetrachtung

3.1 Gut gestellte Probleme

- praktisch alle Aufgabenstellungen der numerischen Mathematik lassen sich als (eventuell sehr kompliziertes) Nullstellenproblem schreiben:
 - gegeben ist eine Funktion g und Eingabedaten x
 - suche y mit $g(x, y) = 0$

Beispiel 23.

- eine lineares Gleichungssystem $Ax = b$ ist äquivalent zu

$$g(x, y) = 0, \quad x = (A, b), \quad g(x, y) = Ay - b$$

- die größere der beiden Nullstellen von $t^2 + 2pt - q = 0$, $p, q > 0$ ist gegeben durch

$$t_2 = -p + \sqrt{p^2 + q},$$

d.h. das Nullstellenproblem ist äquivalent zu

$$g(x, y) = 0, \quad x = (p, q), \quad g(x, y) = y - (-p + \sqrt{p^2 + q})$$

- folgende Eigenschaften sollte ein solches Problem sinnvollerweise erfüllen:

- **Existenz:** es sollte Lösungen geben
- **Eindeutigkeit:** zu jedem x sollte es *genau* ein y geben
- **stetige Abhängigkeit** der Lösung y von den Eingabedaten x , d.h.

$$x' \rightarrow x \quad \Rightarrow \quad y' \rightarrow y$$

- stetige Abhängigkeit garantiert, dass (hinreichend) kleine Störungen der Eingabedaten nur kleine Veränderungen der Ergebnisse verursachen

Definition 24. Gilt für das Problem $g(x, y) = 0$ Existenz, Eindeutigkeit und stetige Abhängigkeit, so nennt man das Problem **gut gestellt** (well posed), ansonsten **schlecht gestellt** (ill posed)

- aus Existenz und Eindeutigkeit folgt, dass $g(x, y) = 0$ nach y aufgelöst werden kann
- dann existiert eine Lösungsfunktion f mit

$$g(x, y) = 0 \Leftrightarrow y = f(x)$$

d.h. f bildet die Eingabedaten auf die zugehörige Lösung ab

- **stetige Abhängigkeit** bedeutet damit

$$x' \rightarrow x \Rightarrow y' = f(x') \rightarrow f(x) = y$$

d.h. f ist stetig

Beispiel 25. Für die Probleme aus Beispiel 23 erhalten wir

$$f(x) = A^{-1}b, \quad x = (A, b), \quad \det(A) \neq 0$$

bzw.

$$f(x) = (-p + \sqrt{p^2 + q}), \quad x = (p, q)^T, \quad p, q > 0.$$

Beide Funktionen sind stetig, so dass beide Probleme gut gestellt sind

- müssen schlecht gestellte Probleme behandelt werden, so approximiert man sie in der Regel durch gut gestellte Ersatzprobleme (Regularisierung), wobei die Approximationseigenschaften genau kontrolliert werden müssen

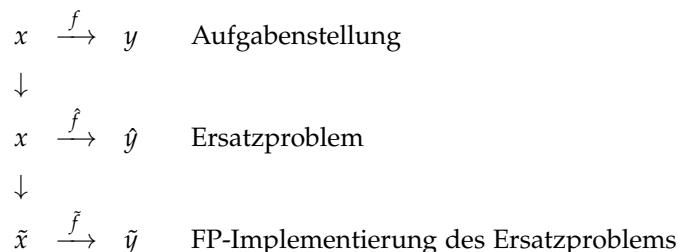
3.2 Fehlerrößen

- wir betrachten ein gut gestelltes Problem

$$y = f(x), \quad x \text{ gegeben}, \quad f : X \rightarrow Y$$

wobei X, Y Vektorräume mit Normen $\|\cdot\|_X, \|\cdot\|_Y$ sind

- zur numerischen Behandlung wird f durch ein (diskretes, endliches) Problem \hat{f} ersetzt, das anschließend in Floating-Point-Arithmetik als \tilde{f} implementiert wird



- y ist die *exakte* Lösung zu den *exakten* Eingangsdaten x
- \tilde{x} ist Floating-Point-Zahl, die sich bei Umwandlung von x ins Floating-Point-Format ergibt
- \hat{y} ist das Ergebnis des in *exakter Arithmetik* durchgeföhrten Ersatzproblems, wobei als Input die *exakten* Eingangsdaten x benutzt werden
- \tilde{y} ist die vom Computer mit Hilfe der Floating-Point-Implementierung des Ersatzproblems berechnete Näherung, wobei als Input die Floating-Point-Version \tilde{x} der exakten Daten x benutzt wird

Beispiel 26.

- wir betrachten das Originalproblem

$$f(x) = e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}, \quad x = \frac{5}{3}$$

mit exakter Lösung

$$y = e^{\frac{5}{3}} \approx 5.29449$$

- als Ersatzproblem benutzen wir

$$\hat{f}(x) = \sum_{k=0}^4 \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24},$$

d.h.

$$\hat{y} = 1 + \frac{5}{3} + \frac{1}{2}\left(\frac{5}{3}\right)^2 + \frac{1}{6}\left(\frac{5}{3}\right)^3 + \frac{1}{24}\left(\frac{5}{3}\right)^4 = \frac{10009}{1944} \approx 5.14866$$

- in dreistelliger Fließkommaarithmetik erhalten wir zunächst

$$\tilde{x} = 1.67$$

und dann durch Auswertung der Summe

$$\tilde{y} = 5.16$$

- die Qualität eines Approximationsverfahrens wird dadurch bestimmt, wie weit \tilde{y} von y abweicht
- um diese Abweichung quantitativ zu fassen, werden die folgenden Fehlergrößen definiert

Definition 27. Seien y und \tilde{y} wie oben, dann heißt

- $e = \tilde{y} - y$ **Fehler**
- $e_a = \|\tilde{y} - y\|_Y$ **absoluter Fehler**
- $e_r = \frac{\|\tilde{y} - y\|_Y}{\|y\|_Y}$ für $\|y\|_Y \neq 0$ **relativer Fehler**

Beispiel 28.

- wir betrachten das Nullstellenproblem aus Beispiel 23
- bringen wir den Algorithmus $t_2 = -p + \sqrt{p^2 + q}$ in die allgemeine Form von oben, so erhalten wir

$$x = (p, q)^T, \quad y = f(x), \quad f(x_1, x_2) = -x_1 + \sqrt{x_1^2 + x_2}, \quad f : \mathbb{R}^2 \rightarrow \mathbb{R}$$

- wir benutzen $\hat{f} = f$, d.h. wir benötigen kein Ersatzproblem
- \hat{f} wird wieder in dreistelliger Fließkommaarithmetik implementiert
- wegen $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ ist $Y = \mathbb{R}$ und wir benutzen $\|y\|_Y = |y|$

- für $p = 10, q = 1$ ist

$$\begin{aligned} y &= \sqrt{101} - 10 \approx 0.04988, & \tilde{y} &= 0, \\ e = \tilde{y} - y &\approx -0.04988, & e_a &= |\tilde{y} - y| \approx 0.04988, & e_r &= \frac{|\tilde{y} - y|}{|y|} = 1 \end{aligned} \quad (3.1)$$

- für $p = 100, q = 150$ ist

$$\begin{aligned} y &\approx 0.7472, & \tilde{y} &= 1, \\ e = \tilde{y} - y &\approx 0.2528, & e_a &= |\tilde{y} - y| \approx 0.2528, & e_r &= \frac{|\tilde{y} - y|}{|y|} = 0.34 \end{aligned} \quad (3.2)$$

- für $p = 10000, q = 1$ ist

$$\begin{aligned} y &\approx 0.00005, & \tilde{y} &= 0, \\ e = \tilde{y} - y &\approx -0.00005, & e_a &= |\tilde{y} - y| \approx 0.00005, & e_r &= \frac{|\tilde{y} - y|}{|y|} = 1 \end{aligned} \quad (3.3)$$

- in numerischen Verfahren versucht man den Fehler zu kontrollieren und dabei (wenn möglich) unter eine vom Benutzer vorgegebene Schranke ε zu drücken
- soll man besser den absoluten oder den relativen Fehler benutzen?
- vergleicht man die Ergebnisse des Beispiels, so fällt folgendes auf
 - in (3.2) ist e_a größer, aber e_r kleiner als in (3.1)
 - (3.3) hat einen sehr großen relativen Fehler, obwohl der absolute Fehler sehr klein ist
- der relative Fehler ist eigentlich aussagekräftiger, kann aber bei kleinen Absolutwerten von y wie in (3.3) eine unnötig hohe Genauigkeit erzwingen
- deshalb benutzt man in der Praxis häufig ein Kriterium, das beide Fehlerarten kombiniert

Definition 29. Seien $\varepsilon_a, \varepsilon_r$ gegebene Schranken für den absoluten bzw. relativen Fehler. Beim **gemischten Fehlerkriterium** versucht man eine Approximation \tilde{y} von y zu finden, mit

$$\|\tilde{y} - y\|_Y \leq \varepsilon_a + \|y\|_Y \varepsilon_r$$

- betrachten wir $\|\tilde{y} - y\|_Y \leq \varepsilon_a + \|y\|_Y \varepsilon_r$ genauer:
 - ist der exakte Absolutwert $\|y\|_Y$ sehr klein, dann dominiert ε_a und

$$e_a = \|\tilde{y} - y\|_Y \lesssim \varepsilon_a$$

- ist $\|y\|_Y$ sehr groß, dann dominiert auf der rechten Seite $\|y\|_Y \varepsilon_r$ und

$$\|\tilde{y} - y\|_Y \lesssim \|y\|_Y \varepsilon_r \quad \text{bzw.} \quad e_r = \frac{\|\tilde{y} - y\|_Y}{\|y\|_Y} \lesssim \varepsilon_r$$

Beispiel 30.

- wir wenden das gemischte Kriterium mit $\varepsilon_a = \varepsilon_r = 10^{-2}$ auf die Aufgabenstellung von Beispiel 28 an
- für $p = 10, q = 1$ war $y = \sqrt{101} - 10 \approx 0.04988, \tilde{y} = 0$, so dass

$$|\tilde{y} - y| \approx 0.04988, \quad \varepsilon_a + |y| \varepsilon_r \approx 1.04988 \cdot 10^{-2}$$

und somit

$$|\tilde{y} - y| > \varepsilon_a + |y| \varepsilon_r,$$

d.h. das Kriterium lehnt die Näherung \tilde{y} als zu ungenau ab

- für $p = 10000, q = 1$ war $y \approx 0.00005, \tilde{y} = 0$, so dass

$$|\tilde{y} - y| \approx 0.00005, \quad \varepsilon_a + |y| \varepsilon_r \approx 1.00005 \cdot 10^{-2}$$

und somit

$$|\tilde{y} - y| < \varepsilon_a + |y| \varepsilon_r,$$

d.h. \tilde{y} wird als hinreichend genau akzeptiert, obwohl der relative Fehler e_r gleich 1 ist

3.3 Fehlerquellen

- wie wir oben gesehen haben, durchlaufen wir zur numerischen Behandlung eines gut gestellten Problems $y = f(x)$ folgende Schritte:

$$\begin{array}{ccc} x & \xrightarrow{f} & y & \text{Originalproblem} \\ \downarrow & & & \\ x & \xrightarrow{\hat{f}} & \hat{y} & \text{Ersatzproblem} \\ \downarrow & & & \\ \tilde{x} & \xrightarrow{\tilde{f}} & \tilde{y} & \text{FP-Implementierung des Ersatzproblems} \end{array}$$

- wir betrachten den Fehler

$$\tilde{y} - y = \tilde{f}(\tilde{x}) - f(x) \tag{3.4}$$

und versuchen die verschiedenen Fehlerquellen zu lokalisieren

- dazu formen wir (3.4) wie folgt um

$$\begin{aligned} \tilde{y} - y &= \tilde{f}(\tilde{x}) - \hat{f}(\tilde{x}) \\ &\quad + \hat{f}(\tilde{x}) - f(\tilde{x}) \\ &\quad + f(\tilde{x}) - f(x) \end{aligned}$$

und untersuchen die einzelnen Summanden

- **Eingabefehler** $f(\tilde{x}) - f(x)$

- wegen der Umwandlung der Eingabedaten x in Floating-Point-Darstellung \tilde{x} (siehe Abschnitt 3.3.1) tritt dieser Fehler immer auf und wird deshalb auch als **unvermeidbarer Fehler** bezeichnet
- die Größe dieses Fehleranteils hängt davon ab, wie empfindlich das Originalproblem f auf Störungen in den Eingabedaten reagiert

- **Verfahrensfehler** $\hat{f}(\tilde{x}) - f(\tilde{x})$

- hier wird das in *exakter Arithmetik* gerechnete Ersatzproblem \hat{f} mit dem Originalproblem f für den selben Input \tilde{x} verglichen
- die Größe dieses Fehleranteils hängt von der Qualität des Ersatzproblems ab
- **Akkumulierte Rundungsfehler** $\tilde{f}(\tilde{x}) - \hat{f}(\tilde{x})$
 - das Ersatzproblem \hat{f} besteht aus einer endlichen Anzahl von Elementaroperationen (Additionen, Multiplikationen etc.)
 - bei der Floating-Point-Implementierung \tilde{f} des Ersatzproblems werden diese Operationen durch ihre Floating-Point-Äquivalente ersetzt
 - jede Floating-Point-Operation erzeugt in der Regel Rundungsfehler
 - diese Fehler akkumulieren sich
 - die Größe dieses Fehleranteils hängt davon ab, wie robust das Ersatzproblem auf eine Floating-Point-Implementierung reagiert
- in den folgenden Abschnitten werden die einzelnen Fehlerquellen genauer untersucht

3.3.1 Floating-Point-Darstellung von Zahlen

- bei der Anwendung numerischer Verfahren müssen grundsätzlich die Input-Daten x (reelle Zahlen) in eine für den Computer geeignete Form \tilde{x} , die entsprechende Floating-Point-Darstellung, umgewandelt werden
- welche Fehler dabei entstehen können, werden wir nun genauer untersuchen
- vom Taschenrechner ist man die Verarbeitung von Dezimalzahlen und -brüchen gewohnt
- dabei unterscheidet man im wesentlichen zwei Darstellungen
 - **Festkommadarstellung**
 - die Anzahl der Nachkommastellen ist fest
 - bei zwei Nachkommastellen erhalten wir z.B.

$$12.47 \quad -1.00$$

- **Exponentiendarstellung**

- die Zahl wird als Produkt einer Festpunktzahl und einer Zehnerpotenz dargestellt

$$1.23E01 = 1.23 \cdot 10^1 = 12.3$$

$$1.23E-01 = 1.23 \cdot 10^{-1} = 0.123$$

- diese Darstellung wird auch als **Fließkommadarstellung** bezeichnet

- Dezimalbrüche sind Zahlen zur Basis 10:

$$12.34 = 1 \cdot 10^1 + 2 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

wobei die Ziffern ganze Zahlen zwischen 0 und 9 sind

- das geht analog auch für andere ganzzahlige Basen $b > 1$

Definition 31. Eine **Fließkommazahl** zur Basis b hat die Darstellung

$$v \cdot m \cdot b^e \quad (3.5)$$

Dabei ist

- b die **Basis**, eine natürliche Zahl > 1
- v das **Vorzeichen**, also ± 1
- m die **Mantisse**, ein b -adischer Bruch

$$m = m_{-k} \dots m_{-1} m_0 \cdot m_1 \dots m_l = m_{-k} b^k + \dots + m_l b^{-l} = \sum_{i=-k}^l m_i b^{-i} \quad (3.6)$$

mit Ziffern $m_i \in \{0, 1, \dots, b-1\}$ und Stellenzahl $k+l+1$

- e der **Exponent**, eine ganze Zahl

Beispiel 32.

- $b = 10, -43.21 \cdot 10^6$
- $b = 2, 10.11 \cdot 2^1 = (1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}) \cdot 2 = (2 + \frac{1}{2} + \frac{1}{4}) \cdot 2 = \frac{11}{2}$

- in dieser Form sind Fließkommadarstellungen einer Zahl nicht eindeutig, denn durch Änderung des Exponenten erhält man z.B.

$$-43210000 = -43.21 \cdot 10^6 = -4.321 \cdot 10^7$$

- verlangt man von der Mantisse, dass sie genau eine Vorkommastelle hat die nicht verschwinden darf (d.h. $k=0$ und $m_0 \neq 0$), dann ist die Fließkommadarstellung für alle Zahlen $\neq 0$ eindeutig

Lemma 33. Die **normalisierte** Darstellung

$$\pm(m_0 \cdot m_1 \dots m_l) b^e = \pm \left(\sum_{i=0}^l m_i b^{-i} \right) b^e, \quad m_0 \neq 0$$

ist für Zahlen $\neq 0$ eindeutig

- es gibt reelle Zahlen, die nicht mit endlicher Mantissenlänge darstellbar sind (in keiner Basis, z.B. $\pi, e, \sqrt{2}$ etc.)
- im Computer steht nur eine endliche (feste) Anzahl von Stellen zur Verfügung
- Floating-Point-Formate fester Länge können also nur eine Teilmenge des \mathbb{R} abbilden
- Eingabedaten müssen zunächst in das benutzte Floating-Point-Format umgewandelt werden, wobei in der Regel gerundet und abgeschnitten werden muss
- wie groß ist der maximale Fehler bei Umwandlung einer reellen Zahl in ein Floating-Point-Format?

Beispiel 34.

- wir betrachten Dezimalzahlen mit dreistelliger Mantisse ($l = 2$)

reelle Zahl	Rundung	max. Fehler
1.234***...	1.23	0.004999...
1.235***...	1.24	0.005

- der maximale Fehler ist kleiner als $5 \cdot 10^{-3} = \frac{10^{-2}}{2} = \frac{10^{-l}}{2}$

- jede reelle Zahl $x \in \mathbb{R}$ kann als normalisierte Fließkommazahl zur Basis b mit "unendlich langer Mantisse"

$$\begin{aligned}\mathbb{R} \ni x &= \pm(m_0 \cdot m_1 \dots m_l m_{l+1} \dots) b^e \\ &\quad b^0 \ b^{-1} \dots b^{-l} b^{-(l+1)} \dots\end{aligned}$$

dargestellt werden

- durch Runden erzeugt man aus x eine Fließkommazahl \tilde{x} mit Stellenzahl $l + 1$
- dazu wird in der Regel dasjenige \tilde{x} ausgewählt, das x am nächsten liegt (**nearest** Rundung)
- dabei treten folgende Probleme auf
 - es kann zwei \tilde{x} geben mit dem gleichen minimalen Abstand zu x , z.B. für $b = 10$ und $l = 2$ erhalten wir für $x = 1.235$

$$\tilde{x}_1 = 1.23, \quad \tilde{x}_2 = 1.24, \quad |x - \tilde{x}_1| = |x - \tilde{x}_2| = 0.005$$

- \tilde{x} muss eventuell renormalisiert werden, z.B. für $b = 10, l = 2$,

$$x = 9.996 \cdot 10^{-4} \quad \tilde{x} = 1.00 \cdot 10^{-3}$$

- ohne Berücksichtigung dieser Spezialfälle ist

$$\tilde{x} = \begin{cases} \pm(m_0.m_1 \dots m_l)b^e & \text{für } m_{l+1} < rd(\frac{b}{2}) \\ \pm(m_0.m_1 \dots (m_l + 1))b^e & \text{für } m_{l+1} \geq rd(\frac{b}{2}) \end{cases}$$

d.h. es wird auf- bzw. abgerundet

- für den Fehler erhalten wir

$$e_a = |\tilde{x} - x| \leq \frac{b}{2} b^{-(l+1)} b^e = \frac{1}{2} b^{-l} b^e$$

und für den relativen Fehler

$$e_r = \frac{e_a}{|x|} \leq \frac{\frac{1}{2} b^{-l} b^e}{(m_0 \cdot m_1 \dots m_l \dots) b^e} \leq \frac{1}{2} b^{-l}$$

wobei es Zahlen gibt, so dass Gleichheit gilt

- man definiert die obere Schranke für e_r als Maschinengenauigkeit

Definition 35. Betrachten wir normierte Fließkommazahlen zur Basis b mit Mantissenlänge l , so bezeichnet

$$\varepsilon_M = \frac{1}{2} b^{-l}$$

die **Maschinengenauigkeit**

Bemerkung 36.

◊ ε_M ist der maximale relative Fehler, der bei der Umwandlung einer reellen Zahl (bei nearest Rundung) entstehen kann

◊ ε_M hängt nur von der Basis und der Mantissenlänge, *nicht* aber vom Exponenten ab

◊ manchmal wird die Maschinengenauigkeit auch als der Abstand von 1 zur nächst größeren Floating-Point-Zahl definiert (z.B. in MATLAB), was zu einem doppelt so großen Wert $\tilde{\varepsilon}_M = b^{-l}$ führt

- Floating-Point-Darstellungen auf dem Computer arbeiten in der Regel mit der Basis $b = 2$
- früher wurden Floating-Point-Operationen als Unterprogramme in Programmbibliotheken implementiert, wodurch sehr viele proprietäre Formate entstanden
- heute werden Floating-Point-Operationen direkt von der Hardware ausgeführt
- die Standardisierung nach **IEEE 754** legt eine ganze Reihe von Floating-Point-Formaten mit $b = 2$ und das Verhalten von arithmetischen Grundoperationen fest
- die wichtigsten Formate sind **single** und **double**

- **single:**

- eine single Zahl besteht aus einem Vorzeichen, einer 24-stelligen Mantisse und einem 8-stelligen Exponenten
- wegen $b = 2$ ist $m_0 = 1$ für alle normalisierten Zahlen und wird deshalb nicht extra gespeichert
- Vorzeichen sowie die (wesentlichen) Stellen der Mantisse und des Exponenten passen deshalb in ein 32bit Wort

$$vm_1 \dots m_{23}\tilde{e}_1 \dots \tilde{e}_8$$

- wegen der 24-stelligen Mantisse ist $l = 23$ und die Maschinengenauigkeit ist

$$\varepsilon_M = \frac{1}{2}2^{-23} = 2^{-24} \approx 5.96 \cdot 10^{-8}$$

was etwa 7 gültigen Dezimalstellen entspricht

- der Exponent wird in einer vorzeichenlosen 8bit int-Zahl \tilde{e} kodiert, d.h. $\tilde{e} \in \{0, \dots, 255\}$
- den tatsächlichen Wert e erhält man durch

$$e = \tilde{e} - 127 = \tilde{e} - (2^7 - 1), \quad e \in \{-127, \dots, 128\}$$

- für *normale* Zahlen ist $e \in \{-126, \dots, 127\}$, die Exponenten -127 und 128 sind für Spezialfälle reserviert:

- $e = 128$ wird bei verschwindender Mantisse als $\pm\infty$, sonst als NaN (Not a Number) interpretiert
- $e = -127$ wird bei verschwindender Mantisse als 0, sonst als "Subnormal" interpretiert

- **double:**

- ist analog zu single als 64bit Format aufgebaut

$$vm_1 \dots m_{52}\tilde{e}_1 \dots \tilde{e}_{11}$$

- mit $l = 52$ erhalten wir $\varepsilon_M = 2^{-53} \approx 1.1 \cdot 10^{-16}$, also ungefähr 16 gültigen Dezimalstellen
- der Exponent e berechnet sich nach $e = \tilde{e} - (2^{10} - 1) = \tilde{e} - 1023$
- oft findet man auch extended double (80bit) Formate
- als Standard benutzt IEEE754 **nearest even** Rundung, d.h. nearest Rundung, wobei bei Mehrdeutigkeiten dasjenige \tilde{x} benutzt wird, dessen letzte Mantissenstelle gleich 0 ist
- bei der Umwandlung von Dezimalbrüchen in das Binärformat entstehen selbst bei einfachen Beispielen Rundungsfehler

Beispiel 37.

- wir wandeln $x = 0.2$ in das IEEE754 single Format um
- $x = 0.2$ ist ein periodischer Binärbruch

$$0.2 = 0.\overline{0011} = 0.00110011\dots$$

- Normalisieren liefert zunächst

$$1.100110011001100110011001100\dots \cdot 2^{-3}$$

- eine nearest even Rundung der Mantisse auf 24 Stellen ergibt

$$\begin{array}{r} 1.100110011001100110011001100\dots \\ 1.10011001100110011001101 \end{array}$$

also

$$\tilde{x} = 1.10011001100110011001101 \cdot 2^{-3} \approx 0.20000000298023$$

3.3.2 Eingabefehler, Kondition

- wir untersuchen nun das Verhalten des Eingabefehlers $f(\tilde{x}) - f(x)$
- da unser Problem gut gestellt ist, ist f stetig, d.h. für alle $\varepsilon > 0$ gibt es ein (hinreichend kleines) $\delta > 0$ so dass

$$||f(\tilde{x}) - f(x)||_Y < \varepsilon \quad \text{falls} \quad ||\tilde{x} - x||_X < \delta$$

- $\tilde{x} - x$ ist die Abweichung zwischen x und seiner Floating-Point-Darstellung \tilde{x} , die in der Regel *nicht* beliebig klein wird
- wie das folgende Beispiel zeigt, kann das zu Problemen führen

Beispiel 38.

- wir betrachten

$$y = f(x) := 1 - \sqrt[13]{x-1}$$

- f ist stetig, also ist das Problem gut gestellt

- für

$$x = 1 + 2^{-26}$$

erhalten wir im IEEE754 single Format ($\varepsilon_M = 2^{-24}$)

$$\tilde{x} = 1$$

und somit

$$\begin{aligned} y &= f(x) = 1 - \sqrt[13]{2^{-26}} = 1 - 2^{-2} = \frac{3}{4} = 0.75 \\ \tilde{y} &= f(\tilde{x}) = 1 \end{aligned}$$

- nach Umwandlung von x in die IEEE754 single Zahl \tilde{x} stimmen bei *exakter* Be-rechnung keine Dezimalstellen von $f(x)$ und $f(\tilde{x})$ überein, d.h. f reagiert sehr empfindlich auf Störungen in x

- betrachtet man die absoluten bzw. relativen Fehler in x bzw. $y = f(x)$

$$|\tilde{x} - x| = 2^{-26} < 1.5 \cdot 10^{-8} \quad \Rightarrow \quad |\tilde{y} - y| = \frac{1}{4}$$

bzw.

$$\frac{|\tilde{x} - x|}{|x|} \stackrel{x > 1}{\leq} |\tilde{x} - x| < 1.5 \cdot 10^{-8} \quad \Rightarrow \quad \frac{|\tilde{y} - y|}{|y|} = \frac{\frac{1}{4}}{\frac{3}{4}} = \frac{1}{3}$$

so sieht man, dass f die Fehler so stark verstärkt, dass die Abweichungen auf der Output-Seite inakzeptabel groß sind

- das Problem ist zwar gut gestellt (f ist stetig), aber **schlecht konditioniert**

- um die Zusammenhänge zwischen Fehlern auf der Input-Seite x und Fehlern auf der Output-Seite $f(x)$ zu beschreiben, führt man Konditionszahlen ein

Definition 39. Sei $f : X \rightarrow Y$. Die kleinsten Konstanten κ_a, κ_r mit

$$\begin{aligned} \|f(x') - f(x)\|_Y &\leq \kappa_a \|x' - x\|_X \\ \frac{\|f(x') - f(x)\|_Y}{\|f(x)\|_Y} &\leq \kappa_r \frac{\|x' - x\|_X}{\|x\|_X} \quad \|x' - x\|_X \leq \delta \end{aligned} \tag{3.7}$$

heißen **absolute Kondition** bzw. **relative Kondition** von f in x und hängen von den benutzten Normen, x und δ ab.

Bemerkung 40.

- ◊ die Konditionszahlen beschreiben, wie stark f Störungen in x schlimmstenfalls verstärkt
- ◊ kleine Konditionszahlen deuten auf ein gut konditioniertes, große auf ein schlecht konditioniertes Problem hin
- ◊ werden die Störungen in x durch die Umwandlung in ein Floating-Point-Format verursacht, d.h. $x' = \tilde{x}$, so gilt

$$e_r(x) = \frac{\|\tilde{x} - x\|_X}{\|x\|_X} \leq \varepsilon_M$$

bzw.

$$e_r(y) = \frac{\|\tilde{y} - y\|_Y}{\|y\|_Y} \leq \kappa_r \frac{\|\tilde{x} - x\|_X}{\|x\|_X} \leq \kappa_r \varepsilon_M,$$

also

$$e_r(y) \leq \varepsilon_r \quad \text{falls} \quad \kappa_r \leq \frac{\varepsilon_r}{\varepsilon_M},$$

d.h. eine sinnvolle Berechnung von $f(x)$ (relativer Fehler unterhalb ε_r) ist nur möglich, wenn die Kondition κ_r in Abhängigkeit von der Maschinengenauigkeit ε_M eine gewisse Grenze nicht überschreitet

Beispiel 41.

- im IEEE754 single Format ist $\varepsilon_M = 2^{-24} \approx 6 \cdot 10^{-8}$
- soll der relative Fehler des Ergebnisses unterhalb von 1% liegen ($\varepsilon_r \leq 10^{-2}$)
- gilt

$$\kappa_r \leq \frac{\varepsilon_r}{\varepsilon_M} = \frac{10^{-2}}{6 \cdot 10^{-8}} \approx 1.67 \cdot 10^5$$

so ist das sicher erfüllt

- für stetig differenzierbares f kann man relativ einfach obere Schranken für die Konditionszahlen herleiten
- wir betrachten zunächst den einfachen Fall $f : \mathbb{R} \rightarrow \mathbb{R}$
- nach dem Mittelwertsatz gilt

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

mit

$$\xi \in \text{co}(x, x') = \{z \mid z = x + t(x' - x), t \in [0, 1]\} = [\min(x, x'), \max(x, x')]$$

und damit

$$|f(x') - f(x)| = |f'(\xi)| |x' - x| \leq \max_{\xi \in \text{co}(x, x')} |f'(\xi)| |x' - x|,$$

- für alle x' mit $|x' - x| \leq \delta$ gilt

$$\text{co}(x, x') \subset [x - \delta, x + \delta],$$

und somit

$$\xi \in \text{co}(x, x') \Rightarrow |\xi - x| \leq \delta$$

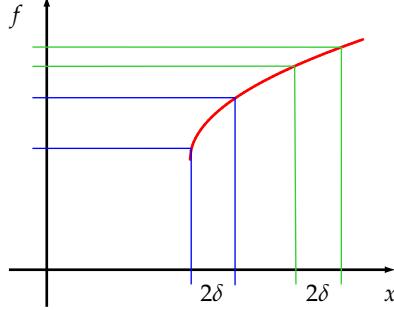
- damit erhalten wir schließlich die Abschätzungen

$$\begin{aligned} \kappa_a &\leq \max_{|\xi-x| \leq \delta} |f'(\xi)| \\ \kappa_r &\leq \frac{|x|}{|f(x)|} \max_{|\xi-x| \leq \delta} |f'(\xi)| \end{aligned} \tag{3.8}$$

Beispiel 42.

- wir betrachten die Funktion

$$f(x) = 1 + \sqrt{x-1}, \quad x > 1$$



- f ist stetig differenzierbar mit

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

- da $|\xi - x| \leq \delta$ gleichbedeutend mit $\xi \in [x - \delta, x + \delta]$ ist, erhalten wir

$$\max_{|\xi-x| \leq \delta} |f'(\xi)| = \max_{\xi \in [x-\delta, x+\delta]} \frac{1}{2\sqrt{\xi-1}} = \frac{1}{2\sqrt{x-\delta-1}}$$

- für $x = 2, \delta = 10^{-8}$ ist

$$\max_{|\xi-x| \leq \delta} |f'(\xi)| = \frac{1}{2\sqrt{1-10^{-8}}} \leq \frac{1}{2\sqrt{\frac{1}{2}}} = \sqrt{\frac{1}{2}} < 1$$

und somit

$$\begin{aligned} \varkappa_a &< 1 \\ \varkappa_r &< \frac{|2|}{|f(2)|} \cdot 1 = \frac{2}{1 + \sqrt{1}} = 1 \end{aligned}$$

- für $x = 1 + 2 \cdot 10^{-8}, \delta = 10^{-8}$ ist

$$\max_{|\xi-x| \leq \delta} |f'(\xi)| = \frac{1}{2\sqrt{10^{-8}}} \leq \frac{10^4}{2} = 5000$$

und somit

$$\begin{aligned} \varkappa_a &\leq 5000 \\ \varkappa_r &\leq \frac{|1 + 2 \cdot 10^{-8}|}{|f(1 + 2 \cdot 10^{-8})|} \cdot 5000 = \frac{1 + 2 \cdot 10^{-8}}{1 + \sqrt{2 \cdot 10^{-8}}} \cdot 5000 < 5000 \end{aligned}$$

- da die Abschätzung des Maximums oft recht aufwändig ist, ersetzt man sie häufig durch eine einfachere Näherung
- für zweimal stetig differenzierbares f folgt aus der Taylor-Entwicklung von f um x

$$f(x') = f(x) + f'(x)(x' - x) + R(x', x), \quad R(x', x) = O(|x' - x|^2)$$

- ist $|x' - x|$ klein, dann vernachlässigt man das Restglied R und erhält die Näherung (Linearisierung)

$$f(x') \approx f(x) + f'(x)(x' - x)$$

- somit ist

$$|f(x') - f(x)| \approx |f'(x)| |x' - x|$$

und

$$\varkappa_a \approx |f'(x)|$$

- analog erhält man für die Verstärkung des relativen Fehlers

$$\varkappa_r \approx \frac{|x|}{|f(x)|} |f'(x)|$$

- diese Näherungen bezeichnet man auch als **Fehlerfortpflanzungsformeln** der **differentiellen Fehleranalyse**

Beispiel 43.

- wir betrachten noch einmal Beispiel 42 von oben

$$f(x) = 1 + \sqrt{x-1}, \quad x > 1$$

- mit den differentiellen Fehlerformeln erhält man für $x > 1$

$$\varkappa_a \approx |f'(x)| = \frac{1}{2\sqrt{x-1}}$$

bzw.

$$\varkappa_r \approx \frac{|x|}{|f(x)|} |f'(x)| = \frac{x}{1 + \sqrt{x-1}} \frac{1}{2\sqrt{x-1}}$$

- für $x = 2$ folgt damit

$$\varkappa_a \approx \frac{1}{2}, \quad \varkappa_r \approx \frac{1}{2}$$

bzw. für $x = 1 + 2 \cdot 10^{-8}$

$$\varkappa_a \approx 3535.5, \quad \varkappa_r \approx 3535.0$$

- für allgemeines $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ geht man analog vor
- der Mittelwertsatz nimmt in diesem Fall die Form

$$f(x') = f(x) + \left(\int_0^1 f'((x + t(x' - x)) dt \right) (x' - x) \quad (3.9)$$

an, wobei

$$f'(x) = \begin{pmatrix} \partial_{x_1} f_1(x) & \dots & \partial_{x_n} f_1(x) \\ \vdots & & \vdots \\ \partial_{x_1} f_m(x) & \dots & \partial_{x_n} f_m(x) \end{pmatrix}$$

die **Jacobi-Matrix** von f in x ist

- benutzt man in \mathbb{R}^n die Norm $\|\cdot\|_X$ und in \mathbb{R}^m die Norm $\|\cdot\|_Y$ so folgt

$$\begin{aligned} \|f(x') - f(x)\|_Y &\leq \left\| \int_0^1 f'((x + t(x' - x)) dt \right\|_{X,Y} \|x' - x\|_X \\ &\leq \max_{\xi \in \text{co}(x, x')} \|f'(\xi)\|_{X,Y} \|x' - x\|_X, \end{aligned}$$

wobei $\|\cdot\|_{X,Y}$ die zugehörige induzierte Matrixnorm ist

- für alle x' mit $\|x' - x\|_X \leq \delta$ gilt dann wieder

$$\begin{aligned} \varkappa_a &\leq \max_{\|\xi - x\|_X \leq \delta} \|f'(\xi)\|_{X,Y} \\ \varkappa_r &\leq \frac{\|x\|_X}{\|f(x)\|_Y} \max_{\|\xi - x\|_X \leq \delta} \|f'(\xi)\|_{X,Y} \end{aligned} \quad (3.10)$$

- analog erhalten wir für die **Fehlerfortpflanzungsformeln** der **differentiellen Fehleranalyse**

$$\varkappa_a \approx \|f'(x)\|_{X,Y}, \quad \varkappa_r \approx \frac{\|x\|_X}{\|f(x)\|_Y} \|f'(x)\|_{X,Y}$$

Beispiel 44.

► wir betrachten

$$f : X = \mathbb{R}^2 \rightarrow Y = \mathbb{R}, \quad (x_1, x_2) = x \mapsto x_1 + x_2$$

und benutzen Normen

$$\|x\|_X = \|x\|_\infty = \max(|x_1|, |x_2|) \quad \|y\|_Y = \|y\|_\infty = |y|$$

► wir erhalten

$$f'(x) = (\partial_{x_1} f(x), \partial_{x_2} f(x)) = (1, 1)$$

► aufgrund der benutzten Vektornormen folgt

$$\|f'(x)\|_{X,Y} = \|f'(x)\|_\infty = \|(1, 1)\|_\infty = 2$$

und somit

$$\varkappa_a \approx 2$$

bzw.

$$\varkappa_r \approx 2 \frac{\|x\|_\infty}{|f(x)|} = 2 \frac{\max(|x_1|, |x_2|)}{|x_1 + x_2|}$$

3.3.3 Verfahrensfehler

- das Originalproblem f muss, je nach Aufgabenstellung, durch ein **Ersatzproblem** \hat{f} endlich bzw. diskret gemacht werden
- für das selbe f können in der Regel viele verschiedene Ersatzprobleme \hat{f} konstruiert werden
- die Differenz zwischen dem in *exakter Arithmetik* gerechneten Ersatzproblem \hat{f} und dem Originalproblem f für den selben Input \tilde{x} definiert den Verfahrensfehler

$$\hat{f}(\tilde{x}) - f(\tilde{x})$$

- Ziel bei der Konstruktion eines Ersatzproblems ist es, den Verfahrensfehler mit möglichst wenig Aufwand möglichst klein zu machen
- in der Regel hängen Qualität und Aufwand des Ersatzproblems von einem **Diskretisierungsparameter** $h > 0$ ab
- ein vernünftig gewähltes Ersatzproblem geht für $h \rightarrow 0$ in das Originalproblem über, d.h. der Verfahrensfehler geht für $h \rightarrow 0$ gegen 0 (und der Aufwand in der Regel gegen unendlich)

Beispiel 45. Die LU Zerlegung zum Lösen linearer Gleichungssysteme liefert in endlichen vielen Schritten die exakte Lösung, d.h. es kann $\hat{f} = f$ benutzt werden, weshalb der Verfahrensfehler hier 0 ist

Beispiel 46.

- ▶ für das Originalproblem

$$f(x) = e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!},$$

benutzen wir das Ersatzproblem

$$\hat{f}(x) = \sum_{k=0}^{n_h} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \dots + \frac{x^{n_h}}{n_h!}, \quad n_h = \lfloor \frac{1}{h} \rfloor,$$

wobei für $z \in \mathbb{R}$ der Ausdruck $\lfloor z \rfloor$ die größte ganze Zahl kleiner z liefert

- ▶ mit $h \rightarrow 0$ gilt $n_h \rightarrow \infty$, also

$$\hat{f}(x) \xrightarrow{h \rightarrow 0} f(x)$$

Beispiel 47.

- ▶ für das Originalproblem

$$f(g) = \int_0^1 g(t) dt, \quad g \text{ stetig}$$

benutzen wir als Ersatzproblem einmal die linksseitige Rechteckregel

$$\hat{f}_1(g) = \sum_{k=0}^{n_h-1} \frac{1}{n_h} g\left(\frac{k}{n_h}\right), \quad n_h = \lfloor \frac{1}{h} \rfloor,$$

bzw. die Trapezregel

$$\hat{f}_2(g) = \sum_{k=0}^{n_h-1} \frac{1}{2n_h} \left(g\left(\frac{k}{n_h}\right) + g\left(\frac{k+1}{n_h}\right) \right), \quad n_h = \lfloor \frac{1}{h} \rfloor,$$

- für $h \rightarrow 0$ gilt

$$\hat{f}_{1,2}(g) \xrightarrow{h \rightarrow 0} f(g)$$

wobei man zeigen kann, dass \hat{f}_2 schneller gegen f konvergiert als \hat{f}_1

- das genaue Verhalten des Verfahrensfehlers wird im Einzelfall für jedes Ersatzproblem separat untersucht, insbesondere das asymptotische Verhalten für $h \rightarrow 0$ (**Konvergenzuntersuchung**)

3.3.4 Akkumulierter Rundungsfehler, Floating-Point-Arithmetik

- der akkumulierte Rundungsfehler $\tilde{f}(\tilde{x}) - \hat{f}(\tilde{x})$ vergleicht das in *exakter* Arithmetik gerechneten Ersatzproblem \hat{f} mit einer seiner Floating-Point-Implementierungen \tilde{f}
- zu einem Ersatzproblem kann es mehrere Floating-Point-Implementierung geben, die mathematisch äquivalent (d.h. in exakter Arithmetik identisch) sind, aber unterschiedliche Floating-Point-Ergebnisse liefern

Beispiel 48.

- für das Originalproblem

$$f(x) = e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!},$$

benutzen wir das Ersatzproblem

$$\hat{f}(x) = \sum_{k=0}^n \frac{x^k}{k!}$$

und die beiden Floating-Point Implementierungen

$$\begin{aligned}\tilde{f}_1(x) &= 1 + x + \frac{x^2}{2} + \dots + \frac{x^n}{n!} \\ \tilde{f}_2(x) &= \frac{x^n}{n!} + \dots + \frac{x^2}{2} + x + 1\end{aligned}$$

- für $x = \frac{1}{2}$ ist der exakte Wert $e^{\frac{1}{2}} \approx 1.6487$ und Floating-Point-Rechnung mit 3 Stellen liefert

n	1	2	3	4	5
$\tilde{f}_1(0.5)$	1.5	1.62	1.64	1.64	1.64
$\tilde{f}_2(0.5)$	1.5	1.62	1.65	1.65	1.65

- das Ersatzproblem \hat{f} setzt sich in der Regel aus einer großen Anzahl von elementaren Rechenoperationen \hat{f}_i zusammen, die aus \tilde{x} über Zwischenergebnisse \hat{x}_i schließlich $\hat{f}(\tilde{x})$ erzeugen

$$\tilde{x} \xrightarrow{\hat{f}_1} \hat{x}_1 \xrightarrow{\hat{f}_2} \hat{x}_2 \xrightarrow{\hat{f}_3} \dots \xrightarrow{\hat{f}_p} \hat{f}(\tilde{x})$$

- in der Implementierung \tilde{f} werden die \hat{f}_i durch ihr Floating-Point-Äquivalent \tilde{f}_i ersetzt

$$\tilde{x} \xrightarrow{\tilde{f}_1} \tilde{x}_1 \xrightarrow{\tilde{f}_2} \tilde{x}_2 \xrightarrow{\tilde{f}_3} \dots \xrightarrow{\tilde{f}_p} \tilde{f}(\tilde{x})$$

- aufgrund der endlichen Mantissenlänge können selbst diese elementaren Operationen keine exakten Ergebnisse liefern, was wir anhand einiger Grundrechenarten jetzt genauer untersuchen werden
- die Floating-Point-Implementierungen der Grundrechenarten bezeichnen wir mit $\oplus, \ominus, \odot, \oslash$
- die Arbeitsweise von \oplus, \odot soll anhand von Beispielen mit $b = 10$ erklärt werden, die realen Implementierungen im Binärsystem arbeiten analog
- betrachten wir zunächst die Addition bzw. Subtraktion
 - die Subtraktion ist eine Addition mit Vorzeichenwechsel, muss also nicht gesondert betrachtet werden
 - für die Addition von Zahlen mit gleichem Vorzeichen wie z.B.

$$1.23 \cdot 10^0 \oplus 7.89 \cdot 10^{-3}$$

werden folgende Schritte durchlaufen:

- Angleichen der Exponenten

$$\begin{array}{r} 1.23 \quad \cdot 10^0 \\ 0.00789 \quad \cdot 10^0 \end{array}$$

- Addieren der Mantissen

$$1.23789 \cdot 10^0$$

- Runden der Mantisse und gegebenenfalls Renormalisieren des Ergebnisses

$$1.24 \cdot 10^0$$

- bei verschiedenen Vorzeichen wie z.B.

$$1.23 \cdot 10^0 \ominus 1.22 \cdot 10^0$$

erhalten wir analog:

- Angleichen der Exponenten

$$\begin{array}{r} 1.23 \cdot 10^0 \\ -1.22 \cdot 10^0 \end{array}$$

- Addieren der Mantissen

$$0.01 \cdot 10^0$$

- durch Runden der Mantisse und Renormalisierung

$$1.** \cdot 10^{-2}$$

erhalten wir zwei Stellen in der Mantisse, über die wir nichts wissen

- diesen Effekt nennt man **Auslöschung**
- die Mantisse könnte also 1.00 aber auch 1.99 sein, was einem relativen Fehler von bis zu 100% entspricht
- deshalb sollten Auslösungen möglichst vermieden werden
- IEEE754 füllt die unbekannten Ziffern bei Auslösungen mit 0 auf
- die Multiplikation ist wesentlich unkritischer, da keine Auslösungen auftreten können
- zur Berechnung von

$$1.23 \cdot 10^4 \odot (-4.56 \cdot 10^7)$$

werden folgende Schritte nötig:

- Bestimmung des Vorzeichens, hier –
- Addition der Exponenten, $4 + 7 = 11$
- Multiplikation der Mantissen

$$1.23 \cdot 4.56 = 5.6088$$

- Rundung auf Mantissenlänge und eventuell Renormalisierung

$$-5.61 \cdot 10^{11}$$

- wegen der jeweils durchgeführten Rundung der Mantisse sind \oplus, \odot nicht assoziativ
- wir betrachten normierte Floating-Point-Zahlen mit dreistelliger Mantisse

$$x = 1.00 \cdot 10^2 = 100$$

$$y = 4.00 \cdot 10^{-1} = 0.4$$

- $x \oplus y$ ergibt

$$\begin{array}{r} 1.00 \quad 10^2 \\ \oplus \quad 0.004 \quad 10^2 \\ \hline = \quad 1.00 \quad 10^2 \end{array}$$

d.h. $x \oplus y = 100$, so dass

$$((x \oplus y) \oplus y) \oplus y = 100$$

- für $y \oplus y$ erhalten wir

$$\begin{array}{r} 4.00 \quad 10^{-1} \\ \oplus \quad 4.00 \quad 10^{-1} \\ \hline = \quad 8.00 \quad 10^{-1} \end{array}$$

d.h. $y \oplus y = 8.00 \cdot 10^{-1}$ und $(y \oplus y) \oplus y = 1.20 \cdot 10^0$, so dass

$$((y \oplus y) \oplus y) \oplus x = 1.01 \cdot 10^2$$

- deswegen sollte man immer zuerst (betrags-) kleinere Werte aufsummieren
- kommen wir nun zurück zu der Floating-Point-Implementierung \tilde{f}

$$\tilde{x} \xrightarrow{\tilde{f}_1} \tilde{x}_1 \xrightarrow{\tilde{f}_2} \tilde{x}_2 \xrightarrow{\tilde{f}_3} \dots \xrightarrow{\tilde{f}_p} \tilde{f}(\tilde{x})$$

unseres Ersatzproblems

$$\tilde{x} \xrightarrow{\hat{f}_1} \hat{x}_1 \xrightarrow{\hat{f}_2} \hat{x}_2 \xrightarrow{\hat{f}_3} \dots \xrightarrow{\hat{f}_p} \hat{f}(\tilde{x})$$

- um zu verstehen, welche Fehler durch die einzelnen \tilde{f}_i entstehen, vergleichen wir schrittweise die Zwischenergebnisse \hat{x}_i und \tilde{x}_i :

- für den ersten Schritt erhalten wir

$$\tilde{x}_1 - \hat{x}_1 = \tilde{f}_1(\tilde{x}) - \hat{f}_1(\tilde{x}),$$

- da beides Mal derselbe Input \tilde{x} benutzt wird, handelt es sich hierbei um den **lokalen Fehler**, der durch das Runden des Ergebnisses bei der Floating-Point-Operation \tilde{f}_1 entsteht

- im zweiten Schritt erhalten wir

$$\begin{aligned}\tilde{x}_2 - \hat{x}_2 &= \tilde{f}_2(\tilde{x}_1) - \hat{f}_2(\hat{x}_1) \\ &= \tilde{f}_2(\tilde{x}_1) - \hat{f}_2(\tilde{x}_1) + \hat{f}_2(\tilde{x}_1) - \hat{f}_2(\hat{x}_1)\end{aligned}$$

- $\tilde{f}_2(\tilde{x}_1) - \hat{f}_2(\tilde{x}_1)$ ist wieder der lokale Fehler durch \tilde{f}_2

- der Term

$$\hat{f}_2(\tilde{x}_1) - \hat{f}_2(\hat{x}_1) \quad (3.11)$$

beschreibt die **Fehlerverstärkung** des Fehlers $\tilde{x}_1 - \hat{x}_1$ aus dem ersten Schritt

- alle weiteren Schritte verlaufen analog zum zweiten
- in jedem Schritt wird also der vorherige Fehler verstärkt und ein neuer lokaler Fehler hinzugefügt
- deswegen bezeichnet man diesen Fehlertyp als **akkumulierten Rundungsfehler**
- zur quantitativen Analyse der Fehlerverstärkung (vergleiche (3.11)) können die Konditionsabschätzungen aus Kapitel 3.3.2 benutzt werden
- falls Rundungsfehler das Ergebnis stark verfälschen (d.h. der akkumulierte Fehler sehr groß ist) nennt man eine Implementierung **instabil**, ansonsten ist sie **stabil**

3.4 Zusammenfassung

- gut gestellte Probleme besitzen eine eindeutige Lösung, die stetig von den Eingabedaten abhängt
- schlecht gestellte Probleme werden durch gut gestellte Probleme angenähert
- bei der numerischen Umsetzung

$$\begin{array}{ccc} x & \xrightarrow{f} & y & \text{Originalproblem} \\ \downarrow & & & \\ x & \xrightarrow{\hat{f}} & \hat{y} & \text{Ersatzproblem} \\ \downarrow & & & \\ \tilde{x} & \xrightarrow{\tilde{f}} & \tilde{y} & \text{FP-Implementierung des Ersatzproblems} \end{array}$$

betrachten wir die folgende Zerlegung des Gesamtfelmers

$$\tilde{f}(\tilde{x}) - f(x) = \underbrace{\tilde{f}(\tilde{x}) - \hat{f}(\tilde{x})}_{\text{akk. Rundungsfehler}} + \underbrace{\hat{f}(\tilde{x}) - f(\tilde{x})}_{\text{Verfahrensfehler}} + \underbrace{f(\tilde{x}) - f(x)}_{\text{Eingabefehler}}$$

- man versucht, den absoluten oder relativen Gesamtfehler (oder ein gemischtes Kriterium) zu kontrollieren
- in der Praxis sollten alle drei Fehlerarten in der gleichen Größenordnung gehalten werden, zu große Genauigkeit bei einer Fehlerart allein reduziert den Gesamtfehler nicht nennenswert und erhöht eventuell den Aufwand beträchtlich
- die einzelnen Fehlerarten lassen sich wie folgt charakterisieren
- **Eingabefehler**

- beschreibt die Empfindlichkeit des Originalproblems f gegenüber Störungen in den Eingabedaten x
- ist bestimmt durch die Kondition von f und damit im Wesentlichen nicht veränderbar
- gut gestellte aber schlecht konditionierte Probleme können sich genauso verhalten wie schlecht gestellte Probleme

- **Verfahrensfehler**

- beurteilt die Qualität des Ersatzproblems
- wird in der Regel für jedes Ersatzproblem separat untersucht (z.B. Konvergenztheorie)
- kann durch ein "besseres" Ersatzproblem kleiner gemacht werden

- **Akkumulierter Rundungsfehler**

- kann durch eine "günstigere" Floating-Point-Implementierung des Ersatzproblems verändert werden (Summationsreihenfolgen, Vermeidung von Auslöschen etc.)
- stabile Implementierungen sind robust gegenüber Rundungsfehlereffekten

Kapitel 4

Lineare Gleichungssysteme – Direkte Löser II

4.1 Fehlerbetrachtung für lineare Gleichungssysteme

- betrachten $Ax = b$, A quadratisch, regulär

- Problem ist gut gestellt mit

$$f(A, b) = A^{-1}b = x$$

- betrachten zunächst die Kondition des Problems, d.h. wie empfindlich reagiert x auf Störungen in A, b

- das gestörte System sei $\tilde{A}\tilde{x} = \tilde{b}$

- setze $\Delta A = \tilde{A} - A$, $\Delta b = \tilde{b} - b$ und $\Delta x = \tilde{x} - x$

Satz 49. Sei $\|\cdot\|$ eine Norm auf \mathbb{R}^n , A regulär und $\|\Delta A\| < \frac{1}{\|A^{-1}\|}$, wobei die durch $\|\cdot\|$ induzierte Matrixnorm benutzt wird. Dann ist \tilde{A} regulär, und es gilt

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\varkappa(A)}{1 - \varkappa(A) \frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right) \quad (4.1)$$

mit $\varkappa(A) = \|A\| \|A^{-1}\|$.

Bemerkung 50.

◊ $\varkappa(A)$ heißt **Konditionszahl** der Matrix A und hat folgende Eigenschaften:

- ◊ \varkappa hängt von der verwendeten Norm ab
- ◊ ist $\alpha\|B\|' \leq \|B\| \leq \beta\|B\|'$ so gilt $\alpha^2\varkappa'(B) \leq \varkappa(B) \leq \beta^2\varkappa'(B)$, d.h. die Äquivalenz der Normen überträgt sich
- ◊ $\varkappa(A) \geq 1$, denn $1 = \|I\| = \|AA^{-1}\| \leq \|A\| \|A^{-1}\| = \varkappa(A)$
- ◊ $\varkappa(A) = \varkappa(A^{-1})$
- ◊ $\varkappa(AB) \leq \varkappa(A)\varkappa(B)$

- ◊ wir betrachten den Verstärkungsfaktor

$$\frac{\varkappa(A)}{1 - \varkappa(A) \frac{\|\Delta A\|}{\|A\|}} \quad (4.2)$$

von (4.1)

- ◊ mit

$$\frac{\|\Delta A\|}{\|A\|} < \frac{\|\Delta A\|}{\|A^{-1}\|} \cdot \frac{1}{\|A\| \|A^{-1}\|} = \frac{1}{\varkappa(A)} \Rightarrow \varkappa(A) \frac{\|\Delta A\|}{\|A\|} < 1$$

folgt, dass der Nenner von (4.2) größer 0 ist

- ◊ $\|\Delta A\|$ ist üblicherweise sehr viel kleiner als $\|A\|$, d.h. der Zähler von (4.2) dominiert, so dass der Eingangsfehler im wesentlichen mit $\varkappa(A)$ verstärkt wird

- $\varkappa(A)$ sollte also so klein wie möglich sein
- Problem: $\varkappa(A) = \|A\| \|A^{-1}\|$ ist wegen $\|A^{-1}\|$ in der Regel nicht berechenbar
- es gibt numerische Verfahren, die $\varkappa(A)$ mit vertretbarem Aufwand abschätzen (z.B. LAPACK Expert Driver Routinen)
- generell sollte man vor der Berechnung versuchen, die Kondition des Problems zu verbessern
- betrachte $Ax = b$ und U, V regulär

$$Ax = b \iff \underbrace{UAV}_{\tilde{A}} \underbrace{V^{-1}x}_{\tilde{x}} = \underbrace{Ub}_{\tilde{b}}$$

- $\tilde{A}\tilde{x} = \tilde{b}$ heißt **vorkonditioniertes System**
- löse $\tilde{A}\tilde{x} = \tilde{b}$ mit $\varkappa(\tilde{A})$ statt $Ax = b$ mit $\varkappa(A)$
- optimal (aber nicht praktikabel) ist z. B.

$$U = A^{-1}, V = I \Rightarrow \tilde{A} = I \Rightarrow \varkappa(\tilde{A}) = 1$$

- Praxis: suche für U bzw. V Näherungen für A^{-1} , die billig zu berechnen sind
- einfachste Ansätze:

- skaliere Zeilen

$$U = \begin{pmatrix} u_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & u_n \end{pmatrix}, \quad V = I$$

- skaliere Spalten

$$U = I, \quad V = \begin{pmatrix} v_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & v_n \end{pmatrix}$$

- skaliere beides

$$U = \begin{pmatrix} u_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & u_n \end{pmatrix}, \quad V = \begin{pmatrix} v_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & v_n \end{pmatrix}$$

- "vernünftige" Skalierung:
 - alle Gleichungen sollten Koeffizienten gleicher Größenordnung haben \Rightarrow Zeilenvektoren sollten gleiche Norm haben
 - alle Unbekannten sollten mit gleichem Gewicht auftauchen \Rightarrow Spalten gleicher Norm
- Spalten bzw. Zeilen auf gleiche Norm bringen nennt man **Äquilibrieren**
- äquilibriert man Spalten und Zeilen gleichzeitig, so führt dies auf ein nichtlineares Gleichungssystem mit $2n - 2$ Unbekannten, das schwieriger zu lösen ist als das ursprüngliche lineare Gleichungssystem
- deshalb werden in der Regel entweder nur Zeilen oder nur Spalten äquilibriert

Beispiel 51.

- wir betrachten ein Gleichungssystem mit Matrix

$$A = \begin{pmatrix} 1000 & 0 \\ 1 & 1 \end{pmatrix}$$

und bestimmen die Konditionszahl $\kappa_\infty(A)$ bezüglich der von der Vektornorm $\|\cdot\|_\infty$ induzierten Matrixnorm, der **Zeilensummennorm**

- mit $\|A\|_\infty = 1000$ und

$$A^{-1} = \begin{pmatrix} \frac{1}{1000} & 0 \\ -\frac{1}{1000} & 1 \end{pmatrix}$$

ist $\|A^{-1}\|_\infty = \frac{1001}{1000}$, so dass

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = 1001$$

- jetzt skalieren wir die i -te Zeile z_i der Matrix A mit dem Faktor

$$u_i = \frac{1}{\|z_i\|_\infty},$$

d.h. wir betrachten das vorkonditionierte System mit Matrix

$$\tilde{A} = UA, \quad U = \begin{pmatrix} \frac{1}{1000} & 0 \\ 0 & 1 \end{pmatrix}$$

- mit

$$\tilde{A} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad \tilde{A}^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

folgt $\|\tilde{A}\|_\infty = \|\tilde{A}^{-1}\|_\infty = 2$ und somit

$$\kappa_\infty(\tilde{A}) = \|\tilde{A}\|_\infty \|\tilde{A}^{-1}\|_\infty = 4$$

- Satz 49 liefert eine reine Konditionsaussage für das exakte Problem, keine Fehlerabschätzung für ein numerisches Verfahren
- ein stabiler Algorithmus verhält sich analog, so dass die Abschätzung als **a-priori Fehlerschätzer** dienen kann
- bei **a-posteriori** Methoden wird zunächst eine Näherungslösung bestimmt und dann getestet, ob sie akzeptabel ist
- dies liefert in der Regel genauere Aussagen als a-priori Methoden
- ein Beispiel dafür ist im nächsten Satz angegeben
- wir betrachten dabei das folgende Szenario:
 - A und b sind nur bis auf Störungen B bzw. c bekannt
 - $Ax = b$ ist also mit einer “gewissen Unschärfe” (z.B. Rundungsfehler oder Messungenauigkeiten) gegeben
 - in dieser Lage ist es nicht sinnvoll, nach der exakten Lösung x zu fragen
 - stattdessen: Plausibilitätsprüfung einer vorgelegten Näherungslösung \tilde{x}
 - ist \tilde{x} Lösung eines Systems “innerhalb der Unschärfe”, dann kann man \tilde{x} als Lösung akzeptieren

Satz 52 (Prager, Oettli). Für $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$ sei

$$|x| = (|x_1|, \dots, |x_n|)^T, \quad |A| = (|a_{ij}|)_{i,j}.$$

Ist $\det(A) \neq 0$ und \tilde{x} eine Näherungslösung von $Ax = b$ mit **Residuum** $\tilde{r} = b - A\tilde{x}$ sowie

$$B \in \mathbb{R}^{n \times n}, c \in \mathbb{R}^n, \quad b_{ij} \geq 0, \quad c_i \geq 0 \quad \forall i, j$$

mit

$$|\tilde{r}| \leq B|\tilde{x}| + c.$$

Dann gibt es $|\Delta A| \leq B$ und $|\Delta b| \leq c$ so dass \tilde{x} exakte Lösung von

$$(A + \Delta A)\tilde{x} = b + \Delta b$$

ist. Die Ungleichheitszeichen sind jeweils komponentenweise anzuwenden.

Bemerkung 53.

- ◊ kennt man B, c so weiß man, dass \tilde{x} die exakte Lösung eines gestörten Ausgangsproblems ist
- ◊ über B, c weiß man auch, wie weit das gestörte Problem höchstens vom Ausgangsproblem entfernt ist
- ◊ sind B, c (komponentenweise) klein, dann ist \tilde{x} eine gute Näherung von x
- ◊ dies ist ein Beispiel für eine so genannte **Rückwärtsanalyse**
- ◊ Vorwärtsanalyse: Man betrachtet die Lösung x des Systems $Ax = b$
- ◊ Rückwärtsanalyse: Man betrachtet ein System $(A + B)\tilde{x} = b + c$ zur Lösung \tilde{x}

Beispiel 54.

- wir betrachten das lineare Gleichungssystem $Ax = b$,

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} -8 \\ 19 \end{pmatrix},$$

d.h. die exakte Lösung ist $x = (1, 10)^T$

- die Koeffizienten von A und b seien gestört mit

$$|\Delta A| \leq \frac{1}{10} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad |\Delta b| \leq \frac{1}{10} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

- wir nehmen an, dass wir mit zwei verschiedenen Verfahren die Näherungen

$$\tilde{x}_1 = \begin{pmatrix} 1 \\ 9 \end{pmatrix}, \quad \tilde{x}_2 = \begin{pmatrix} 0 \\ 10 \end{pmatrix}$$

berechnet haben

- wir testen mit Prager-Oettli, ob wir die Näherungen akzeptieren können
- \tilde{x}_1 :

- es ist

$$\tilde{r} = b - A\tilde{x}_1 = \begin{pmatrix} -8 \\ 19 \end{pmatrix} - \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 9 \end{pmatrix} = \begin{pmatrix} -8 \\ 19 \end{pmatrix} - \begin{pmatrix} -7 \\ 17 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

also

$$|\tilde{r}| = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- nach Voraussetzung gilt

$$B = \frac{1}{10} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad c = \frac{1}{10} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

und deshalb

$$B|\tilde{x}_1| + c = \frac{1}{10} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 9 \end{pmatrix} + \frac{1}{10} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{10} \begin{pmatrix} 12 \\ 20 \end{pmatrix} = \begin{pmatrix} 1.2 \\ 2 \end{pmatrix}$$

- damit erhalten wir

$$|\tilde{r}| = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \leq \begin{pmatrix} 1.2 \\ 2 \end{pmatrix} = B|\tilde{x}_1| + c,$$

d.h. \tilde{x}_1 kann akzeptiert werden, es gibt also ein $|\Delta A| \leq B$ und $|\Delta b| \leq c$ mit

$$(A + \Delta A)\tilde{x}_1 = b + \Delta b$$

- \tilde{x}_2 :

- für \tilde{x}_2 erhalten wir analog

$$|\tilde{r}| = |b - A\tilde{x}_2| = \left| \begin{pmatrix} -8 \\ 19 \end{pmatrix} - \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 10 \end{pmatrix} \right| = \left| \begin{pmatrix} 2 \\ -1 \end{pmatrix} \right| = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

bzw.

$$B|\tilde{x}_2| + c = \frac{1}{10} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 10 \end{pmatrix} + \frac{1}{10} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.1 \\ 2.1 \end{pmatrix}$$

- damit ist $|\tilde{r}| \leq B|\tilde{x}_2| + c$ nicht erfüllt und \tilde{x}_2 kann nicht akzeptiert werden

- in der Praxis wird das Resultat in der Regel wie folgt benutzt

- wir nehmen an, dass die Störungen ähnlich wie A, b aussehen und setzen

$$B = \varepsilon|A|, \quad c = \varepsilon|b|$$

mit ε unbekannt

- berechne \tilde{x} mit einem numerischen Verfahren und damit \tilde{r}
- bestimme jetzt das kleinste ε mit

$$|\tilde{r}| \leq B|\tilde{x}| + c = \varepsilon \underbrace{(|A| |\tilde{x}| + |b|)}_s$$

d.h.

$$\varepsilon = \max_i \frac{|\tilde{r}_i|}{\tilde{s}_i}$$

unter der Voraussetzung, dass $\tilde{s}_i \neq 0$

- ist dies möglich, dann gilt

$$|\tilde{r}| \leq \varepsilon|A| |\tilde{x}| + \varepsilon|b|$$

und nach Satz 52 ist \tilde{x} die exakte Lösung von

$$(A + \Delta A)\tilde{x} = b + \Delta b$$

mit

$$|\Delta A| \leq \varepsilon|A|, \quad |\Delta b| \leq \varepsilon|b|,$$

d.h. ε liefert eine obere Abschätzung für die Störung der Eingabedaten

- damit stellt die Berechnung von ε eine Art "numerische Rückwärtsanalyse" dar

Beispiel 55.

- wir betrachten

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 2\delta & 2\delta \\ 1 & 2\delta & -\delta \end{pmatrix}, \quad b = \begin{pmatrix} 4+3\delta \\ 6\delta \\ 2\delta \end{pmatrix} \Rightarrow x = \begin{pmatrix} \delta \\ 1 \\ 1 \end{pmatrix}$$

- wir bestimmen wie oben beschrieben ε nach Prager-Oettli bei einfacher Genauigkeit, wenn \tilde{x} mit LU-Zerlegung berechnet wird

δ	ε	$\kappa_2(A)$	$\kappa_\infty(A)$
10^{-4}	0.000068	$\approx 2.1/\delta$	$\approx 3.6/\delta$
10^{-5}	0.0011		
10^{-6}	0.013	\vdots	\vdots
10^{-7}	0.052		

Zusammenfassung.

- $\kappa(A)$ bestimmt die Kondition von $Ax = b$
- Vorkonditionierung durch Äquilibrieren bzw. approximative Inverse
- a-posteriori Fehlerkontrolle über Residuum

4.2 LU-Zerlegung mit Pivot-Strategie

- die LU-Zerlegung transformiert $Ax = b$ in $Ux = y$ mit $U = F_{n-1}P_{n-1} \dots F_1P_1A$, d.h. ein lineares Gleichungssystem in ein anderes
- $Ux = y$ einfach lösbar (Rückwärtseinsetzen)
- wie verändert sich das Problem dadurch, d.h., wie verändert sich die Kondition und damit die Fehlerverstärkung?
- vergleichen jetzt $\kappa(A)$ mit $\kappa(U)$

Beispiel 56.

► betrachte $A = \begin{pmatrix} 1 & a \\ a & 1 \end{pmatrix}$, $a \geq 2$ \Rightarrow $A^{-1} = \frac{1}{a^2 - 1} \begin{pmatrix} -1 & a \\ a & -1 \end{pmatrix}$

► benutzen κ_∞

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = (1+a) \frac{1+a}{a^2-1} = \frac{a+1}{a-1} \xrightarrow{a \rightarrow \infty} 1,$$

d.h. gute Kondition für großes a

► durch Gaußelimination erhalten wir

$$U = \begin{pmatrix} 1 & a \\ 0 & 1-a^2 \end{pmatrix}, \quad U^{-1} = \frac{1}{a^2-1} \begin{pmatrix} a^2-1 & a \\ 0 & -1 \end{pmatrix}$$

so dass

$$\begin{aligned} \|U\|_\infty &= \max(1+a, |1-a^2|) \stackrel{a \geq 2}{\cong} a^2 - 1, \\ \|U^{-1}\|_\infty &= \max\left(\frac{a^2-1+a}{a^2-1}, \frac{1}{a^2-1}\right) = \frac{a^2+a-1}{a^2-1} \end{aligned}$$

und damit

$$\kappa_\infty(U) = a^2 + a - 1 \xrightarrow{a \rightarrow \infty} \infty$$

- für wachsendes a wird $\kappa_\infty(A)$ immer besser, $\kappa_\infty(U)$ immer schlechter, d.h. die Elimination erhöht die Fehleranfälligkeit
- kann man das verhindern oder mindestens abmildern?

Beispiel 57.

- A wie oben, aber tausche Zeilen

$$\tilde{A} = \begin{pmatrix} a & 1 \\ 1 & a \end{pmatrix}, \quad \tilde{U} = \begin{pmatrix} a & 1 \\ 0 & a - \frac{1}{a} \end{pmatrix},$$

$$\kappa_\infty(\tilde{A}) = \frac{a+1}{a-1}, \quad \kappa_\infty(\tilde{U}) = \frac{a+1 - \frac{1}{a}}{a-1}$$

- damit gilt $\kappa_\infty(\tilde{A}) \xrightarrow{a \rightarrow \infty} 1, \quad \kappa_\infty(\tilde{U}) \xrightarrow{a \rightarrow \infty} 1$

Bemerkung 58.

- ◊ mit $a \geq 2$ steht in der ersten Spalte von \tilde{A} das betragsgrößte Element auf der Diagonalen
- ◊ \tilde{U} hat für große a eine sehr viel bessere Kondition κ_∞ als U
- wir betrachten nun den allgemeinen Fall (siehe (2.7), (2.8), (2.9)):

- $U = L^{-1}PA, \quad L^{-1} = F_{n-1}P_{n-1} \cdot \dots \cdot F_2P_2F_1P_2 \cdot \dots \cdot P_{n-1}, \quad P = P_{n-1} \cdot \dots \cdot P_1$
- mit $\kappa_\infty(BC) \leq \kappa_\infty(B)\kappa_\infty(C)$ folgt

$$\kappa_\infty(U) \leq \kappa_\infty(L^{-1})\kappa_\infty(P)\kappa_\infty(A), \quad (4.3)$$

$$\kappa_\infty(L^{-1}) \leq \kappa_\infty(F_{n-1})\kappa_\infty(P_{n-1}) \dots \kappa_\infty(F_2)\kappa_\infty(P_2)\kappa_\infty(F_1)\kappa_\infty(P_2) \dots \kappa_\infty(P_{n-1}), \quad (4.4)$$

$$\kappa_\infty(P) \leq \kappa_\infty(P_{n-1}) \dots \kappa_\infty(P_1) \quad (4.5)$$

- $\kappa_\infty(U)$ kann nicht stark anwachsen, wenn $\kappa_\infty(F_k), \kappa_\infty(P_k)$ klein sind
- wegen $\kappa_\infty(C) = \|C\|_\infty \|C^{-1}\|_\infty, \|\cdot\|_\infty$ Zeilensummennorm, folgt
 - für die Permutationsmatrizen P_k (siehe (2.6)) wegen $P_k = P_k^{-1}$,

$$\|P_k\|_\infty = \|P_k^{-1}\|_\infty = 1$$

und somit

$$\kappa_\infty(P_k) = 1$$

- für die Frobeniusmatrizen F_k (siehe (2.4)) und ihre Inversen (2.5)

$$\|F_k\|_\infty = \|F_k^{-1}\|_\infty = 1 + \max_{i>k} |l_{ik}|$$

und somit

$$\kappa_\infty(F_k) = \left(1 + \max_{i>k} |l_{ik}|\right)^2, \quad (4.6)$$

$$l_{ik} = \frac{\tilde{a}_{ik}^{(k-1)}}{\tilde{a}_{kk}^{(k-1)}}, \quad \tilde{A}^{(k-1)} = P_k F_{k-1} P_{k-1} \cdot \dots \cdot F_1 P_1 A,$$

wobei $\tilde{A}^{(k-1)}$ die Systemmatrix *nach* dem k -ten Zeilentausch und *vor* der Elimination der k -ten Spalte ist

- $\kappa_\infty(P_k)$ ist optimal
- $\kappa_\infty(F_k)$ wird dann klein, wenn $\max_{i>k} |l_{ik}|$ klein ist
- die Koeffizienten

$$l_{ik} = \frac{\tilde{a}_{ik}^{(k-1)}}{\tilde{a}_{kk}^{(k-1)}}$$

werden durch den zuletzt durchgeföhrten Zeilentausch (d.h. die Permutation P_k) beeinflusst

- bisher haben wir Zeilen immer so getauscht, dass anschließend *irgendein* Element ungleich 0 auf der Diagonale steht
- jetzt tauschen wir die Zeilen so, dass das *betragssgrößte* Element der Spalte auf die Diagonale kommt, d.h. nach dem Zeilentausch gilt

$$|\tilde{a}_{kk}^{(k-1)}| \geq |\tilde{a}_{ik}^{(k-1)}| \quad \forall i > k$$

- damit ist

$$|\tilde{l}_{ik}| \leq 1$$

und mit (4.6)

$$\kappa_\infty(F_k) \leq 4$$

- zusammen erhalten wir aus (4.3),(4.4),(4.5)

$$\kappa_\infty(U) \leq \kappa_\infty(L^{-1}) \kappa_\infty(P) \kappa_\infty(A) \leq \kappa_\infty(F_{n-1}) \dots \kappa_\infty(F_1) \kappa_\infty(A),$$

also

$$\kappa_\infty(U) \leq 4^{n-1} \kappa_\infty(A) \tag{4.7}$$

- dieses Verfahren nennt sich **Spalten-Pivot-Suche**

- *Praktische Durchführung:*

- $PA = LU$, mit anderem P, L, U
- Suchschleife zur Bestimmung der P_k
- sonst keine Änderung nötig

Beispiel 59.

► wir betrachten nochmal die Matrix aus Beispiel 18

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

► 1. Spalte: tausche z_1 mit $z_3 \rightarrow j_1 = 3$

$$\begin{array}{cccc} 4 & 4 & 0 & 0 \\ 2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 3 & 1 & 0 \end{array} \Rightarrow \begin{array}{c|ccc} 4 & 4 & 0 & 0 \\ \hline \frac{1}{2} & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2} & 1 & 1 & 0 \end{array}$$

► 2. Spalte: kein Tausch $\rightarrow j_2 = 2$

$$\left| \begin{array}{cccc} 4 & 4 & 0 & 0 \\ \frac{1}{2} & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2} & 1 & 1 & 0 \end{array} \right| \Rightarrow \left| \begin{array}{cccc} 4 & 4 & 0 & 0 \\ \frac{1}{2} & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2} & -1 & 3 & 0 \end{array} \right|$$

► 3. Spalte: tausche z_3 mit $z_4 \rightarrow j_3 = 4$

$$\left| \begin{array}{cccc} 4 & 4 & 0 & 0 \\ \frac{1}{2} & -1 & 2 & 0 \\ \frac{1}{2} & -1 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right| \Rightarrow \left| \begin{array}{cccc} 4 & 4 & 0 & 0 \\ \frac{1}{2} & -1 & 2 & 0 \\ \frac{1}{2} & -1 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right|$$

► damit haben wir:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 4 & 4 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad p = \begin{pmatrix} 3 \\ 2 \\ 4 \end{pmatrix}$$

- betrachten wir nun die Stabilität des Gauß-Verfahrens mit Spaltenpivotisierung etwas genauer
- Annahme: exakte Arithmetik, A und b komponentenweise bekannt mit Maschinengenauigkeit ε_M , also mit $\Delta A = A - \tilde{A}$, $\Delta b = b - \tilde{b}$

$$\|\Delta A\|_\infty / \|A\|_\infty \leq \varepsilon_M, \quad \|\Delta b\|_\infty / \|b\|_\infty \leq \varepsilon_M$$

- (höchst optimistische) Annahme: exakte Lösung von $Ly = b$ komponentenweise bekannt bis auf Maschinengenauigkeit, also $\|\Delta y\|_\infty / \|y\|_\infty \leq \varepsilon_M$, $\Delta y = y - \tilde{y}$
- Störungssatz 49 angewendet auf $Ux = y$:

$$\|\Delta x\|_\infty = \|x - \tilde{x}\|_\infty \leq \frac{\kappa_\infty(U)}{1 - \kappa(U)_\infty \varepsilon_M} 2\varepsilon_M$$

- oben gesehen: für $\kappa_\infty(U)\varepsilon_M \ll 1$ gilt

$$\frac{\kappa_\infty(U)}{1 - \kappa_\infty(U)\varepsilon_M} \approx \kappa_\infty(U)$$

- bei exakter Arithmetik erhält man aus (4.7) die ungefähre obere Schranke

$$\|x - \tilde{x}\|_\infty \lesssim c \cdot 4^{n-1} \kappa_\infty(A) \varepsilon_M, \quad c \approx 2$$

- bei inexakter Arithmetik und realistischer Störung von y : Fehler i. A. eher größer
- Konsequenz dieser Überlegungen: das Gauß-Verfahren mit Spaltenpivotisierung ist zwar stabil im Sinn der Definition, aber scheint in der Praxis für große n unbrauchbar.

Beispiel 60.

- sei M die Matrix aus dem LGS (1.4) zum Widerstandsnetzwerk aus Beispiel 3 in Kapitel 1
- es gilt $n = 10$, $\kappa_\infty(M) \approx 1438$
- nach Formel (4.7): $\kappa_\infty(U) \leq 4^9 \cdot 1438 \approx 3.77 \cdot 10^8$
- aber: man erhält bei Spaltenpivotisierung $\kappa_\infty(U) \approx 2388 < 2 \kappa_\infty(A)$
- Abschätzung (4.7) viel zu pessimistisch!
- $\kappa_\infty(L) \approx 7.70$

- früheste Untersuchungen zur Stabilität des Gauß-Verfahrens stammen von Hotelling (1943):

- A sei regulär mit $|a_{ij}| \leq 1$, $a_{ii} = 1$, $|b_{ij}| \leq 1$
- bei inexakter Arithmetik mit Maschinengenauigkeit ε_M gilt für den relativen Fehler der LU-Zerlegung ohne Pivotisierung:

$$\|\Delta x\|_\infty \leq \hat{c} 4^{n-1} \varepsilon_M$$

- \hat{c} hängt von A ab
- Grund für die Einschränkungen an A und b : Hotelling war Statistiker und betrachtete ausschließlich Kovarianzmatrizen
- in der Praxis: Gauß-Verfahren mit Spaltenpivotisierung höchst erfolgreich
- Widerspruch zwischen Theorie und Praxis wurde schon früh bemerkt
- Alan Turing 1947: für beliebiges reguläres A und Gauß-Verfahren mit Spaltenpivotisierung gilt

$$\|\Delta x\|_\infty \leq \tilde{c} \kappa_\infty^2(A) 2^{n-1} \varepsilon_M$$

- genauere Untersuchung durch Wilkinson in den 1960er Jahren:

- **Rückwärtsanalyse**: die gestörte Lösung \tilde{x} entspricht der exakten Lösung (bei exakter Arithmetik) des Systems $(A + \Delta A)\tilde{x} = b$
- im Falle der Spaltenpivotisierung erhält man

$$\|\Delta A\|_\infty \leq p(n)g(A)\|A\|_\infty \varepsilon_M \quad (4.8)$$

- $p(n)$: kubisches Polynom; repräsentiert den akkumulierten Rundungsfehler der $\mathcal{O}(n^3)$ Operationen der LU-Zerlegung
- für $g(A)$ gilt

$$g(A) = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|}, \quad (4.9)$$

wobei $A^{(k)} = (a_{ij}^{(k)})_{ij}$ die Matrix im k -ten Eliminierungsschritt bezeichnet

- damit entscheidend für die Stabilität der LU-Zerlegung: Wachstum von $g(A)$
- bei Spaltenpivotisierung gilt $g(A) \leq 2^{n-1}$, diese obere Schranke wird angenommen
- Abschätzung für $\|\Delta x\|_\infty$ aus dem Störungssatz 49 und Formel (4.8):

$$\begin{aligned} \frac{\|\Delta x\|_\infty}{\|x\|_\infty} &\leq \frac{\kappa_\infty(A)}{1 - \kappa_\infty(A) \frac{\|\Delta A\|_\infty}{\|A\|_\infty}} \left(\frac{\|\Delta A\|_\infty}{\|A\|_\infty} + \frac{\|\Delta b\|_\infty}{\|b\|_\infty} \right) \\ &\stackrel{(4.8)}{\leq} \frac{\kappa_\infty(A)}{1 - \kappa_\infty(A) p(n) g(A) \varepsilon_M} (p(n) g(A) \varepsilon_M + \varepsilon_M) \end{aligned}$$

- für $n > 1$ gilt $p(n) < cn^3$

• Annahme: $\kappa_\infty(A)p(n)g(A)\varepsilon_M < 1/2$; $cn^3g(A) > 1$

- man erhält dann

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq 4c n^3 \kappa_\infty(A) g(A) \varepsilon_M \quad (4.10)$$

- van Veldhuizen 1977: (4.8) gilt für jede Pivotisierungsstrategie (also auch für Nicht-Pivotisierung)
- $g(A)$ leicht während der LU-Zerlegung berechenbar und einfach zu überwachen
- Alternative zur Spaltenpivotisierung: vollständiger Pivot-Suche
 - suche betragsgrößtes Element nicht nur in der Spalte, sondern auf der ganzen noch nicht eliminierten unteren Dreiecksmatrix
 - tausche dann Zeilen *und* gegebenenfalls Spalten
 - bei vollständiger Pivotisierung: $g(A) \sim c n^{1/2} n^{1/4} \log n$ (Wilkinson)
 - aber: vollständige Pivotisierung erfordert zusätzlichen Programmieraufwand wegen Spaltentausch
 - der quadratische Suchaufwand ist für große Systeme sehr hoch
 - vollständiger Pivot wird selten benutzt
- in der Praxis verwendet man Spaltenpivotisierung
- warum ist das Gauß-Verfahren mit Spaltenpivotisierung in der Praxis so erfolgreich, obwohl $g(A) = 2^{n-1}$ auftreten kann?
- viele praxisrelevante Matrizen sind spd, z. B. die Matrix zur Simulation der Kurbel aus Beispiel 4, Kapitel 1
- allgemein lässt sich zeigen: ist A spd, dann ist Pivotisierung überflüssig, und es gilt $g(A) \leq 1$.
- ähnliche Aussagen gelten für andere spezielle Matrizenklassen
- empirisch: $p(n)$ wächst erheblich langsamer als n^3 , man vermutet $p(n) = \mathcal{O}(n^{3/2})$ (z. B. Higham 2011)
- bis hierher: **worst case analysis**
- für allgemeines reguläres A : **average case analysis**
 - Wie groß ist $g(A)$ für irgendeine reguläre, zufällige Matrix A ?
 - erforderlich: Methoden der Stochastik
 - Experimente legen $\mathbb{E}(g(A)) = \mathcal{O}(n^{3/2})$ für Spaltenpivotisierung nahe (Trefethen & Schreiber 1990) für A mit unkorrelierten, normalverteilten Einträgen mit Mittelwert 0 und Standardabweichung 1
 - Gegenstand aktueller Forschung
 - Konsequenz: es gibt reguläre Matrizen mit $g(A)$ groß, aber in der Regel wächst $g(A)$ moderat

Zusammenfassung.

- benutze Spalten-Pivotsuche, um Fehleranfälligkeit zu reduzieren
- Spalten-Pivotsuche ist in der Praxis stabil
- einfache Implementierung

4.3 Orthogonale Transformationen

4.3.1 Grundlagen

- betrachten k -ten Schritt des LU-Verfahrens $A^{(k)} = F_k P_k A^{(k-1)}$ mit

$$\begin{pmatrix} a_{kk}^{(k-1)} \\ a_{k+1k}^{(k-1)} \\ \vdots \\ a_{nk}^{(k-1)} \end{pmatrix} \xrightarrow{F_k P_k} \begin{pmatrix} a_{kk}^{(k)} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

d.h., die Spalte ab der Diagonale abwärts wird auf ein Vielfaches des ersten Einheitsvektors transformiert

- damit ist

$$\varkappa(A^{(k)}) \leq \varkappa(T_k) \varkappa(A^{(k-1)}), \quad T_k = F_k P_k$$

bzw.

$$\varkappa(U) \leq \varkappa(T_{n-1}) \dots \varkappa(T_1) \varkappa(A)$$

- eine optimale Abschätzung für $\varkappa(U)$ erhalten wir, wenn $\varkappa(T_k) = 1$ für alle k
- $T_k = F_k P_k$, P_k Permutationsmatrix,

$$F_k = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{k+1k} & \ddots & & \\ & & \vdots & & \ddots & \\ & & -l_{nk} & & & 1 \end{pmatrix}, \quad l_{ik} = \frac{\tilde{a}_{ik}^{(k-1)}}{\tilde{a}_{kk}^{(k-1)}}$$

- für \varkappa_∞ erhält man analog zu (4.6)

$$\varkappa_\infty(T_k) = (1 + \max_{i>k} |l_{ik}|)^2$$

- damit ist

$$\varkappa_\infty(T_k) = 1 \iff l_{ik} = 0 \quad \forall i > k,$$

d.h. $\varkappa_\infty(T_k) = 1$ nur wenn nichts zu eliminieren ist, sonst ist $\varkappa_\infty(T_k) > 1$

- wie kann man das verbessern:

- andere Norm: es gibt keine induzierte Norm, so dass $\varkappa(T_k) = 1$, falls mindestens ein $l_{ik} \neq 0$
- anderes T_k (mit geeigneter Norm)
- wir suchen also $Q \in \mathbb{R}^{n \times n}$ mit folgenden Eigenschaften:

- (1) Q regulär
- (2) $\|Qx\| = \|x\| \quad \forall x \in \mathbb{R}^n$ weshalb auch

$$\|x\| = \|QQ^{-1}x\| = \|Q^{-1}x\| \quad \forall x \in \mathbb{R}^n$$

gelten muss, so dass $\|Q\| = 1$, $\|Q^{-1}\| = 1$ und damit $\varkappa(Q) = 1$

(3) mit Q muss man Spalten eliminieren können, d.h.

$$Q \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad a \in \mathbb{R}$$

- aus der Geometrie kennt man solche linearen Abbildungen, nämlich Drehungen und Spiegelungen:
 - (1) sie sind regulär (Inverse ist gegeben durch Drehung um negativen Winkel bzw. nochmaliges Spiegeln)
 - (2) die euklidische Länge (d.h. die Vektornorm $\|\cdot\|_2$) der gedrehten/gespiegelten Vektoren ändert sich nicht
 - (3) Eliminationseigenschaft: weisen wir später nach
- Dreh- und Spiegelmatrizen Q sind Spezialfall von orthonormalen Matrizen

Definition 61. $Q \in \mathbb{R}^{n \times n}$ heißt **orthonormal**, falls die Spaltenvektoren q_j paarweise senkrecht stehen und Länge 1 haben, d.h.

$$\langle q_i, q_j \rangle = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{sonst} \end{cases}.$$

Satz 62. Sind $Q, \tilde{Q} \in \mathbb{R}^{n \times n}$ orthonormal, dann gilt:

- (1) $QQ^T = Q^TQ = I$, d.h. $Q^{-1} = Q^T$
- (2) $\varkappa_2(Q) = 1$
- (3) $Q\tilde{Q}$ ist orthonormal

Beweis.

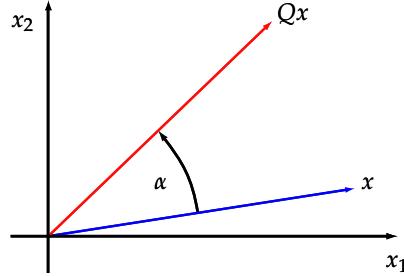
- (1) klar
- (2) $\|Qx\|_2^2 = \langle Qx, Qx \rangle = x^T Q^T Q x \stackrel{(1)}{=} x^T x = \|x\|_2^2$, analog für $Q^{-1} = Q^T$
- (3) trivial

□

Beispiel 63.

- ▶ eine Drehung im \mathbb{R}^2 um den Winkel α gegen den Uhrzeigersinn ist gegeben durch

$$Q = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}$$

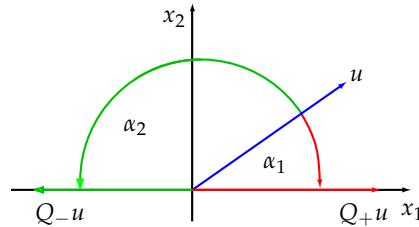


- ▶ für $u \in \mathbb{R}^2$ beliebig gibt es Drehungen Q_+, Q_- mit

$$Q_+ u = \|u\|_2 e_1, \quad Q_- u = -\|u\|_2 e_1$$

wobei für $u = (u_1, u_2)^T$ gilt

$$Q_+ = \frac{1}{\|u\|_2} \begin{pmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{pmatrix} = -Q_-$$



- ▶ damit können Q_{\pm} Spalten eliminieren

Zusammenfassung.

- Spaltentransformation auf Vielfaches des Einheitsvektors ist Elementaroperation bei der Matrixzerlegung
- für orthonormales Q ist $\kappa_2(Q) = 1$ (minimal) und damit $\kappa_2(U) \leq \kappa_2(A)$
- Drehungen und Spiegelungen sind orthonormal
- im \mathbb{R}^2 können Spalten mit Drehungen eliminiert werden

4.3.2 Verfahren von Givens

- benutzen Drehungen zum Eliminieren der Spalten
- nach dem letzten Kapitel gilt in \mathbb{R}^2 :

- ist $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ gegeben, $\|u\|_2 \neq 0$, so gilt

$$Qu = \pm \|u\|_2 e_1$$

mit

$$Q = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = \pm \frac{u_1}{\|u\|_2}, \quad s = \pm \frac{u_2}{\|u\|_2}$$

- für $\|u\|_2 = 0$ ist $u = (0, 0)^T$ so dass mit $Q = I$ folgt

$$Qu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \pm \|u\|_2 e_1$$

- die Verallgemeinerung für $u = (u_1, \dots, u_{i-1}, u_i, u_{i+1}, \dots, u_n)^T \in \mathbb{R}^n$ sieht wie folgt aus:

- sei $w = \sqrt{u_1^2 + u_i^2}$, $i > 1$
- setze für $w = 0$, $Q_{i1} = I$ bzw. für $w \neq 0$

$$Q_{i1} = \begin{pmatrix} c & & & & & s & & & \\ & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & 1 & & & & & \\ & & & & c & & & & \\ & & & & & 1 & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \end{pmatrix}, \quad c = \pm \frac{u_1}{w}, \quad s = \pm \frac{u_i}{w}$$

- dann ist Q_{i1} orthonormal und

$$Q_{i1}u = \begin{pmatrix} \pm w \\ u_2 \\ \vdots \\ u_{i-1} \\ 0 \\ u_{i+1} \\ \vdots \\ u_n \end{pmatrix}$$

d.h. u_i wird eliminiert und außer u_1, u_i wird nichts verändert

- benutzt man das positive Vorzeichen, so liefert die Anwendung auf die erste Spalte $s_1 =$

$(a_{11}, \dots, a_{n1})^T$ von A

$$s_1^{(2)} = Q_{21}s_1 = \begin{pmatrix} \sqrt{a_{11}^2 + a_{21}^2} \\ 0 \\ a_{31} \\ \vdots \\ a_{n1} \end{pmatrix}$$

- analog kann man die restlichen Elemente der Spalte eliminieren:

$$s_1^{(3)} = Q_{31}s_1^{(2)} = \begin{pmatrix} \sqrt{a_{11}^2 + a_{21}^2 + a_{31}^2} \\ 0 \\ 0 \\ a_{41} \\ \vdots \\ a_{n1} \end{pmatrix} \quad \dots \quad s_1^{(n)} = \underbrace{Q_{n1} \cdot \dots \cdot Q_{21}}_{=:Q_1} s_1 = \begin{pmatrix} \|s_1\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- insgesamt:

- $Q_1 = Q_{n1} \cdot \dots \cdot Q_{21}$ ist orthonormal
- $Q_1 s_1 = \|s_1\| e_1$ und damit

$$Q_1 A = \begin{pmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{pmatrix}$$

- Wiederholung desselben Algorithmus für die Spalten 2 bis $n - 1$ liefert

$$Q_{n-1} \cdot \dots \cdot Q_1 A = U$$

mit Q_i orthonormal und wegen $Q_i^{-1} = Q_i^T$

$$A = \underbrace{Q_1^T \cdot \dots \cdot Q_{n-1}^T}_{Q} U$$

- damit ist folgender Satz bewiesen

Satz 64. $A \in \mathbb{R}^{n \times n}$, dann existiert Q orthonormal mit $A = QU$, U obere Dreiecksmatrix

Bemerkung 65.

- ◊ aus "historischen" Gründen benutzt man häufiger die Notation $A = QR$
- ◊ Q, R sind nicht eindeutig
- ◊ die oben benutzte Konstruktion nennt sich **Givens-Verfahren**
- ◊ Q ist in der Regel voll besetzt, d.h. Q hat keine Dreiecksgestalt

- praktische Durchföhrung:

- gehe analog zur LU-Zerlegung vor:

$$Ax = b, \quad A = QR \quad \Leftrightarrow \quad Qy = b, \quad Rx = y$$

- letzter Teil funktioniert wie bei LU-Zerlegung, da R obere Dreiecksmatrix ist
- die Handhabung von Q ist komplizierter, da Q keine untere Dreiecksmatrix ist
- betrachte Elimination des Koeffizienten an Position i, j :
 - alle vorhergehenden Koeffizienten sind schon eliminiert, d.h.

$$A \rightarrow \tilde{A} = \begin{pmatrix} \tilde{a}_{11} & \dots & \dots & \dots & \dots & \dots & \dots & \tilde{a}_{1n} \\ 0 & \ddots & & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ & & 0 & \tilde{a}_{jj} & & & & \vdots \\ \vdots & & & \vdots & 0 & \tilde{a}_{j+1j+1} & & \vdots \\ \vdots & & & & \vdots & \vdots & \ddots & \vdots \\ \vdots & & & & 0 & \vdots & & \vdots \\ \vdots & & & & \vdots & \tilde{a}_{ij} & & \vdots \\ \vdots & & & & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \tilde{a}_{nj} & \tilde{a}_{nj+1} & \dots & \tilde{a}_{nn} \end{pmatrix}$$

- benutze

$$Q_{ij} = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c & & s & & \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & -s & & & c \\ & & & & & & & 1 \\ & & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix}$$

mit geeignetem c, s

- $Q_{ij}\tilde{A}$ verändert nur die i -te und j -te Zeile von \tilde{A} :

$$\tilde{a}_{jk} \rightarrow c\tilde{a}_{jk} + s\tilde{a}_{ik}, \quad \tilde{a}_{ik} \rightarrow -s\tilde{a}_{jk} + c\tilde{a}_{ik}$$

- anders als bei LU Zerlegung werden *beide* Zeilen verändert (auch die Diagonalzeile)
- Q_{ij} eliminiert \tilde{a}_{ij} , d.h. $\tilde{a}_{ij} \rightarrow 0$
- Q_{ij} ist durch *eine* reelle Zahl (Drehwinkel) definiert \Rightarrow frei gewordener Platz von \tilde{a}_{ij} reicht, um alle Informationen (Drehwinkel) für Q_{ij} abzuspeichern
- in der Praxis wird nicht der Drehwinkel benutzt, da zur Berechnung trigonometrische Funktionen nötig wären, was sehr aufwändig ist
- speichere z.B. $s = \sin(\alpha)$ und berechne $c = \pm\sqrt{1-s^2}$ \Rightarrow Vorzeichen von c ist nicht klar
- Lösung:

- habe bei Wahl von Q_{ij} noch Vorzeichen frei
- wähle Vorzeichen so, dass $c \geq 0$, d.h. der Drehwinkel liegt in $[-\frac{\pi}{2}, \frac{\pi}{2}]$
- speichere $\rho_{ij} = s$ an Stelle von \tilde{a}_{ij}
- nach Transformation sämtlicher Spalten finden wir im Speicherbereich der Ausgangsmatrix folgende Werte:

$$\begin{array}{ccccccccc} r_{11} & \dots & \dots & & r_{1n} \\ & \ddots & & & \vdots \\ \rho_{21} & & & & \vdots \\ & \ddots & \ddots & & \vdots \\ \vdots & & & & \vdots \\ \rho_{n1} & \dots & \dots & \rho_{nn-1} & r_{nn} \end{array}$$

wobei r_{ij} , $j \geq i$ die Koeffizienten der oberen Dreiecksmatrix R sind und aus den ρ_{ij} die Matrizen Q_{ij} rekonstruiert werden können

- zur Lösung von $Qy = b$ benutzen wir die ρ_{ij} :
- $y = Q^T b = Q_{n-1} \cdot \dots \cdot Q_1 b = Q_{nn-1} \cdot \dots \cdot Q_{n1} \cdot \dots \cdot Q_{21} b$
- betrachte ρ_{ij} in Eliminationsreihenfolge und rekonstruiere Q_{ij} über

$$s = \rho_{ij}, \quad c = \sqrt{1 - s^2}$$

- führe Matrix-Vektor-Produkte aus
- dabei verändert $Q_{ij}z$, $z \in \mathbb{R}^n$ nur z_i und z_j
- Auflösen von $Rx = y$ erfolgt wieder durch Rückwärtseinsetzen

Beispiel 66.

- wir lösen $Ax = b$,

$$A = \begin{pmatrix} 12 & -40 & 46 \\ -9 & 30 & 153 \\ 20 & 100 & 135 \end{pmatrix}, \quad b = \begin{pmatrix} -16 \\ 12 \\ 140 \end{pmatrix}$$

mit Hilfe der QR-Zerlegung nach dem Givens-Verfahrens

- Elimination des Koeffizienten mit Index $i = 2, j = 1$:

$$\blacktriangleright c = \frac{12}{\sqrt{12^2 + (-9)^2}} = \frac{4}{5}, \quad s = \frac{-9}{\sqrt{12^2 + (-9)^2}} = -\frac{3}{5} \quad \text{und damit}$$

$$Q_{21}A = \begin{pmatrix} \frac{4}{5} & -\frac{3}{5} & 0 \\ \frac{3}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 12 & -40 & 46 \\ -9 & 30 & 153 \\ 20 & 100 & 135 \end{pmatrix} = \begin{pmatrix} 15 & -50 & -55 \\ 0 & 0 & 150 \\ 20 & 100 & 135 \end{pmatrix}$$

- speichere $\rho_{21} = s = -\frac{3}{5}$ an Stelle der 0:

$$\begin{array}{r} 15 & -50 & -55 \\ \boxed{-\frac{3}{5}} & 0 & 150 \\ \hline 20 & 100 & 135 \end{array}$$

- Elimination des Koeffizienten mit Index $i = 3, j = 1$:

► $c = \frac{15}{\sqrt{15^2 + 20^2}} = \frac{3}{5}$, $s = \frac{20}{\sqrt{15^2 + 20^2}} = \frac{4}{5}$ und damit

$$Q_{31}Q_{21}A = \begin{pmatrix} \frac{3}{5} & 0 & \frac{4}{5} \\ 0 & 1 & 0 \\ -\frac{4}{5} & 0 & \frac{3}{5} \end{pmatrix} \begin{pmatrix} 15 & -50 & -55 \\ 0 & 0 & 150 \\ 20 & 100 & 135 \end{pmatrix} = \begin{pmatrix} 25 & 50 & 75 \\ 0 & 0 & 150 \\ 0 & 100 & 125 \end{pmatrix}$$

► speichere $\rho_{31} = s = \frac{4}{5}$ an Stelle der 0:

$$\begin{array}{r} 25 & 50 & 75 \\ \hline -\frac{3}{5} & 0 & 150 \\ \hline \frac{4}{5} & 100 & 125 \end{array}$$

► Elimination des Koeffizienten mit Index $i = 3, j = 2$:

► $c = 0, s = 1$ und damit

$$Q_{32}Q_{31}Q_{21}A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 25 & 50 & 75 \\ 0 & 0 & 150 \\ 0 & 100 & 125 \end{pmatrix} = \begin{pmatrix} 25 & 50 & 75 \\ 0 & 100 & 125 \\ 0 & 0 & -150 \end{pmatrix}$$

► speichere $\rho_{31} = s = 1$:

$$\begin{array}{r} 25 & 50 & 75 \\ \hline -\frac{3}{5} & 100 & 125 \\ \hline \frac{4}{5} & 1 & -150 \end{array} \quad (4.11)$$

► damit ist

$$Q = (Q_{32}Q_{31}Q_{21})^T = \frac{1}{25} \begin{pmatrix} 12 & -16 & -15 \\ -9 & 12 & -20 \\ 20 & 15 & 0 \end{pmatrix}, \quad R = Q_{32}Q_{31}Q_{21}A = 25 \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & -6 \end{pmatrix}$$

► $Ax = b$ bestimmen wir über $Qy = b$ (d.h. $y = Q^T b$) und $Rx = y$

► für $y = Q^T b = Q_{32}Q_{31}Q_{21}b$ rekonstruieren wir die Q_{ij} Matrizen aus den in (4.11) abgespeicherten Koeffizienten ρ_{ij} :

► $\rho_{21} = -\frac{3}{5} \Rightarrow s = -\frac{3}{5}, c = \sqrt{1 - s^2} = \frac{4}{5}$ und damit

$$Q_{21}b = \begin{pmatrix} \frac{4}{5} & -\frac{3}{5} & 0 \\ \frac{3}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -16 \\ 12 \\ 140 \end{pmatrix} = \begin{pmatrix} -20 \\ 0 \\ 140 \end{pmatrix}$$

► $\rho_{31} = \frac{4}{5} \Rightarrow s = \frac{4}{5}, c = \sqrt{1 - s^2} = \frac{3}{5}$ und damit

$$Q_{31}Q_{21}b = \begin{pmatrix} \frac{3}{5} & 0 & \frac{4}{5} \\ 0 & 1 & 0 \\ -\frac{4}{5} & 0 & \frac{3}{5} \end{pmatrix} \begin{pmatrix} -20 \\ 0 \\ 140 \end{pmatrix} = \begin{pmatrix} 100 \\ 0 \\ 100 \end{pmatrix}$$

► $\rho_{32} = 1 \quad \Rightarrow \quad s = 1, \quad c = 0$ und damit

$$y = Q_{32}Q_{31}Q_{21}b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 100 \\ 0 \\ 100 \end{pmatrix} = \begin{pmatrix} 100 \\ 100 \\ 0 \end{pmatrix}$$

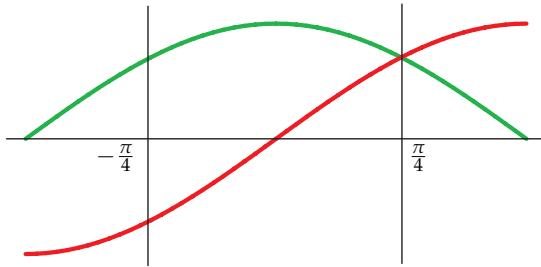
► $Rx = y$ hat dann die Form

$$25 \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & -6 \end{pmatrix} x = 25 \begin{pmatrix} 4 \\ 4 \\ 0 \end{pmatrix},$$

so dass $x = (2, 1, 0)^T$

Bemerkung 67.

- ◊ für $|s| \approx 1$ liefert $c = \sqrt{1-s^2}$ sehr schlechte Werte für c (Auslöschung)
- ◊ deswegen benutzt man folgende stabilisierte Variante, um die Informationen zu den Drehmatrizen zu speichern:
 - ◊ es sei α der Drehwinkel, $c = \cos(\alpha)$ und $s = \sin(\alpha)$
 - ◊ wähle freies Vorzeichen wieder so, dass $c \geq 0$, d.h. $\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}]$



◊ $\alpha \in [-\frac{\pi}{4}, \frac{\pi}{4}]$:

◊ hier gilt

$$|s| \leq \frac{\sqrt{2}}{2} \leq c$$

◊ wegen $|s| \leq \frac{\sqrt{2}}{2} < 0.8$ sind Auslösungen bei $c = \sqrt{1-s^2}$ unkritisch

◊ wir speichern wie vorher

$$\rho = s$$

◊ damit ist

$$|\rho| = |s| \leq \frac{\sqrt{2}}{2} < 1$$

und c, s berechnen sich durch

$$s = \rho, \quad c = \sqrt{1-s^2}$$

◊ $\alpha \in (-\frac{\pi}{2}, -\frac{\pi}{4}) \cup (\frac{\pi}{4}, \frac{\pi}{2})$:

◊ hier ist

$$|s| > \frac{\sqrt{2}}{2} > c > 0$$

- ◊ wegen $0 < c < \frac{\sqrt{2}}{2} < 0.8$ sind Auslöschen bei $|s| = \sqrt{1 - c^2}$ unkritisch
- ◊ damit wäre es günstiger, c statt s abzuspeichern
- ◊ c legt nur den Betrag von s fest, das Vorzeichen muss separat behandelt werden
- ◊ um zusätzlich diesen Fall vom vorherigen unterscheiden zu können benutzen wir schließlich

$$\rho = \frac{\text{sign}(s)}{c}$$

◊ damit ist

$$|\rho| = \frac{1}{c} > \frac{2}{\sqrt{2}} = \sqrt{2} > 1$$

und c, s berechnen sich durch

$$c = \frac{1}{|\rho|}, \quad s = \text{sign}(\rho) \sqrt{1 - c^2}$$

◊ $\alpha \in \{-\frac{\pi}{2}, \frac{\pi}{2}\}$:

◊ hier gilt

$$s = \pm 1, \quad c = 0$$

◊ wir speichern wieder

$$\rho = s$$

◊ damit ist

$$|\rho| = |s| = 1$$

und c, s berechnen sich durch

$$s = \rho, \quad c = \sqrt{1 - s^2} = 0$$

◊ insgesamt erhalten wir

$$\rho = \begin{cases} s & \text{falls } |s| \leq c \\ \frac{\text{sign}(s)}{c} & \text{falls } |s| > c > 0 \\ s & \text{falls } c = 0 \end{cases}$$

◊ aus ρ rekonstruiert man c, s über

$$\begin{array}{lll} s = \rho, & c = \sqrt{1 - s^2} & \text{falls } |\rho| < 1 \\ c = \frac{1}{|\rho|}, & s = \text{sign}(\rho) \sqrt{1 - c^2} & \text{falls } |\rho| > 1 \\ c = 0, & s = \rho & \text{falls } |\rho| = 1 \end{array}$$

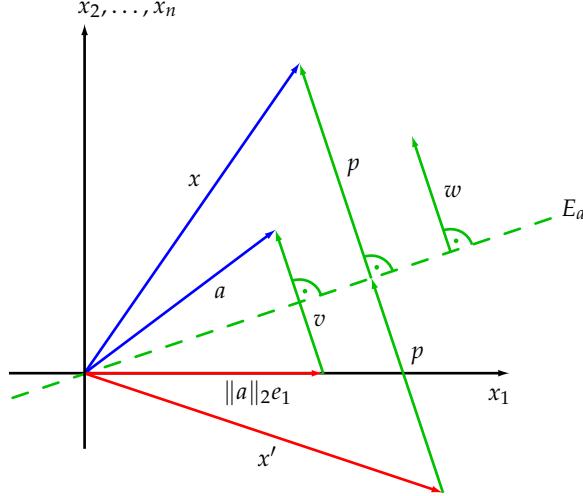
◊ der Aufwand des Givens-Verfahrens ist 4 mal so hoch wie bei der LU-Zerlegung

Zusammenfassung.

- Givens benutzt Drehungen zum Eliminieren
- liefert Zerlegung $A = QR$ mit $\kappa_2(R) \leq \kappa_2(A)$
- Handhabung von Q etwas aufwändiger als bei LU
- Aufwand = $4 \cdot LU$
- Implementierung ähnlich

4.3.3 Verfahren von Householder

- außer Drehungen erhalten auch Spiegelungen $\|\cdot\|_2$
- wir versuchen die erste Spalte $a = (a_{11}, \dots, a_{n1})^T$ von A auf $\pm \|a\|_2 e_1$ zu spiegeln
- wir betrachten zunächst nur $\|a\|_2 e_1$, negatives Vorzeichen geht analog



- a legt Spiegelebene E_a fest
- E_a steht senkrecht auf v , $v = a - \|a\|_2 e_1$
- um ein beliebiges x an E_a zu spiegeln wird
 - der Anteil von x senkrecht E_a bestimmt

$$p = w \langle w, x \rangle, \quad w = \frac{v}{\|v\|_2}$$

- p zweimal von x subtrahiert:

$$x' = x - 2p = x - 2w \langle w, x \rangle = x - 2ww^T x = (I - 2ww^T)x,$$

wobei $ww^T \in \mathbb{R}^{n \times n}$

- insgesamt ist die Spiegelung an E_a also eine lineare Abbildung mit Matrix

$$Q_+ = I - 2ww^T, \quad w = \frac{v}{\|v\|_2}, \quad v = a - \|a\|_2 e_1 \quad (4.12)$$

Satz 68.

- $Q_+ = I - 2ww^T$, $\|w\|_2 = 1$ definiert eine Spiegelung an der Hyperebene $\langle w, x \rangle = 0$
- $Q_+ = Q_+^T = Q_+^{-1}$ und damit orthonormal
- $\det(Q_+) = -1$ d.h. die Orientierung kehrt sich um

- soll a auf $-\|a\|_2 e_1$ gespiegelt werden, ist

$$Q_- = I - 2ww^T, \quad w = \frac{v}{\|v\|_2}, \quad v = a + \|a\|_2 e_1$$

- benutzen wir $Q_1 = Q_+$ bzw. $Q_1 = Q_-$ zu $a = a_1, a_1$ erste Spalte von A , und multiplizieren beide Seiten von $Ax = b$, so erhalten wir

$$Q_1 Ax = Q_1 b, \quad Q_1 A = \begin{pmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{pmatrix}$$

mit $\kappa_2(Q_1) = 1$, da Q_1 orthonormal ist

- Wiederholung desselben Arguments auf der $(n-1) \times (n-1)$ Untermatrix ergibt schließlich

$$Q_{n-1} \dots Q_1 Ax = Q_{n-1} \dots Q_1 b$$

mit

$$Q_{n-1} \dots Q_1 A = R$$

obere Dreiecksmatrix

- somit ist

$$A = QR, \quad Q = (Q_{n-1} \dots Q_1)^{-1} \stackrel{\text{Satz 68}}{=} Q_1 \dots Q_{n-1}$$

- Q ist orthonormal, d.h. auch mit Spiegelungen erhalten wir eine (andere) QR-Zerlegung von A

- dieser Zugang heißt **Householder-Verfahren**

- praktische Durchführung:

- betrachte $A = (a_1, \dots, a_n)$
- bestimme im ersten Schritt

$$Q_1 = I - \frac{2}{\|v\|_2^2} vv^T, \quad v = a_1 \mp \|a_1\|_2 e_1$$

dann gilt $Q_1 a_1 = \pm \|a_1\|_2 e_1$ und damit

$$Q_1 A = (Q_1 a_1, Q_1 a_2, \dots, Q_1 a_n) = (\pm \|a_1\|_2 e_1, Q_1 a_2, \dots, Q_1 a_n)$$

d.h. die erste Spalte ist eliminiert, die Matrix Q_1 muss auf die restlichen Spalten von links her multipliziert werden

- wiederhole analoge Berechnungen für die Schritte $2, \dots, n-1$
- Frage:

- wie bestimmt und speichert man die Q_i ?
- wie berechnet man $Q_i z$, $z \in \mathbb{R}^n$ effizient?
- Berechnen und Speichern der Q_i am Beispiel von Q_1 :
 - berechne $\|a_1\|_2$ und speichere Ergebnis zwischen
 - bestimme

$$v = a_1 \mp \|a_1\|_2 e_1 = \begin{pmatrix} a_{11} \mp \|a_1\|_2 \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}$$

- v unterscheidet sich nur in der ersten Komponente von der ersten Spalte der Ausgangsmatrix \Rightarrow speichere v über erste Spalte
- wähle freies Vorzeichen, so dass Auslöschungen vermieden werden:

$$\begin{aligned} a_{11} < 0 &\Rightarrow "-" \Rightarrow Q_1 a_1 = \|a_1\|_2 e_1 \\ a_{11} \geq 0 &\Rightarrow "+" \Rightarrow Q_1 a_1 = -\|a_1\|_2 e_1 \end{aligned}$$

- für Q_1 brauchen wir auch $\|v\|_2$ (siehe (4.12)):

$$\begin{aligned} \|v\|_2^2 &= (\mp |a_{11}| \mp \|a_1\|_2)^2 + a_{21}^2 + \dots + a_{n1}^2 \\ &= \|a_1\|_2^2 + 2|a_{11}| \|a_1\|_2 + \underbrace{a_{11}^2 + a_{21}^2 + \dots + a_{n1}^2}_{=\|a_1\|_2^2} \\ &= 2(\|a_1\|_2^2 + |a_{11}| \|a_1\|_2) \end{aligned}$$

- die Produkte $Q_1 z$ werden wie folgt effizient berechnet:

- betrachte

$$Q_1 z = (I - \frac{2}{\|v\|_2^2} v v^T) z = z - \frac{2}{\|v\|_2^2} v \underbrace{v^T z}_{<v,z>} = z - \frac{2 <v,z>}{\|v\|_2^2} v$$

- berechne zuerst $<v,z>$ und dann $\alpha = \frac{2 <v,z>}{\|v\|_2^2}$, wobei $\|v\|_2^2$ wie oben beschafft wird
- schließlich ist

$$Q_1 z = z - \alpha v$$

- Speicherproblem:

- v belegt ganze Spalte ab Diagonalelement abwärts
- auf Diagonale sollte aber $\pm \|a_1\|_2$ stehen
- speichere Diagonale in separatem Vektor ab

- insgesamt:

- wiederhole $n - 1$ Spiegelungen
- erhalte $Rx = Q^T b$, $Q^T = Q_{n-1} \cdot \dots \cdot Q_1$
- jedes Q_i ist aus v_i rekonstruierbar
- $Rx = Qb$ ist durch Rückwärtseinsetzen lösbar
- behandle mehrere rechte Seiten wie üblich

- Aufwand: $\sim \frac{4}{3}n^3 \cong 2 \cdot \text{LU} \cong \frac{1}{2} \cdot \text{Givens}$

Beispiel 69.

- wir lösen $Ax = b$,

$$A = \begin{pmatrix} 20 & 18 & 44 \\ 0 & 40 & 45 \\ -15 & 24 & -108 \end{pmatrix}, \quad b = \begin{pmatrix} -4 \\ -45 \\ 78 \end{pmatrix}$$

mit Hilfe der QR-Zerlegung des Householder-Verfahrens

- Elimination der ersten Spalte $a = (20, 0, -15)^T$:

- $\|a\|_2 = \sqrt{20^2 + (-15)^2} = 25$, $a_{11} = 20 > 0$ und damit

$$v = a + 25e_1 = \begin{pmatrix} 20 \\ 0 \\ -15 \end{pmatrix} + \begin{pmatrix} 25 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 45 \\ 0 \\ -15 \end{pmatrix},$$

- mit $\|v\|_2^2 = 45^2 + (-15)^2 = 2250$ folgt

$$\begin{aligned} Q_1 &= I - \frac{2}{\|v\|_2^2} vv^T \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{2}{2250} \begin{pmatrix} 45 \\ 0 \\ -15 \end{pmatrix} \begin{pmatrix} 45 & 0 & -15 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{1125} \begin{pmatrix} 2025 & 0 & -675 \\ 0 & 0 & 0 \\ -675 & 0 & 225 \end{pmatrix} \\ &= \begin{pmatrix} -\frac{4}{5} & 0 & \frac{3}{5} \\ 0 & 1 & 0 \\ \frac{3}{5} & 0 & \frac{4}{5} \end{pmatrix} \end{aligned}$$

und

$$Q_1 A = \begin{pmatrix} -25 & 0 & -100 \\ 0 & 40 & 45 \\ 0 & 30 & -60 \end{pmatrix}$$

- speichere den Vektor v zu Q_1 in der ersten Spalte und das neue Diagonalelement in einem separaten Vektor

$$\begin{array}{c|ccc} 45 & 0 & -100 & -25 \\ 0 & 40 & 45 & * \\ -15 & 30 & -60 & * \end{array}$$

- Elimination der zweiten Spalte $a = (40, 30)^T$:

- $\|a\|_2 = \sqrt{40^2 + 30^2} = 50$, $a_{22} = 40 > 0$ und damit

$$v = a + 50e_1 = \begin{pmatrix} 40 \\ 30 \end{pmatrix} + \begin{pmatrix} 50 \\ 0 \end{pmatrix} = \begin{pmatrix} 90 \\ 30 \end{pmatrix},$$

- mit $\|v\|_2^2 = 90^2 + 30^2 = 9000$ folgt

$$\begin{aligned} \tilde{Q}_2 &= I - \frac{2}{\|v\|_2^2}vv^T \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{2}{9000} \begin{pmatrix} 90 \\ 30 \end{pmatrix} \begin{pmatrix} 90 & 30 \end{pmatrix} \\ &= \begin{pmatrix} -\frac{4}{5} & -\frac{3}{5} \\ -\frac{3}{5} & \frac{4}{5} \end{pmatrix} \end{aligned}$$

- setzt man

$$Q_2 = \begin{pmatrix} 1 & & \\ & \tilde{Q}_2 & \\ & & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{4}{5} & -\frac{3}{5} \\ 0 & -\frac{3}{5} & \frac{4}{5} \end{pmatrix}$$

so folgt

$$R = Q_2 Q_1 A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{4}{5} & -\frac{3}{5} \\ 0 & -\frac{3}{5} & \frac{4}{5} \end{pmatrix} \begin{pmatrix} -25 & 0 & -100 \\ 0 & 40 & 45 \\ 0 & 30 & -60 \end{pmatrix} = \begin{pmatrix} -25 & 0 & -100 \\ 0 & -50 & 0 \\ 0 & 0 & -75 \end{pmatrix}$$

- speichere den Vektor v zu Q_2 (bzw. \tilde{Q}_2) in der zweiten Spalte und das neue Diagonalelement in dem separaten Vektor

$$\begin{array}{c|ccc} 45 & 0 & -100 & -25 \\ 0 & 90 & 0 & -50 \\ -15 & 30 & -75 & * \end{array}$$

- Elimination der dritten Spalte:

- hier ist nichts zu eliminieren, lediglich das Diagonalelement wird in den separaten Vektor umgespeichert:

$$\begin{array}{c|cc|c} 45 & 0 & -100 & -25 \\ 0 & 90 & 0 & -50 \\ -15 & 30 & * & -75 \end{array} \quad (4.13)$$

- damit ist

$$Q = (Q_2 Q_1)^T = \frac{1}{25} \begin{pmatrix} -20 & -9 & 12 \\ 0 & -20 & -15 \\ 15 & -12 & 16 \end{pmatrix}, \quad R = Q_2 Q_1 A = \begin{pmatrix} -25 & 0 & -100 \\ 0 & -50 & 0 \\ 0 & 0 & -75 \end{pmatrix}$$

- $Ax = b$ bestimmen wir über $Qy = b$ (d.h. $y = Q^T b$) und $Rx = y$

- für $y = Q^T b = Q_2 Q_1 b$ rekonstruieren wir die Matrizen Q_i aus den in (4.13) abgespeicherten Spaltenvektoren und erhalten

$$y = Q_2 Q_1 b = \begin{pmatrix} 50 \\ 0 \\ 75 \end{pmatrix}$$

- $Rx = y$ hat dann die Form

$$\begin{pmatrix} -25 & 0 & -100 \\ 0 & -50 & 0 \\ 0 & 0 & -75 \end{pmatrix} x = \begin{pmatrix} 50 \\ 0 \\ 75 \end{pmatrix},$$

so dass $x = (2, 0, -1)^T$

Zusammenfassung.

- Householder benutzt Spiegelungen zum Eliminieren
- liefert Zerlegung $A = QR$ mit $\kappa_2(R) \leq \kappa_2(A)$
- Handhabung von Q aufwändiger als bei LU, brauche zusätzlichen Diagonalenvektor
- Aufwand ist halb so hoch wie bei Givens

Kapitel 5

Iterative Löser für lineare Gleichungssysteme

5.1 Grundlagen

- betrachten $Ax = b$, $A \in \mathbb{R}^{n \times n}$ regulär
- direkte Löser liefern bei exakter Arithmetik nach endlich vielen Schritten die exakte Lösung x
- gibt es auch völlig anders strukturierte Algorithmen dafür?

Beispiel 70.

- wir betrachten das Gleichungssystem

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} x = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \quad x = \begin{pmatrix} u \\ v \end{pmatrix}$$

mit exakter Lösung $x = (2, 1)^T$

- schreiben wir die Gleichungssysteme explizit auf, so erhalten wir

$$\begin{aligned} 2u - v &= 3 \\ -u + 2v &= 0 \end{aligned}$$

bzw.

$$u = \frac{3+v}{2}, \quad v = \frac{u}{2} \tag{5.1}$$

- wir benutzen nun als erste Näherung für die Lösung $x = (u, v)$

$$u_0 = 0, \quad v_0 = 0,$$

setzen diese Werte auf der rechten Seite von (5.1) ein und berechnen damit eine neue Näherung

$$u_1 = \frac{3+v_0}{2} = \frac{3}{2}, \quad v_1 = \frac{u_0}{2} = 0$$

- wiederholen wir diesen Schritt, so erhalten wir weitere Näherungen

$$u_2 = \frac{3+v_1}{2} = \frac{3+0}{2} = \frac{3}{2} \quad v_2 = \frac{u_1}{2} = \frac{3}{4},$$

$$\begin{aligned}
u_3 &= \frac{3+v_2}{2} = \frac{3+\frac{3}{4}}{2} = \frac{15}{8}, & v_3 &= \frac{u_2}{2} = \frac{3}{4}, \\
u_4 &= \frac{3+v_3}{2} = \frac{3+\frac{3}{4}}{2} = \frac{15}{8}, & v_4 &= \frac{u_3}{2} = \frac{15}{16}, \\
&\vdots
\end{aligned}$$

- wir erzeugen also ausgehend von der Anfangsnäherung

$$x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

iterativ eine Folge weiterer Näherungen

$$x_1 = \begin{pmatrix} \frac{3}{2} \\ 0 \end{pmatrix}, \quad x_2 = \begin{pmatrix} \frac{3}{2} \\ \frac{3}{4} \end{pmatrix}, \quad x_3 = \begin{pmatrix} \frac{15}{8} \\ \frac{3}{4} \end{pmatrix}, \quad x_4 = \begin{pmatrix} \frac{15}{8} \\ \frac{15}{16} \end{pmatrix}, \quad \dots$$

- man kann in diesem Fall zeigen, dass die Folge der x_k gegen die exakte Lösung konvergiert, d.h.

$$x_k \xrightarrow{k \rightarrow \infty} \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

- für $k < \infty$ gilt $x_k \neq x$, d.h. erst nach unendlich vielen Schritten erhalten wir die exakte Lösung
- andererseits sind nach wenigen Schritten die Abweichungen zwischen x_k und x schon relativ klein
- wir wollen die Überlegungen aus dem Beispiel verallgemeinern
- dazu betrachten wir ein allgemeines **iteratives Verfahren**:
 - starte mit Anfangsnäherung x_0 von x
 - erzeuge Folge von Näherungen

$$x_0 \rightarrow x_1 \rightarrow \dots \rightarrow x_k \rightarrow \dots$$

so lange bis x_k nahe genug an x liegt

- grundlegende Frage:
 - wie erzeugt man neue x_k
 - wie stellt man fest, ob x_k "nahe" bei x ist (x ist unbekannt), d.h. wann bricht man das Verfahren ab
- zunächst zur Erzeugung der x_k
- benutzt man zur Berechnung von x_{k+1} nur den direkten Vorgänger x_k und keine x_j , $j < k$, so heißt das Verfahren **einstufig**, sonst **mehrstufig**
- wir betrachten *nur* einstufige Verfahren, d.h.

$$x_{k+1} = \Phi_k(x_k),$$

mit **Verfahrensvorschrift** $\Phi_k : \mathbb{R}^n \rightarrow \mathbb{R}^n$

- ist $\Phi_k = \Phi$, d.h. wird in jedem Schritt die selbe Vorschrift benutzt, so heißt das Verfahren **stationär**, sonst **instationär**

Definition 71. Ist $x_{k+1} = \Phi_k(x_k)$ ein iteratives Verfahren zur Lösung von $Ax = b$, dann sind der **Fehler** e_k und das **Residuum** r_k im k -ten Schritt gegeben durch

$$e_k = x - x_k, \quad r_k = b - Ax_k.$$

Lemma 72. $r_k = Ae_k$

Beweis. $Ae_k = A(x - x_k) = Ax - Ax_k = b - Ax_k = r_k$ □

- von einem guten Verfahren Φ_k erwartet man, dass e_k für große k klein wird und zwar unabhängig vom Startwert x_0

Definition 73. Φ_k heißt **konvergent**, falls $\lim_{k \rightarrow \infty} e_k = 0$ für alle $x_0 \in \mathbb{R}^n$

5.2 Lineare stationäre Verfahren

5.2.1 Grundlagen, Konvergenz

- ein stationäres Verfahren hat die Form

$$x_{k+1} = \Phi(x_k) \tag{5.2}$$

- wie sehen die Abbildungen Φ typischerweise aus?
- nach oben gilt

$$e_k = x - x_k = A^{-1}r_k,$$

d.h. aus x_k, r_k kann x durch

$$x = x_k + A^{-1}r_k$$

bestimmt werden

- aber:
 - A^{-1} unbekannt
 - das Berechnen von A^{-1} ist genauso aufwendig wie das Lösen des Ausgangsproblems $Ax = b$
- Idee: ersetze A^{-1} durch M^{-1} mit $M \approx A$ und M^{-1} "einfach" berechenbar
- damit erhalten wir

Definition 74. Die **Residueniteration** mit **Vorkonditionierer** M ist gegeben durch

$$x_{k+1} = x_k + M^{-1}r_k \tag{5.3}$$

Bemerkung 75.

- ◊ M wird **Vorkonditionierer** genannt da

$$x_{k+1} = x_k + M^{-1}Ae_k = x_k + M^{-1}A(x - x_k)$$

- ◊ mit $M^{-1}A \approx I$ folgt $x_{k+1} \approx x_k + x - x_k = x$

- bringen wir (5.3) in die Standardform (5.2) so erhalten wir

$$x_{k+1} = x_k + M^{-1}(b - Ax_k) = \underbrace{(I - M^{-1}A)}_{=:B} x_k + \underbrace{M^{-1}b}_{=:d} = Bx_k + d = \Phi(x_k) \quad (5.4)$$

- damit ist (5.3) ein Spezialfall der folgenden Verfahrensklasse

Definition 76. Ein *einstufiges, stationäres lineares Iterationsverfahren* ist gegeben durch

$$x_{k+1} = \Phi(x_k) = Bx_k + d$$

wobei B die **Iterationsmatrix** und d der **Iterationsvektor** ist.

Bemerkung 77.

- ◊ für die Residueniteration (5.3) gilt nach (5.4)

$$B = I - M^{-1}A, \quad d = M^{-1}b \quad (5.5)$$

bzw.

$$M = A(I - B)^{-1} \quad (5.6)$$

- ◊ wir werden im wesentlichen die Residueniteration (5.3) benutzen und in den nächsten Kapiteln Verfahren zu verschiedenen M vorstellen

- die praktische Durchführung ist einfach:

- berechne pro Schritt:
 - $r_k = b - Ax_k$
 - löse $Mp_k = r_k$ d.h. $p_k = M^{-1}r_k$
 - $x_{k+1} = x_k + p_k$
- Aufwand besteht aus Matrix-Vektor-Produkt mit A und Invertieren von M (wobei M einfach invertierbar sein soll)

Beispiel 78.

- wir betrachten nochmal Beispiel 70
- wenden wir das dort beschriebene Verfahren auf ein System $Ax = b$ an, so erhalten wir als Iterationsvorschrift $x^{(k+1)} = \Phi_k(x^{(k)})$

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

- mit

$$D = \begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{pmatrix}$$

erhalten wir

$$\begin{aligned} x^{(k+1)} &= D^{-1}(b - (A - D)x^{(k)}) \\ &= D^{-1}(b - Ax^{(k)} + Dx^{(k)}) \\ &= x^{(k)} + D^{-1}r^{(k)} \end{aligned}$$

- somit ist dieses Verfahren eine Residueniteration mit $M = D$
- mit (5.5) kann das Verfahren als stationäres lineares Iterationsverfahren geschrieben werden mit

$$x^{(k+1)} = Bx^{(k)} + d, \quad B = I - D^{-1}A, \quad d = D^{-1}b$$

- dieses Verfahren heißt **Jacobi-Verfahren** (siehe Abschnitt 5.2.2)

- wann konvergieren stationäre lineare Iterationsverfahren $x_{k+1} = \Phi(x_k) = Bx_k + d$?
- um dies zu untersuchen, benötigen wir das folgende Resultat

Lemma 79. Alle durch (5.3) definierten Verfahren Φ besitzen die **Fixpunkteigenschaft**

$$x = \Phi(x) \iff Ax = b \tag{5.7}$$

Beweis.

$$\begin{aligned} x = \Phi(x) &\iff x = (I - M^{-1}A)x + M^{-1}b \\ &\iff x = x - M^{-1}Ax + M^{-1}b \\ &\iff M^{-1}Ax = M^{-1}b \\ &\iff Ax = b \end{aligned}$$

□

- nun betrachten wir die Fehlerfortpflanzung

$$\begin{aligned}
e_{k+1} &= x - x_{k+1} \stackrel{(5.7)}{=} \Phi(x) - \Phi(x_k) \\
&= Bx + d - (Bx_k + d) \\
&= B(x - x_k) \\
&= Be_k
\end{aligned}$$

und damit

$$e_k = B^k e_0, \quad e_0 = x - x_0$$

- für Konvergenz muss $e_k \xrightarrow{k \rightarrow \infty} 0 \quad \forall x_0 \in \mathbb{R}^n$ gelten, d.h

$$B^k e_0 \xrightarrow{k \rightarrow \infty} 0 \quad \forall e_0 \in \mathbb{R}^n$$

- für welche B ist das erfüllt?

Bemerkung 80. d beeinflusst die Konvergenz nicht, es stellt nur die Fixpunkteigenschaft (5.7) sicher

Lemma 81. Sei $\|\cdot\|$ eine beliebige Vektornorm auf \mathbb{R}^n und $\|\cdot\|_M$ eine beliebige Matrixnorm auf $\mathbb{R}^{n \times n}$. Für

$$x_{k+1} = \Phi(x_k) = Bx_k + d$$

gilt dann

$$\lim_{k \rightarrow \infty} \|e_k\| = 0 \quad \forall e_0 \in \mathbb{R}^n \quad \Leftrightarrow \quad \lim_{k \rightarrow \infty} \|B^k\|_M = 0$$

Beweis.

" \Leftarrow "

- nach Voraussetzung gilt $\lim_{k \rightarrow \infty} \|B^k\|_M = 0$

- nach oben ist

$$\|e_k\| = \|B^k e_0\| \leq \|B^k\| \|e_0\|$$

wobei $\|B^k\|$ die von der Vektornorm induzierte Matrixnorm ist

- alle Matrixnormen sind äquivalent, d.h. $\exists \beta > 0$ mit $\|B^k\| \leq \beta \|B^k\|_M$

- damit gilt

$$\|e_k\| \leq \beta \|B^k\|_M \|e_0\|$$

so dass

$$\lim_{k \rightarrow \infty} \|e_k\| \leq \beta \|e_0\| \lim_{k \rightarrow \infty} \|B^k\|_M = 0$$

" \Rightarrow "

- nach Voraussetzung gelte $\lim_{k \rightarrow \infty} \|e_k\| = 0 \forall e_0 \in \mathbb{R}^n$
- Annahme: die Behauptung ist falsch, d.h. $\lim_{k \rightarrow \infty} \|B^k\|_M$ existiert nicht oder ist $\neq 0$
- dann existiert eine Teilfolge k_j mit $\|B^{k_j}\|_M \geq c' > 0$
- wegen der Normäquivalenz existiert ein $\alpha > 0$, so dass für die induzierte Matrixnorm gilt

$$\|B^{k_j}\| \geq \alpha \|B^{k_j}\|_M \geq \alpha c' =: c$$

- da $\|\cdot\|$ induziert ist, gibt es Vektoren $x_j \in \mathbb{R}^n$ mit $\|x_j\| = 1$ und $\|B^{k_j} x_j\| = \|B^{k_j}\| \geq c$
- die Folge $(x_j)_{j \in \mathbb{N}}$ ist beschränkt und besitzt deshalb eine konvergente Teilfolge $(x_{j_i})_{i \in \mathbb{N}}$ mit

$$\lim_{i \rightarrow \infty} x_{j_i} = \tilde{x}, \quad \tilde{x} \in \mathbb{R}^n,$$

$$\text{d.h. } \lim_{i \rightarrow \infty} \|x_{j_i} - \tilde{x}\| = 0$$

- setze $e_0 = \tilde{x}$ und betrachte $e_{k_{j_i}} = B^{k_{j_i}} \tilde{x}$:

$$\|e_{k_{j_i}}\| = \|B^{k_{j_i}} \tilde{x}\| = \|B^{k_{j_i}} x_{j_i} - B^{k_{j_i}}(\tilde{x} - x_{j_i})\| \geq \underbrace{\|B^{k_{j_i}} x_{j_i}\|}_{\geq c} - \underbrace{\|B^{k_{j_i}}\|}_{\geq c} \underbrace{\|x_{j_i} - \tilde{x}\|}_{< \varepsilon} \geq c(1 - \varepsilon)$$

- für i groß genug ist also

$$\|e_{k_{j_i}}\| \geq c(1 - \varepsilon)$$

- damit kann das Verfahren für $e_0 = x - x_0 = \tilde{x}$, d.h. $x_0 = x - \tilde{x}$ nicht konvergieren
- nach Voraussetzung war aber $\lim_{k \rightarrow \infty} \|e_k\| = 0 \forall x_0 \in \mathbb{R}^n$, was zu einem Widerspruch führt

□

Bemerkung 82. Die Konvergenz des Verfahrens ist also *unabhängig* von der benutzten *Matrix- und Vektornorm*

- das folgende Kriterium gibt nun genauere Resultate
- dazu benötigen wir nochmal den Spektralradius

Lemma 83. Für den **Spektralradius** $\rho(A) = \max\{|\lambda|, \lambda \text{ Eigenwert von } A\}$ gilt:

- (1) $\rho(A) \leq \|A\|$ für alle induzierten Matrixnormen
- (2) $\forall \varepsilon > 0$ existiert eine induzierte Matrixnorm mit $\rho(A) \leq \|A\| \leq \rho(A) + \varepsilon$
- (3) für jede Matrixnorm gilt

$$\rho(A) = \lim_{k \rightarrow \infty} \|A^k\|^{\frac{1}{k}}$$

Satz 84. Sei A regulär, $x_{k+1} = \Phi(x_k) = Bx_k + d$. Φ konvergiert genau dann wenn $\rho(B) < 1$

Beweis.
 $"\Leftarrow"$

- für eine beliebige Matrixnorm $\|\cdot\|$ gilt nach Lemma 83, (3)

$$\lim_{k \rightarrow \infty} \|B^k\| = \lim_{k \rightarrow \infty} \rho(B)^k \stackrel{\rho(B) < 1}{=} 0$$

- mit Lemma 81 erhalten wir daraus

$$\lim_{k \rightarrow \infty} \|e_k\| = 0 \quad \forall x_0 \in \mathbb{R}^n$$

in jeder Vektornorm $\|\cdot\|$

$"\Rightarrow"$

- nach Voraussetzung soll das Verfahren konvergieren
- nehmen wir an, dass $\rho(B) \geq 1$, dann existiert $\lim_{k \rightarrow \infty} \|B^k\| = \lim_{k \rightarrow \infty} \rho(B)^k$ nicht oder ist ≥ 1
- nach Lemma 81 kann das Verfahren damit nicht konvergieren, d.h. die Annahme $\rho(B) \geq 1$ ist falsch

□

Beispiel 85.

- wir wenden das Jacobi-Verfahren aus Beispiel 78 auf das Gleichungssystem

$$\begin{pmatrix} 4 & 8 \\ -1 & 4 \end{pmatrix} x = b$$

an

- nach oben ist $B = I - D^{-1}A$, also

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 4 & 8 \\ -1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ -\frac{1}{4} & 1 \end{pmatrix} = \begin{pmatrix} 0 & -2 \\ \frac{1}{4} & 0 \end{pmatrix}$$

- die Eigenwerte von B sind komplex

$$\lambda_{1,2} = \pm i \sqrt{\frac{1}{2}}$$

und als Spektralradius erhalten wir

$$\rho(B) = \max(|\lambda_1|, |\lambda_2|) = \sqrt{\frac{1}{2}}$$

- wegen $\rho(B) < 1$ ist das Verfahren für dieses Gleichungssystem konvergent

- wir wissen jetzt genau, wann Φ konvergiert
- wie schnell konvergiert Φ , fällt der Fehler monoton?
- wir müssen $\|e_k\|$ betrachten, wobei die Ergebnisse von der benutzten Norm abhängen
- nach oben muss $\rho(B) < 1$ gelten
- nach Lemma 83 existiert $\forall \varepsilon > 0$ eine induzierte Matrixnorm $\|\cdot\|$ mit

$$\rho(B) \leq \|B\| \leq \rho(B) + \varepsilon$$

- damit gibt es auch eine induzierte Matrixnorm $\|\cdot\|$ mit $\|B\| < 1$ und

$$\|e_{k+1}\| = \|Be_k\| \leq \|B\| \|e_k\| < \|e_k\|,$$

d.h. der Fehler fällt in dieser Norm monoton

- Problem: $\|\cdot\|$ ist in der Regel unbekannt, man benutzt meistens $\|\cdot\|_2$ oder $\|\cdot\|_\infty$
- oft ist $\|B\| \geq 1$ obwohl $\rho(B) < 1$, das Verfahren konvergiert dann zwar auch in dieser Norm, aber nicht notwendig monoton

Beispiel 86.

- wir wenden das Jacobi-Verfahren aus Beispiel 78 auf das System

$$\begin{pmatrix} 4 & 8 \\ -1 & 4 \end{pmatrix} x = \begin{pmatrix} 12 \\ 3 \end{pmatrix}$$

mit exakter Lösung $x = (1, 1)^T$ an

- nach Beispiel 85 ist

$$B = \begin{pmatrix} 0 & -2 \\ \frac{1}{4} & 0 \end{pmatrix}, \quad \rho(B) = \sqrt{\frac{1}{2}} < 1,$$

d.h. das Verfahren konvergiert für jeden beliebigen Startwert $x_0 \in \mathbb{R}^2$ gegen die exakte Lösung

- andererseits ist

$$\|B\|_\infty = 2$$

- für $x_0 = (0, 0)^T$ erhalten wir

k	0	1	2	3	4	5	6	7	8
x_k	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3 \\ \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{2} \\ \frac{3}{2} \end{pmatrix}$	$\begin{pmatrix} 0 \\ \frac{9}{8} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} \\ \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{2} \\ \frac{15}{16} \end{pmatrix}$	$\begin{pmatrix} \frac{9}{8} \\ \frac{9}{8} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} \\ \frac{33}{32} \end{pmatrix}$	$\begin{pmatrix} \frac{15}{16} \\ \frac{15}{16} \end{pmatrix}$
$\ e_k\ _\infty$	1	2	$\frac{1}{2}$	1	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{16}$

- der Fehler e_k geht in der Norm $\|\cdot\|_\infty$ nicht monoton gegen 0

- Konvergenzgeschwindigkeit:

- wir erhalten zunächst die einfache Abschätzung

$$\|e_k\| = \|B^k e_0\| \leq \|B^k\| \|e_0\| \leq \|B\|^k \|e_0\|$$

- nach oben gilt

$$\lim_{k \rightarrow \infty} \|B^k\| = \lim_{k \rightarrow \infty} \rho(B)^k,$$

d.h. für große k ist $\rho(B)$ maßgeblich für Fehlergröße

- für genügend große k liefert Φ' mit $\rho(B') < \rho(B)$ kleinere Fehler als Φ
- für kleine k muss das nicht gelten, da $\rho(B') < \rho(B)$ sein kann und trotzdem $\|B'\| \geq \|B\|$

Beispiel 87.

- wir betrachten nochmal das System

$$\begin{pmatrix} 4 & 8 \\ -1 & 4 \end{pmatrix} x = \begin{pmatrix} 12 \\ 3 \end{pmatrix}$$

aus Beispiel 86 mit exakter Lösung $x = (1, 1)^T$

- wir vergleichen die Ergebnisse des Jacobi-Verfahrens

$$x_{k+1} = x_k + M^{-1}r_k, \quad M = D = \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}$$

aus Beispiel 86 mit denen einer zweiten Residueniteration

$$x'_{k+1} = x'_k + M'^{-1}r'_k, \quad M' = \begin{pmatrix} 4 & 0 \\ -1 & 4 \end{pmatrix}$$

- für das erste Verfahren gilt

$$B = \begin{pmatrix} 0 & 2 \\ \frac{1}{4} & 0 \end{pmatrix}, \quad \rho(B) = \sqrt{\frac{1}{2}}, \quad \|B\|_\infty = 2$$

- für das zweite Verfahren erhalten wir mit $B' = I - M'^{-1}A$

$$B' = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 4 & 0 \\ -1 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 4 & 8 \\ -1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{4} & 0 \\ \frac{1}{16} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 4 & 8 \\ -1 & 4 \end{pmatrix} = \begin{pmatrix} 0 & -2 \\ 0 & -\frac{1}{2} \end{pmatrix}$$

- da die Eigenwerte von B' $\lambda_1 = -\frac{1}{2}$ bzw. $\lambda_2 = 0$ sind, folgt für das zweite Verfahren

$$B' = \begin{pmatrix} 0 & -2 \\ 0 & -\frac{1}{2} \end{pmatrix}, \quad \rho(B') = \frac{1}{2}, \quad \|B'\|_\infty = 2$$

- beide Verfahren sind also konvergent und es gilt

$$\rho(B') < \rho(B), \quad \|B'\|_\infty = \|B\|_\infty$$

- für die Iterierten zum Startwert $x_0 = (0, 0)^T$ erhalten wir

k	0	1	2	3	4	5	6	7	8
x_k	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3 \\ \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{2} \\ \frac{3}{2} \end{pmatrix}$	$\begin{pmatrix} 0 \\ \frac{9}{8} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} \\ \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{2} \\ \frac{15}{16} \end{pmatrix}$	$\begin{pmatrix} \frac{9}{8} \\ \frac{9}{8} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} \\ \frac{33}{32} \end{pmatrix}$	$\begin{pmatrix} \frac{15}{16} \\ \frac{15}{16} \end{pmatrix}$
$\ e_k\ _\infty$	1	2	$\frac{1}{2}$	1	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{16}$
x'_k	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 3 \\ \frac{3}{2} \end{pmatrix}$	$\begin{pmatrix} 0 \\ \frac{3}{4} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{2} \\ \frac{9}{8} \end{pmatrix}$	$\begin{pmatrix} \frac{3}{4} \\ \frac{15}{16} \end{pmatrix}$	$\begin{pmatrix} \frac{9}{8} \\ \frac{33}{32} \end{pmatrix}$	$\begin{pmatrix} \frac{15}{16} \\ \frac{63}{64} \end{pmatrix}$	$\begin{pmatrix} \frac{33}{32} \\ \frac{129}{128} \end{pmatrix}$	$\begin{pmatrix} \frac{63}{64} \\ \frac{255}{256} \end{pmatrix}$
$\ e'_k\ _\infty$	1	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$

- beide Verfahren konvergieren nicht monoton in der Norm $\|\cdot\|_\infty$
- $\|e'_1\|_\infty \geq \|e_1\|_\infty$ und $\|e'_2\|_\infty > \|e_2\|_\infty$
- für größere k geht $\|e'_k\|_\infty$ deutlich schneller gegen 0 als $\|e_k\|_\infty$
- schränkt man die Klasse der betrachteten Matrizen A, M ein, so kann man für die Residueniteration (5.3) explizit eine Norm angeben, in der sie *monoton* konvergiert

Lemma 88. Sei $A \in \mathbb{R}^{n \times n}$ spd, dann ist durch

$$\langle x, y \rangle_A = x^T A y, \quad \|x\|_A = \sqrt{\langle x, x \rangle_A}$$

ein Skalarprodukt und eine Norm auf \mathbb{R}^n definiert

Satz 89. A, M spd. Ist $2M - A$ spd, so konvergiert

$$\Phi(x) = Bx + d, \quad B = I - M^{-1}A, \quad d = M^{-1}b$$

und $\|B\|_A = \|B\|_M = \rho(B) < 1$

Satz 90. A spd, Φ wie im letzten Satz. Ist $M + M^T - A$ spd, dann ist M regulär und Φ konvergiert mit $\rho(B) \leq \|B\|_A < 1$

- wann bricht man die Iteration ab?
- beobachte $\|e_k\| = \|x - x_k\| \Rightarrow$ geht nicht, da x unbekannt
- teste $\|r_k\| = \|Ae_k\|$:
 - meistens wird r_k sowieso berechnet (Residueniteration: $x_{k+1} = x_k + M^{-1}r_k$)
 - ist $\|A\|$ sehr klein, so kann wegen $\|r_k\| \leq \|A\| \|e_k\|$ das Residuum r_k klein sein, obwohl $\|e_k\|$ groß ist
 - Skalierungen des Systems $Ax = b$ beeinflussen die Residuen:

- betrachte

$$\begin{aligned} Ax = b, \quad M &\Rightarrow x_{k+1} = x_k + M^{-1}Ae_k \\ \gamma Ax = \gamma b, \quad \gamma M &\Rightarrow x'_{k+1} = x'_k + \frac{1}{\gamma}M^{-1}\gamma Ae'_k = x'_k + M^{-1}Ae'_k \end{aligned}$$

- mit $x_0 = x'_0$ ist auch $e_0 = e'_0$ und somit $x_k = x'_k, e_k = e'_k \forall k$
- andererseits ist

$$r_k = b - Ax_k, \quad r'_k = \gamma(b - Ax_k) = \gamma r_k,$$

d.h. das Residuum skaliert mit γ

- deswegen prüft man nicht $\|r_k\|$ sondern $\frac{\|r_k\|}{\|r_0\|}$

Zusammenfassung.

- ein stationäres, einstufiges Verfahren $\Phi(x) = Bx + d$ konvergiert genau dann, wenn $\rho(B) < 1$, die Konvergenz ist unabhängig von den benutzten Normen
- $\rho(B)$ bestimmt $\|e_k\|$ für große k , $\|B\|$ für kleine k
- teste $\frac{\|r_k\|}{\|r_0\|}$ als Abbruchkriterium
- einfaches Verfahren: $x_{k+1} = x_k + M^{-1}r_k$
- ist A spd und M geeignet, so erhalten wir für das einfache Verfahren monotone Konvergenz in der $\|\cdot\|_A$ Norm

5.2.2 Jacobi-Verfahren (Gesamtschritt-Verfahren)

- zerlege A in $A = D - E - F$,

$$D = \begin{pmatrix} a_{11} & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{nn} \end{pmatrix}, \quad E = -\begin{pmatrix} 0 & \dots & \dots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn-1} & 0 \end{pmatrix}, \quad F = -\begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1n} \\ & & & 0 \end{pmatrix}$$

- gilt $a_{ii} \neq 0$ so ist D invertierbar
- wir benutzen $M = D$ in unserem allgemeinen Verfahren

$$x^{(k+1)} = x^{(k)} + M^{-1}r^{(k)}, \quad r^{(k)} = b - Ax^{(k)}$$

und erhalten

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + D^{-1}(b - (D - E - F)x^{(k)}) \\ &= x^{(k)} + D^{-1}(b + (E + F)x^{(k)}) - x^{(k)} \\ &= D^{-1}(b + (E + F)x^{(k)}) \end{aligned}$$

- wegen

$$D^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \frac{1}{a_{nn}} \end{pmatrix}$$

erhalten wir komponentenweise

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, \dots, n \tag{5.8}$$

- dieses Verfahren heißt **Jacobi-Verfahren** oder auch **Gesamtschritt-Verfahren**
- es ist ein lineares, stationäres Iterationsverfahren
- wegen

$$x^{(k+1)} = D^{-1}(b + (E + F)x^{(k)}) = D^{-1}(E + F)x^{(k)} + D^{-1}b$$

erhalten wir als Iterationsmatrix bzw. -vektor

$$B = I - M^{-1}A = I - D^{-1}(D - E - F) = D^{-1}(E + F), \quad d = M^{-1}b = D^{-1}b$$

Beispiel 91.

- wir wenden das Jacobi-Verfahren auf das System $Ax = b$,

$$A = \begin{pmatrix} 2 & 2 \\ 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 3 \end{pmatrix},$$

d.h. $x = (1, 1)^T$ ist die exakte Lösung

- zur Berechnung der Iterationsvektoren $x^{(k)}$ benutzt man die komponentenweise Darstellung (5.8):

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)}) = \frac{1}{2}(4 - 2x_2^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)}) = \frac{1}{2}(3 - x_1^{(k)}) \end{aligned}$$

- für den Startvektor $x^{(0)} = (0, 0)^T$ erhalten wir

$$\begin{aligned} x_1^{(1)} &= \frac{1}{2}(4 - 2 \cdot 0) = 2 \\ x_2^{(1)} &= \frac{1}{2}(3 - 0) = \frac{3}{2} \end{aligned} \Rightarrow x^{(1)} = \begin{pmatrix} 2 \\ \frac{3}{2} \end{pmatrix}, \quad \frac{\|r^{(1)}\|_2}{\|r^{(0)}\|_2} \approx 0.722$$

$$\begin{aligned} x_1^{(2)} &= \frac{1}{2}(4 - 2 \cdot \frac{3}{2}) = \frac{1}{2} \\ x_2^{(2)} &= \frac{1}{2}(3 - 2) = \frac{1}{2} \end{aligned} \Rightarrow x^{(2)} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \quad \frac{\|r^{(2)}\|_2}{\|r^{(0)}\|_2} \approx 0.5$$

$$\begin{aligned} x_1^{(3)} &= \frac{1}{2}(4 - 2 \cdot \frac{1}{2}) = \frac{3}{2} \\ x_2^{(3)} &= \frac{1}{2}(3 - \frac{1}{2}) = \frac{5}{4} \end{aligned} \Rightarrow x^{(3)} = \begin{pmatrix} \frac{3}{2} \\ \frac{5}{4} \end{pmatrix}, \quad \frac{\|r^{(3)}\|_2}{\|r^{(0)}\|_2} \approx 0.361$$

$$\begin{aligned} x_1^{(4)} &= \frac{1}{2}(4 - 2 \cdot \frac{5}{4}) = \frac{3}{4} \\ x_2^{(4)} &= \frac{1}{2}(3 - \frac{3}{4}) = \frac{3}{4} \end{aligned} \Rightarrow x^{(4)} = \begin{pmatrix} \frac{3}{4} \\ \frac{3}{4} \end{pmatrix}, \quad \frac{\|r^{(4)}\|_2}{\|r^{(0)}\|_2} \approx 0.25$$

- für $k = 10$ erhalten wir

$$x^{(10)} = \begin{pmatrix} \frac{31}{32} \\ \frac{31}{32} \end{pmatrix} = 0.96875 \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \frac{\|r^{(10)}\|_2}{\|r^{(0)}\|_2} \approx 0.031$$

- die zugehörige Iterationsmatrix

$$B = I - M^{-1}A = I - D^{-1}A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 & 2 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -\frac{1}{2} & 0 \end{pmatrix}$$

hat die Eigenwerte $\lambda_{1,2} = \pm \frac{1}{\sqrt{2}}$

- damit ist $\rho(B) = \frac{1}{\sqrt{2}} < 1$ und das Verfahren konvergiert $\forall x^{(0)} \in \mathbb{R}^n$

- für welche Matrix-Klassen konvergiert das Verfahren, d.h. wann ist $\rho(B) < 1$

Satz 92. Ist A **streng diagonaldominant**, d.h.

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \forall i = 1, \dots, n$$

dann konvergiert das Jacobi-Verfahren (und A ist regulär)

Beweis.

- zu zeigen ist $\rho(B) < 1$ für

$$B = D^{-1}(E + F) = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \ddots & \vdots \\ \vdots & & \ddots & -\frac{a_{n-1n}}{a_{n-1n-1}} \\ -\frac{a_{n1}}{a_{nn}} & \dots & -\frac{a_{nn-1}}{a_{nn}} & 0 \end{pmatrix}$$

- wir wissen, dass $\rho(B) \leq \|B\|$ für jede Norm, also auch $\rho(B) \leq \|B\|_\infty$
- da A streng diagonaldominant ist, gilt aber

$$\|B\|_\infty = \max_i \sum_{j=1}^n |b_{ij}| = \max_i \frac{\sum_{j \neq i} |a_{ij}|}{|a_{ii}|} < 1$$

□

Bemerkung 93. In Satz 92 geben wir ein Kriterium für A an um $\rho(B)$ abzuschätzen

- das Kriterium ist einfach anzuwenden
- es ist nur hinreichend (die Matrix A aus Beispiel 91 erfüllt es nicht, das Jacobi-Verfahren konvergiert aber trotzdem)
- eine wichtige Klasse von Matrizen A , die z.B. in der Numerik partieller Differentialgleichungen auftritt, wird nicht abgedeckt, wie z.B.

$$A = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

die nur schwach diagonaldominant, aber positiv definit ist

- analoge Systeme treten in der Praxis häufig auf \Rightarrow der folgende Satz liefert eine wichtige Aussage

Satz 94. Ist A spd und $2D - A$ ebenfalls spd, so konvergiert das Jacobi-Verfahren und

$$\|B\|_A = \|B\|_D = \rho(B) < 1$$

d.h. es konvergiert monoton in $\|\cdot\|_A$ bzw. $\|\cdot\|_D$

Beweis. Folgt aus Satz 89 mit $M = D$ □

Beispiel 95.

- die Matrix

$$A = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

hat die Eigenwerte

$$\lambda_k = 2 \left(1 - \cos \left(\frac{k\pi}{n+1} \right) \right) \quad k = 1, \dots, n$$

d.h. $\lambda_k > 0$ und A spd

- für $2D - A$ erhalten wir

$$2D - A = \begin{pmatrix} 2 & 1 & & \\ 1 & 2 & 1 & \\ & & \ddots & \end{pmatrix}$$

mit Eigenwerten

$$\mu_k = 4 - \lambda_k = 2 \left(1 + \cos\left(\frac{k\pi}{n+1}\right) \right) \quad k = 1, \dots, n$$

d.h. $\mu_k > 0$, $2D - A$ spd, weshalb nach dem letzten Satz das Jacobi-Verfahren konvergiert

- es gibt weitere Verallgemeinerungen (A schwach diagonaldominant und $A^{-1} \geq 0$ komponentenweise) die allerdings komplizierter zu beweisen sind

5.2.3 Gauß-Seidel-Verfahren (Einzelschritt-Verfahren)

- betrachten das Jacobi-Verfahren komponentenweise (siehe (5.8))

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

d.h. für die neue Komponente $x_i^{(k+1)}$ benutzt man auf der rechten Seite die alten Komponenten $x_j^{(k)}$

- wenn das Verfahren konvergiert, sollte $x^{(k+1)}$ "genauer" als $x^{(k)}$ sein
- wenn $x_i^{(k+1)}$ berechnet wird, kennt man schon $x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}$
- benutze auf der rechten Seite von $x_i^{(k+1)}$ diese neuen Komponenten
- damit erhalten wir das **Gauß-Seidel-Verfahren**, das auch **Einzelschritt-Verfahren** genannt wird

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij} x_j^{(k+1)} - \sum_{j > i} a_{ij} x_j^{(k)} \right), \quad i = 1, \dots, n \quad (5.9)$$

- spalten wir A wieder in $A = D - E - F$ auf, so ist

$$\begin{aligned} x^{(k+1)} &= D^{-1}(b + Ex^{(k+1)} + Fx^{(k)}) \\ \Leftrightarrow \quad Dx^{(k+1)} &= b + Ex^{(k+1)} + Fx^{(k)} \\ \Leftrightarrow \quad (D - E)x^{(k+1)} &= Fx^{(k)} + b \\ &= (A + F)x^{(k)} + b - Ax^{(k)} \\ &= (D - E)x^{(k)} + r^{(k)} \\ \Leftrightarrow \quad x^{(k+1)} &= x^{(k)} + (D - E)^{-1}r^{(k)} \end{aligned}$$

- das Gauß-Seidel-Verfahren ist also ein Verfahren mit Vorkonditionierer

$$M_{GS} = D - E$$

bzw.

$$\begin{aligned} B_{GS} &= I - M_{GS}^{-1}A \\ &= I - (D - E)^{-1}(D - E - F) \\ &= (D - E)^{-1}F \\ d_{GS} &= M_{GS}^{-1}b \\ &= (D - E)^{-1}b \end{aligned}$$

Beispiel 96.

- wir wenden das Gauß-Seidel-Verfahren auf das System aus Beispiel 91 an
- zur Berechnung der Iterationsvektoren $x^{(k)}$ benutzt man die komponentenweise Darstellung (5.9):

$$\begin{aligned}x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)}) = \frac{1}{2}(4 - 2x_2^{(k)}) \\x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k+1)}) = \frac{1}{2}(3 - x_1^{(k+1)})\end{aligned}$$

- für den Startvektor $x^{(0)} = (0, 0)^T$ erhalten wir

$$\begin{aligned}x_1^{(1)} &= \frac{1}{2}(4 - 2 \cdot 0) = 2 \\x_2^{(1)} &= \frac{1}{2}(3 - 2) = \frac{1}{2}\end{aligned} \Rightarrow x^{(1)} = \begin{pmatrix} 2 \\ \frac{1}{2} \end{pmatrix}, \quad \frac{\|r^{(1)}\|_2}{\|r^{(0)}\|_2} \approx 0.2$$

$$\begin{aligned}x_1^{(2)} &= \frac{1}{2}(4 - 2 \cdot \frac{1}{2}) = \frac{3}{2} \\x_2^{(2)} &= \frac{1}{2}(3 - \frac{3}{2}) = \frac{3}{4}\end{aligned} \Rightarrow x^{(2)} = \begin{pmatrix} \frac{3}{2} \\ \frac{3}{4} \end{pmatrix}, \quad \frac{\|r^{(2)}\|_2}{\|r^{(0)}\|_2} \approx 0.1$$

$$\begin{aligned}x_1^{(3)} &= \frac{1}{2}(4 - 2 \cdot \frac{3}{4}) = \frac{5}{4} \\x_2^{(3)} &= \frac{1}{2}(3 - \frac{5}{4}) = \frac{7}{8}\end{aligned} \Rightarrow x^{(3)} = \begin{pmatrix} \frac{5}{4} \\ \frac{7}{8} \end{pmatrix}, \quad \frac{\|r^{(3)}\|_2}{\|r^{(0)}\|_2} \approx 0.05$$

$$\begin{aligned}x_1^{(4)} &= \frac{1}{2}(4 - 2 \cdot \frac{7}{8}) = \frac{9}{8} \\x_2^{(4)} &= \frac{1}{2}(3 - \frac{9}{8}) = \frac{15}{16}\end{aligned} \Rightarrow x^{(4)} = \begin{pmatrix} \frac{9}{8} \\ \frac{15}{16} \end{pmatrix}, \quad \frac{\|r^{(4)}\|_2}{\|r^{(0)}\|_2} \approx 0.025$$

- für $k = 10$ erhalten wir

$$x^{(10)} = \begin{pmatrix} \frac{513}{512} \\ \frac{1023}{1024} \end{pmatrix} = \begin{pmatrix} 1.001953125 \\ 0.9990234375 \end{pmatrix}, \quad \frac{\|r^{(10)}\|_2}{\|r^{(0)}\|_2} \approx 0.000391$$

- die zugehörige Iterationsmatrix

$$B_{GS} = I - M_{GS}^{-1}A = I - (D - E)^{-1}A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{2} & 0 \\ -\frac{1}{4} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 & 2 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 0 & \frac{1}{2} \end{pmatrix}$$

hat die Eigenwerte $\lambda_1 = 0, \lambda_2 = \frac{1}{2}$

- damit ist $\rho(B_{GS}) = \frac{1}{2} < 1$ und das Verfahren konvergiert $\forall x^{(0)} \in \mathbb{R}^2$

- beim Jacobi-Verfahren ist $M_J = D$, beim Gauß-Seidel-Verfahren gilt aber $M_{GS} = D - E$
- wir invertieren beim Gauß-Seidel-Verfahren also einen "größeren Teil" von A und hoffen damit einen besseren Vorkonditionierer und damit schnellere Konvergenz zu erhalten

Satz 97. A strikt diagonaldominant, dann gilt

$$\|B_{GS}\|_\infty \leq \|B_J\|_\infty < 1$$

d.h. das Gauß-Seidel-Verfahren konvergiert

Beweis. analog zu Jacobi, etwas aufwändigere Abschätzungen □

Bemerkung 98. Der letzte Satz garantiert noch nicht, dass das Gauß-Seidel-Verfahren schneller als das Jacobi-Verfahren konvergiert, dazu brauchen wir

$$\rho(B_{GS}) < \rho(B_J)$$

Satz 99 (Stein, Rosenberg). Sei $A = D - E - F$, D invertierbar, $E + F \geq 0$ (komponentenweise) und $\rho(B_J) < 1$. Dann gilt

$$\rho(B_{GS}) < \rho(B_J)$$

Bemerkung 100. Für allgemeines A gibt es Gegenbeispiele mit:

- ◊ Jacobi konvergent, Gauß-Seidel nicht
 - ◊ Gauß-Seidel konvergent, Jacobi nicht
 - ◊ Jacobi schneller als Gauß-Seidel
 - ◊ Gauß-Seidel schneller als Jacobi
- für A spd erhalten wir folgendes Resultat

Satz 101. Ist A spd, so konvergiert das Gauß-Seidel-Verfahren. Die Konvergenz ist monoton in der Norm $\|\cdot\|_A$

Beweis.

- benutzen wir Satz 90, so müssen wir zeigen, dass $M_{GS} + M_{GS}^T - A$ spd ist
- ist A spd dann folgt aus der Symmetrie $A = D - E - F = D - E - E^T$ und mit $M_{GS} = D - E$, $M_{GS}^T = D - E^T$ erhalten wir

$$M_{GS} + M_{GS}^T - A = D - E + (D - E)^T - (D - E - E^T) = D$$

- da A spd ist gilt $a_{ii} > 0 \forall i$ weshalb

$$D = \begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{pmatrix}$$

auch spd ist □

Bemerkung 102. Das Gauß-Seidel-Verfahren konvergiert damit für alle A spd, ohne weitere Voraussetzung

5.2.4 Nachiteration

- je besser M die Matrix A approximiert desto schneller konvergiert die Residueniteration (5.3)
- betrachte direkten Löser wie z.B. LU-Zerlegung
- wegen Rundungsfehler generiert man $\tilde{L}\tilde{U} \approx A$ (P lassen wir der Einfachheit halber mal weg) und \tilde{x} mit $A\tilde{x} \approx b$
- \tilde{L}, \tilde{U} sind Dreiecksmatrizen, also leicht invertierbar
- wir erzeugen eine lineare stationäre Iteration nach (5.3) mit $M = \tilde{L}\tilde{U}$, die sogenannte **Nachiteration**, die die Qualität von \tilde{x} verbessert
- praktische Durchführung:
 - bestimme mit LU-Zerlegung $\tilde{x}, \tilde{L}, \tilde{U}$
 - setze $x_0 = \tilde{x}$ und wiederhole:
 - $r_k = b - Ax_k$
 - $p_k = M^{-1}r_k$ d.h. $\tilde{L}\tilde{U}p_k = r_k$
 - $x_{k+1} = x_k + p_k$
 - bis $\frac{\|p_k\|}{\|x_{k+1}\|} < \varepsilon$
- ohne Rundungsfehler wäre $M = LU = A$ und $M^{-1} = A^{-1}$ d.h. nach einem Schritt hätten wir die exakte Lösung
- mit Rundungsfehlern können wir das nicht erwarten, aber trotzdem ist in der Regel

$$B = I - M^{-1}A = I - (\tilde{L}\tilde{U})^{-1}A$$

nahe an der Nullmatrix und damit $\rho(B) \ll 1$, d.h. der Iterationsprozess sollte sehr schnell konvergieren

- Problem: r_k liegt nahe bei 0, d.h. Fehler durch Auslöschung spielen eventuell eine große Rolle
- deswegen wird r_k oft in höherer Genauigkeit berechnet
- r_k mit hoher Genauigkeit \Rightarrow eine Iteration reicht
- r_k mit gleicher Genauigkeit wie LU-Zerlegung lohnt sich ebenfalls, da Stabilitätseigenschaften verbessert werden
- schlechte Konvergenz ist ein Indikator für schlechte Kondition des Ausgangsproblems $Ax = b$

5.2.5 Relaxationsverfahren

- wir betrachten nochmal das Jacobi-Verfahren

$$x^{(k+1)} = x^{(k)} + D^{-1}r^{(k)}$$

d.h. $M_J = D$, $B_J = I - M_J^{-1}A = D^{-1}(E + F)$

- fügen wir vor dem Korrekturterm einen Parameter $\omega \in \mathbb{R}$ ein und versuchen darüber die Konvergenz günstig zu beeinflussen so erhalten wir das **relaxierte Jacobi-Verfahren** J_ω

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}r^{(k)} \quad (5.10)$$

d.h. $M_{J_\omega} = \frac{1}{\omega}D$ und

$$\begin{aligned} B_{J_\omega} &= I - M_{J_\omega}^{-1}A = I - \omega D^{-1}(D - E - F) = (1 - \omega)I + \omega D^{-1}(E + F) \\ &= (1 - \omega)I + \omega B_J \\ d_{J_\omega} &= M_{J_\omega}^{-1}b = \omega D^{-1}b \\ &= \omega d_J \end{aligned}$$

- komponentenweise gilt dann

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij}x_j^{(k)} \right), \quad i = 1, \dots, n \quad (5.11)$$

- die Konvergenz wird durch $\rho(B_{J_\omega})$ bestimmt, d.h. von den Eigenwerten von B_{J_ω}
- wegen $B_{J_\omega} = (1 - \omega)I + \omega B_J$ gilt

$$\lambda_i(B_{J_\omega}) = 1 - \omega + \omega \lambda_i(B_J) \quad (5.12)$$

Satz 103. Ist $\omega \in (0, 1]$, und konvergiert das Jacobi-Verfahren dann konvergiert auch das relaxierte Jacobi-Verfahren

Beweis. Konvergiert das Jacobi-Verfahren dann ist $|\lambda_i(B_J)| < 1$ und wegen (5.12)

$$|\lambda_i(B_{J_\omega})| \leq (1 - \omega) \cdot 1 + \omega |\lambda_i(B_J)| < 1$$

□

- wähle ω so, dass $\rho(B_{J_\omega})$ möglichst klein ist, d.h. ω_{opt} ist Lösung von

$$\max_{i=1, \dots, n} |1 - \omega + \omega \lambda_i(B_J)| \rightarrow \min$$

- für einfaches B_J kann man ω_{opt} angeben

Satz 104. Hat B_J reelle Eigenwerte $-1 < \lambda_1 \leq \dots \leq \lambda_n < 1$ (damit ist auch $\rho(B_J) < 1$, d.h. das Jacobi-Verfahren konvergiert), so ist $\rho(B_{J_\omega})$ minimal für

$$\omega_{opt} = \frac{2}{2 - \lambda_1 - \lambda_n}$$

- das Problem beim Ergebnis des letzten Satzes ist, dass die Eigenwerte λ_i in der Regel nicht bekannt sind
- sie werden deswegen entweder abgeschätzt oder ω wird empirisch bestimmt

Beispiel 105.

- ▶ wir betrachten nochmals das System aus Beispiel 91
- ▶ das Jacobi-Verfahren konvergiert und B_J hat die Eigenwerte $\lambda_{1,2} = \pm \frac{1}{\sqrt{2}}$, erfüllt also die Voraussetzungen von Satz 104
- ▶ für ω_{opt} erhalten wir

$$\omega_{opt} = \frac{2}{2 - \lambda_1 - \lambda_2} = 1$$

- ▶ das Jacobi-Verfahren ist also bereits optimal

Beispiel 106.

- ▶ das System

$$\begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} x = \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}$$

hat die exakte Lösung $x = (1, 1, 1)^T$

- ▶ für das Jacobi-Verfahren erhalten wir

$$B_J = \frac{1}{3} \begin{pmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}$$

mit Eigenwerten $\lambda_1(B_J) = -\frac{2}{3}$, $\lambda_{2,3}(B_J) = \frac{1}{3}$, also

$$\rho(B_J) = \frac{2}{3}$$

- ▶ für ω_{opt} erhalten wir nach Satz 104

$$\omega_{opt} = \frac{2}{2 - \lambda_1 - \lambda_3} = \frac{2}{2 + \frac{2}{3} - \frac{1}{3}} = \frac{6}{7}$$

- ▶ wegen (5.12) hat B_{J_ω} die Eigenwerte $\lambda_1(B_{J_\omega}) = -\frac{3}{7}$, $\lambda_{2,3}(B_{J_\omega}) = \frac{3}{7}$ und somit

$$\rho(B_{J_\omega}) = \frac{3}{7} < \rho(B_J)$$

- ein Vergleich der beiden Verfahren zum Startwert $x^{(0)} = (0, 0, 0)^T$ liefert

	Jakobi	relaxierter Jakobi
$x^{(0)T}$	(0, 0, 0)	(0, 0, 0)
$x^{(1)T}$	(1.667, 1.667, 1.667)	(1.429, 1.429, 1.429)
$x^{(2)T}$	(0.556, 0.556, 0.556)	(0.817, 0.817, 0.817)
$x^{(3)T}$	(0.803, 0.803, 0.803)	(1.079, 1.079, 1.079)
$x^{(4)T}$	(1.132, 1.132, 1.132)	(0.967, 0.967, 0.967)
$x^{(5)T}$	(0.912, 0.912, 0.912)	(1.015, 1.015, 1.015)
$x^{(6)T}$	(1.059, 1.059, 1.059)	(0.994, 0.994, 0.994)

- nach (5.11) gilt für J_ω

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

- wendet man den Gauß-Seidel-Trick an, so erhält man das **SOR-Verfahren** (Successive-Over-Relaxation)

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij}x_j^{(k+1)} - \sum_{j > i} a_{ij}x_j^{(k)} \right), \quad i = 1, \dots, n \quad (5.13)$$

- für SOR gilt also:

$$M_{SOR} = \frac{1}{\omega} D - E$$

und

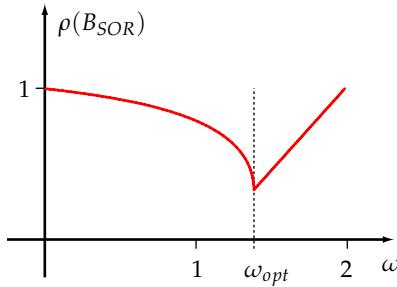
$$\begin{aligned} B_{SOR} &= (I - \omega D^{-1}E)^{-1}((1 - \omega)I + \omega D^{-1}F), \\ d_{SOR} &= M_{SOR}^{-1}b \end{aligned}$$

Satz 107. (Kahan) $\rho(B_{SOR}) \geq |\omega - 1|$, d.h. SOR kann für $\omega \notin (0, 2)$ nicht konvergent sein

Satz 108. (Ostrowski, Reich) Ist A spd dann konvergiert das SOR-Verfahren genau dann wenn $\omega \in (0, 2)$. Die Konvergenz ist monoton in $\|\cdot\|_A$

- für eine spezielle Klasse von Matrizen kann man ω_{opt} angeben (A konsistent geordnet mit A -Property, B_J hat nur reelle Eigenwerte und $\rho(B_J) < 1$):

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(B_J)^2}} > 1, \quad \rho(B_{SOR}) = \omega_{opt} - 1 < \rho(B_J)$$



- deswegen ist es günstiger ω zu groß als zu klein abzuschätzen

5.2.6 Symmetrische Varianten

- für das Gauß-Seidel-Verfahren ist $M_{GS} = D - E$, so dass für A spd M_{GS} in der Regel nicht mehr spd ist
- es gibt Anwendungen (PCG, Multigrid) wo M spd erwünscht ist
- betrachten nochmal die komponentenweise Darstellung des Jacobi- bzw. Gauß-Seidel-Verfahrens
- für das Jacobi-Verfahren hatten wir (vergleiche (5.8))

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

- durchlaufen wir den Index i in umgekehrter Reihenfolge, so ändert sich nichts
- das Gauß-Seidel-Verfahren entsteht durch Benutzung der bereits berechneten Komponenten von $x^{(k+1)}$
- durchlaufen wir i von 1 bis n , so benutzen wir für $x_i^{(k+1)}$ alle $x_j^{(k+1)}$ mit $j < i$ und erhalten wie oben (siehe (5.9))

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

- Vorkonditionierer, Iterationsmatrix und Iterationsvektor sind gegeben durch

$$M_{GS} = D - E, \quad B_{GS} = (D - E)^{-1} F, \quad d_{GS} = M_{GS}^{-1} b$$

- durchlaufen wir i von n bis 1, so benutzen wir für $x_i^{(k+1)}$ alle $x_j^{(k+1)}$ mit $j > i$ und erhalten als neues Verfahren das Rückwärts-Gauß-Seidel-Verfahren

$$x_i^{(k+1)} = \frac{1}{a_{ii}} (b_i - \sum_{j < i} a_{ij} x_j^{(k)} - \sum_{j > i} a_{ij} x_j^{(k+1)}), \quad i = n, \dots, 1 \quad (5.14)$$

- Vorkonditionierer, Iterationsmatrix und Iterationsvektor sind dann

$$M_{RGS} = D - F, \quad B_{RGS} = (D - F)^{-1} E, \quad d_{RGS} = M_{RGS}^{-1} b$$

- für das **symmetrische Gauß-Seidel-Verfahren (SGS)** führt man zunächst einen Gauß-Seidel Schritt und dann einen Rückwärts-Gauß-Seidel Schritt durch

$$x^{(k)} \xrightarrow{GS} B_{GS} x^{(k)} + d_{GS} \xrightarrow{RGS} x^{(k+1)} = B_{RGS} (B_{GS} x^{(k)} + d_{GS}) + d_{RGS}$$

d.h.

$$x^{(k+1)} = \underbrace{B_{RGS} B_{GS}}_{=:B_{SGS}} x^{(k)} + \underbrace{B_{RGS} d_{GS} + d_{RGS}}_{=:d_{SGS}}$$

und somit:

$$\begin{aligned} B_{SGS} &= (D - F)^{-1} E (D - E)^{-1} F, \\ M_{SGS} &= (D - E) D^{-1} (D - F), \\ d_{SGS} &= M_{SGS}^{-1} b \end{aligned}$$

Satz 109. Ist A symmetrisch und $a_{ii} > 0$, $i = 1, \dots, n$, so ist M_{SGS} spd.

Beweis.

- $A = D - E - F$, A symmetrisch, dann ist $F = E^T$ und

$$M_{SGS} = (D - E) D^{-1} (D - F) = (D - E) D^{-1} (D - E^T) = C^T D^{-1} C, \quad C = D - E^T$$

- damit gilt

$$M_{SGS}^T = (C^T D^{-1} C)^T = C^T (D^{-1})^T C = C^T D^{-1} C = M_{SGS},$$

d.h. M_{SGS} ist symmetrisch

- wegen $a_{ii} > 0$ ist D, D^{-1} spd und $C = D - E^T$ regulär

- aus C regulär und D^{-1} spd folgt dann

$$\langle x, M_{SGS} x \rangle = \langle x, C^T D^{-1} C x \rangle = \langle C x, D^{-1} C x \rangle > 0 \quad \forall x \neq 0,$$

weshalb M_{SGS} auch positiv definit ist

□

- analog kann man SOR zu SSOR symmetrisieren

5.3 Nichtstationäre Iterationen

5.3.1 Steepest-Descent-Verfahren (Methode des steilsten Abstiegs)

- starten mit einfacher stationärer Iteration

$$x_{k+1} = x_k + M^{-1} r_k$$

- fügen Parameter α_k ein (vergleiche mit J_ω (5.10))

$$x_{k+1} = x_k + \alpha_k M^{-1} r_k$$

und bestimmen in jedem Schritt α_k so, dass x_{k+1} möglichst "günstige" Eigenschaften hat

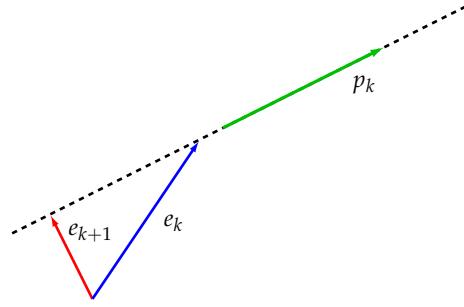
- in der Regel erhalten wir für unterschiedliche k auch unterschiedliche α_k , weshalb die Iteration im Gegensatz zu den oben besprochenen Relaxationsverfahren nicht mehr stationär ist ($x_{k+1} = \Phi_k(x_k)$)
- x_{k+1} "günstig" bedeutet, dass

$$e_{k+1} = x - x_{k+1} = x - x_k - \alpha_k M^{-1} r_k = e_k - \alpha_k p_k, \quad p_k = M^{-1} r_k,$$

in einer geeigneten Norm möglichst klein werden soll

- Geometrie:

- e_{k+1} liegt auf einer Geraden durch e_k in Richtung p_k
- $\|e_{k+1}\|_2$ wird minimal genau dann wenn p_k senkrecht auf e_{k+1}



- $\|e_{k+1}\|_2 \leq \|e_k\|_2$, da schlimmstenfalls e_k senkrecht auf p_k stehen kann und damit $\alpha_k = 0$ ist
- Verallgemeinerung:

- ist $[x, y]$ ein beliebiges Skalarprodukt auf \mathbb{R}^n dann definiert

$$\|x\|_{[\cdot, \cdot]} = \sqrt{[x, x]}$$

eine Norm auf \mathbb{R}^n

- $\|e_{k+1}\|_{[\cdot, \cdot]} = \|e_k - \alpha_k p_k\|_{[\cdot, \cdot]}$ wird minimal, falls p_k senkrecht auf e_{k+1} d.h.

$$0 = [p_k, e_{k+1}] = [p_k, e_k - \alpha_k p_k] = [p_k, e_k] - \alpha_k [p_k, p_k] \Leftrightarrow \alpha_k = \frac{[p_k, e_k]}{[p_k, p_k]}$$

- zusammen haben wir folgendes Verfahren konstruiert:

- starte mit $x_0 \in \mathbb{R}^n$ und wiederhole

$$\begin{aligned} r_k &= b - Ax_k \\ p_k &= M^{-1}r_k \\ \alpha_k &= \frac{[p_k, e_k]}{[p_k, p_k]} \\ x_{k+1} &= x_k + \alpha_k p_k \end{aligned}$$

- x_{k+1} ist mit wenigen Operationen zu berechnen
- $\|e_k\|_{[\cdot, \cdot]}$ ist monoton nicht wachsend
- Problem: bei der Berechnung von α_k muss $[p_k, e_k]$ bestimmt werden mit $e_k = x - x_k$, wobei x unbekannt ist
- dieses Problem kann durch eine spezielle Wahl des Skalarprodukts (und damit der Norm) gelöst werden
- mit A spd ist

$$[x, y] = \langle x, y \rangle_A = \langle x, Ay \rangle = x^T A y$$

ein Skalarprodukt mit zugehöriger Norm

$$\|x\|_{[\cdot, \cdot]} = \|x\|_A = \sqrt{x^T A x}$$

- damit erhalten wir

$$\alpha_k = \frac{\langle p_k, e_k \rangle_A}{\langle p_k, p_k \rangle_A} = \frac{\langle p_k, A e_k \rangle}{\langle p_k, A p_k \rangle} \stackrel{\text{Lemma 72}}{=} \frac{\langle p_k, r_k \rangle}{\langle p_k, A p_k \rangle},$$

d.h. α_k ist über p_k, r_k berechenbar

- benutzt man noch

$$r_{k+1} = b - Ax_{k+1} = b - A(x_k + \alpha_k p_k) = b - Ax_k - \alpha_k Ap_k = r_k - \alpha_k Ap_k$$

so erhalten wir für A spd das folgende Verfahren:

- starte mit $x_0, r_0 = b - Ax_0$ und wiederhole

$$\begin{aligned} p_k &= M^{-1}r_k \\ s_k &= Ap_k \\ \alpha_k &= \frac{\langle p_k, r_k \rangle}{\langle p_k, s_k \rangle} \\ x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k s_k \end{aligned} \tag{5.15}$$

Beispiel 110.

- wir wenden das Verfahren (5.15) mit $M = I$ auf das System $Ax = b$,

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

d.h. $x = (1, 0)^T$ ist die exakte Lösung

- für den Startvektor $x^{(0)} = (0, 0)^T$ erhalten wir

k	p_{k-1}^T	s_{k-1}^T	α_{k-1}	x_k^T	r_k^T	$\frac{\ r^{(k)}\ _2}{\ r^{(0)}\ _2} \approx$
0	—	—	—	$(0, 0)$	$(2, 1)$	1
1	$(2, 1)$	$(5, 4)$	$\frac{5}{14}$	$(\frac{5}{7}, \frac{5}{14})$	$(\frac{3}{14}, -\frac{3}{7})$	0.214
2	$(\frac{3}{14}, -\frac{3}{7})$	$(0, -\frac{9}{14})$	$\frac{5}{6}$	$(\frac{25}{28}, 0)$	$(\frac{3}{14}, \frac{3}{28})$	0.107
3	$(\frac{3}{14}, \frac{3}{28})$	$(\frac{15}{28}, \frac{3}{7})$	$\frac{5}{14}$	$(\frac{95}{98}, \frac{15}{392})$	$(\frac{9}{392}, -\frac{9}{196})$	0.023

- für $k = 10$ erhalten wir

$$x^{(10)} \approx \begin{pmatrix} 0.9999858806 \\ 0 \end{pmatrix}, \quad \frac{\|r^{(10)}\|_2}{\|r^{(0)}\|_2} \approx 0.0000141$$

- Konvergenz:

- wir wissen, dass $\|e_k\|_A$ nicht anwachsen kann
- $\|e_k\|_A$ könnte aber stagnieren

Satz 111. A, M spd, dann konvergiert das obige Verfahren und

$$\|e_{k+1}\|_A \leq \frac{\frac{\lambda_{\max}}{\lambda_{\min}} - 1}{\frac{\lambda_{\max}}{\lambda_{\min}} + 1} \|e_k\|_A$$

wobei $\lambda_{\min}, \lambda_{\max}$ kleinster bzw. größter Eigenwert von $M^{-1}A$ ist

Bemerkung 112.

◊ dass $M^{-1}A$ diagonalisierbar mit reellen, positiven Eigenwerten ist, ist nicht offensichtlich:

◊ da M^{-1}, A spd sind folgt aus der Cholesky-Zerlegung $M^{-1} = KK^T, A = LL^T$ und damit

$$\begin{aligned} M^{-1}Ax &= \lambda x &\Leftrightarrow & KK^T LL^T x = \lambda x \\ &&\Leftrightarrow & \underbrace{L^T K}_{\tilde{L}} \underbrace{K^T L}_{y} \underbrace{L^T x}_{y} = \lambda \underbrace{L^T x}_{y} \\ &&\Leftrightarrow & \underbrace{\tilde{L}\tilde{L}^T}_{\tilde{A}} y = \lambda y \\ &&\Leftrightarrow & \tilde{A}y = \lambda y \end{aligned} \tag{5.16}$$

◊ \tilde{A} ist symmetrisch, denn

$$\tilde{A}^T = (\tilde{L}\tilde{L}^T)^T = \tilde{L}\tilde{L}^T = \tilde{A}$$

◊ weiterhin ist

$$\langle z, \tilde{A}z \rangle = z^T \tilde{L}\tilde{L}^T z = \langle \tilde{L}^T z, \tilde{L}^T z \rangle \geq 0$$

◊ da \tilde{L} regulär ist erhalten wir $\langle z, \tilde{A}z \rangle = 0$ nur für $z = 0$

◊ somit ist \tilde{A} spd, d.h. sämtliche Eigenwerte von \tilde{A} sind reell und positiv und die zugehörigen Eigenvektoren bilden eine Basis des \mathbb{R}^n

◊ nach (5.16) sind die Eigenwerte von \tilde{A} und $M^{-1}A$ identisch

◊ ist y ein Eigenvektor von \tilde{A} zum Eigenwert λ dann erhalten wir mit (5.16) dass

$$x = (L^T)^{-1}y$$

Eigenvektor von A zum selben Eigenwert ist

◊ damit folgt die Behauptung

Bemerkung 113.

◊ $\kappa(M^{-1}A) \approx \frac{\text{max. Verlängerung von } M^{-1}A}{\text{max. Verlängerung von } (M^{-1}A)^{-1}} = \frac{\lambda_{\max}}{\lambda_{\min}} =: \beta$

◊ ist β groß ($M^{-1}A$ schlecht konditioniert) so ist $\frac{\beta-1}{\beta+1} \approx 1$ d.h. Konvergenz des Verfahrens ist extrem langsam

◊ für $M = I$ erhalten wir das klassische **Steepest-Descent-Verfahren**:

◊ betrachte $f : \mathbb{R}^n \rightarrow \mathbb{R}, f(x) = \frac{1}{2}x^T Ax - x^T b$ ("Energie")

◊ ist A spd so hat f ein globales Minimum wenn

$$0 = \nabla f(x) = Ax - b$$

also bei der exakten Lösung von $Ax = b$

◊ wegen $M = I$ ist

$$p_k = r_k = -\nabla f(x_k), \quad x_{k+1} = x_k + \alpha_k r_k,$$

d.h. man geht von x_k in Richtung des steilsten Abstiegs $(-\nabla f(x_k))$, bis $\|e_{k+1}\|_A$ minimal wird

◊ für $M \neq I$ spd erhalten wir das **vorkonditionierte Steepest-Descent-Verfahren**

5.3.2 Konjugierte Gradienten (CG)

- beim Steepest-Descent-Verfahren wird e_{k+1} so bestimmt, dass p_k senkrecht auf e_{k+1} d.h.

$$0 = \langle p_k, e_{k+1} \rangle_A = \langle p_k, r_{k+1} \rangle$$

- für vorherige Suchrichtungen $p_j, j < k$ gilt das nicht, d.h. in der Regel ist

$$0 \neq \langle p_j, e_{k+1} \rangle_A, \quad j < k$$

- Idee: bestimme e_{k+1} so dass

$$\langle p_k, e_{k+1} \rangle_A = \langle p_{k-1}, e_{k+1} \rangle_A = 0$$

- $\|e_{k+1}\|_A$ wird über einen zweidimensionalen Unterraum minimiert, wodurch wir ein schnelleres Abklingen des Fehlers erwarten
- geschicktes Ausrechnen liefert für $M = I$ und A spd das klassische **CG-Verfahren** (Conjugate Gradients):

- x_0 gegeben, $p_0 = r_0 = b - Ax_0$
- wiederhole für $k \geq 0$:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k A p_k & \alpha_k &= \frac{\langle r_k, r_k \rangle}{\langle p_k, A p_k \rangle} \\ p_{k+1} &= r_{k+1} + \beta_k p_k & \beta_k &= \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle} \end{aligned}$$

Beispiel 114.

- wir wenden das CG-Verfahren auf das System $Ax = b$,

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

an, d.h. $x = (1, 0)^T$ ist die exakte Lösung

- für den Startvektor $x^{(0)} = (0, 0)^T$ erhalten wir

k	Ap_{k-1}	α_{k-1}	x_k^T	r_k^T	β_{k-1}	p_k	$\frac{\ r^{(k)}\ _2}{\ r^{(0)}\ _2} \approx$
0	—	—	(0, 0)	(2, 1)	—	(2, 1)	1
1	(5, 4)	$\frac{5}{14}$	$(\frac{5}{7}, \frac{5}{14})$	$(\frac{3}{14}, -\frac{3}{7})$	$\frac{9}{196}$	$(\frac{15}{49}, -\frac{75}{196})$	0.214
2	$(\frac{45}{196}, -\frac{45}{98})$	$\frac{14}{15}$	(1, 0)	(0, 0)	0	(0, 0)	0

- nach $n = 2$ Schritten erreichen wir die exakte Lösung

Satz 115. Ist A spd so gilt für das CG-Verfahren:

- (1) $\langle p_j, e_{k+1} \rangle_A = 0 \quad \forall j = 0, \dots, k$
- (2) nach höchstens n Schritten ist $e_k = 0$
- (3) $\|e_k\|_A \leq 2(\frac{\sqrt{\beta}-1}{\sqrt{\beta}+1})^k \|e_0\|_A, \quad \beta = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$

Bemerkung 116.

- ◊ aus (1) folgt, dass im k -ten Schritt eine k -dimensionale Optimierung erfolgt
- ◊ nach n Schritten würde also über \mathbb{R}^n optimiert \Rightarrow erhalte spätestens dann die exakte Lösung
- ◊ bricht das Verfahren vorher ab ($p_k = 0$ oder $r_k = 0$) so kann man zeigen, dass $x_k = x$ ist
- ◊ da in der Fehlerabschätzung $\sqrt{\beta}$ statt β auftaucht erhalten wir für CG für $\beta > 1$ kleinere Schranken als für Steepest-Descent

- die Rekursionsvorschrift für $M \neq I$ ist etwas aufwändiger und liefert das **vorkonditionierte CG-Verfahren (PCG)**, Preconditioned Conjugate Gradient) für M, A spd:

- x_0 gegeben, $r_0 = b - Ax_0$, $p_0 = q_0 = M^{-1}r_0$
- wiederhole für $k \geq 0$:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k A p_k & \alpha_k &= \frac{\langle q_k, r_k \rangle}{\langle p_k, A p_k \rangle} \\ q_{k+1} &= M^{-1} r_{k+1} \\ p_{k+1} &= q_{k+1} + \beta_k p_k & \beta_k &= \frac{\langle q_{k+1}, r_{k+1} \rangle}{\langle q_k, r_k \rangle} \end{aligned}$$

- als Vorkonditionierer M benutzt man unter anderem:

- $M = \text{diag}(A) = \begin{pmatrix} a_{11} & & & \\ & \ddots & & \\ & & a_{nn} & \end{pmatrix} = M_J$

- M aus einem symmetrischen, stationären Verfahren, wie z.B. **SGS**

$$M = M_{SGS} = (D - E)D^{-1}(D - E)^T$$

- unvollständige Cholesky Zerlegung

Bemerkung 117. PCG ist (neben Multigrid) *das* Verfahren für große, dünn besetzte Matrizen A die spd sind

5.4 Vergleich verschiedener Verfahren

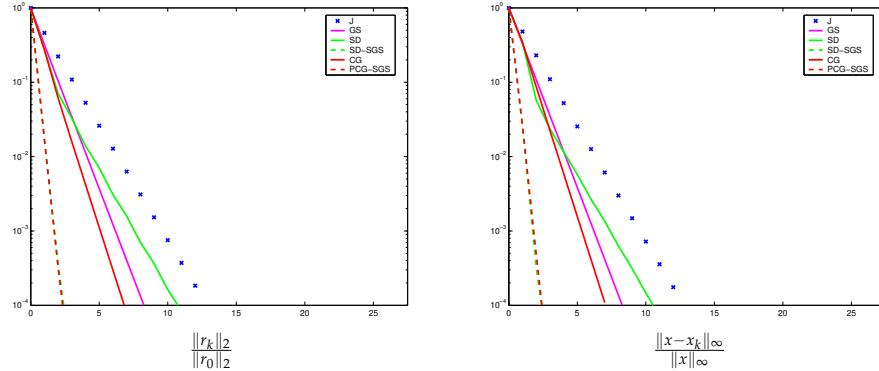
- zum Abschluss vergleichen wir das Verhalten der verschiedenen iterativen Verfahren anhand einiger Beispiele

Beispiel 118.

- wir betrachten $Ax = b$, wobei A die Tridiagonalmatrix

$$\begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{25 \times 25}$$

ist und b so gewählt wird, dass $x = (1, 2, \dots, 25)^T$ die exakte Lösung ist:

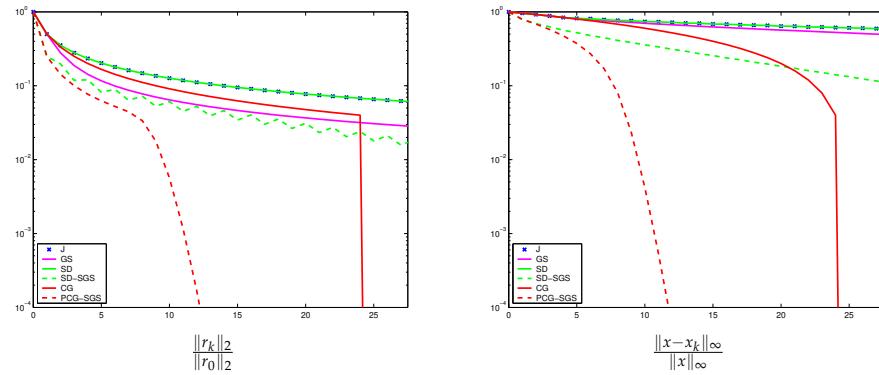


Beispiel 119.

- wir betrachten $Ax = b$, wobei A die Tridiagonalmatrix

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{25 \times 25}$$

ist und b so gewählt wird, dass $x = (1, 2, \dots, 25)^T$ die exakte Lösung ist



5.5 Dünn besetzte Matrizen

- bei (fast) allen Iterationsverfahren besteht der wesentliche Aufwand in einem Matrix-Vektor-Produkt
- der Aufwand dafür ist proportional zu der Anzahl der Matrixelemente
- bei vollbesetzten Matrizen in $\mathbb{R}^{n \times n}$ erhalten wir also $\mathcal{O}(n^2)$
- geht man davon aus, dass man höchstens n Iterationen durchführt, so erhält man wieder, wie bei den direkten Lösern, einen Gesamtaufwand von $\mathcal{O}(n^3)$
- in Anwendungen treten aber oft Gleichungssysteme auf, deren Matrix dünn besetzt ist (sparse), d.h. die Anzahl n_{nz} der Elemente ungleich 0 ist sehr gering (deutlich unter 1%)
- berücksichtigt man bei Matrix-Vektor-Produkten die dünne Besetzungsstruktur der Matrix, so reduziert sich der Aufwand auf $\mathcal{O}(n_{nz})$
- in vielen realen Anwendungen ist $n_{nz} \sim n$, so dass der Aufwand dann nur $\mathcal{O}(n)$ ist
- dünn besetzte Matrizen werden nicht im Standardformat als zweidimensionales Array abgelegt, sondern in einem angepassten Format, dass einerseits das einfache Implementieren von Matrix-Vektor-Produkten zulässt und andererseits möglichst wenig Speicherplatz benötigt
- ein gängiges Datenformat dafür ist Compressed Sparse Row (CSR):
 - bei CSR werden nur die Nichtnullelemente der Matrix abgelegt sowie entsprechende Indexinformationen, in welcher Zeile bzw. Spalte der betreffende Eintrag in der Matrix zu finden ist
 - betrachten wir als Beispiel die Matrix

$$\begin{pmatrix} 1 & 0 & 7 & 0 & 8 \\ 6 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 4 & 5 & 0 & 0 & 0 \end{pmatrix}$$

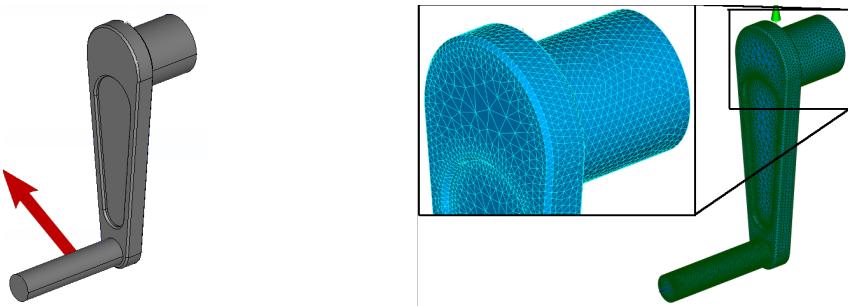
- in CSR werden dafür folgende Daten gespeichert

val	:	1	7	8	6	3	2	4	5
col_ind	:	1	3	5	1	2	4	1	2
row_ind	:	1	4	6	7	9			

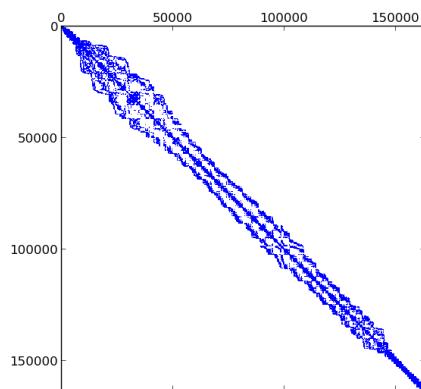
- im Vektor `val` stehen alle Nichtnullelemente der Matrix in zeilenweiser Anordnung (deshalb Compressed Sparse Row)
- der Index-Vektor `col_ind` enthält für jedes Element in `val` den zugehörigen Spaltenindex j
- die Einträge in `col_row` geben an, ab dem wievielten Element in `val` jeweils eine neue Zeile beginnt (i -Index), d.h. in unserem Beispiel gehören die Elemente 1-3 zu Zeile 1, 4-5 zu Zeile 2, 6 zu Zeile 3 und 7-8 zu Zeile 4
- zum Speichern der Matrix in CSR benötigen wir damit 8 Floats und 13 Integer (statt 20 Floats im vollbesetzten Fall)
- bei großen dünn besetzten Matrizen ist das Verhältnis noch wesentlich günstiger

Beispiel 120.

- wir betrachten nochmal das Kurbelproblem aus der Einleitung



- die zugehörige Matrix A für $n = 163\,233$ Unbekannte hat die folgende Besetzungsstruktur



- die Anzahl der Nichtrullelemente beträgt $n_{nz} = 12\,185\,775$, d.h. nur 0.0457% der Einträge sind ungleich 0
- im CSR Format belegt die Matrix ca. 147 MB
- würde man sie als vollbesetzte Matrix (double) abspeichern, so bräuchte man $163\,233^2 \cdot 8$ Byte, also mehr als 213 GB

Bemerkung 121. Weitere gängige Formate sind:

- ◊ Compressed Sparse Column (CSC), analog zu CSR, nur spaltenweise
- ◊ Coordinate List (COO), alle Einträge werden als Liste von Tupeln (i, j, a_{ij}) abgespeichert

Kapitel 6

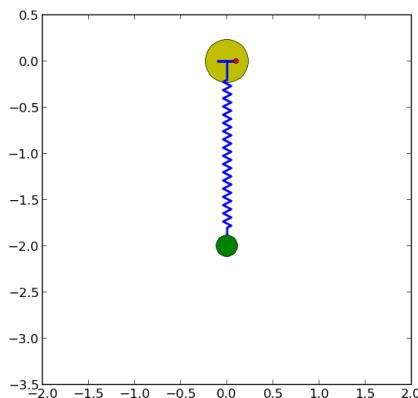
Eigenwerte

6.1 Motivation

- in vielen Anwendungen spielen Resonanzphänomene eine wichtige Rolle

Beispiel 122.

- wir betrachten einen einfachen Federschwinger



- fixiert man den oberen Aufhängepunkt der Feder und lässt die Masse los, so führt das System eine harmonische Schwingung mit einer bestimmten Frequenz und abnehmender Amplitude durch (Dämpfung)
- jetzt regen wir das System zusätzlich an:
 - der obere Aufhängepunkt wird durch eine Exzenter-Scheibe periodisch auf und ab bewegt
 - wir betrachten einen Anfahrprozess, d.h. die Drehzahl des Exzentrers wird langsam von 0 auf eine feste Nenndrehzahl gesteigert
 - ist die Nenndrehzahl erreicht, so wird diese konstant gehalten
- beim Hochfahren der Exzenter-Drehzahl fällt folgendes auf:

- ▶ zu Beginn sind die Schwingungsamplituden sehr klein
- ▶ dreht sich der Exzenter mit einer Frequenz die etwa der Schwingungsfrequenz des nicht angeregten Systems entspricht, so wachsen die Amplituden sehr stark an
- ▶ erhöhen wir die Exzenter-Drehzahl weiter, so gehen die Amplituden wieder zurück
- wir sehen also, dass das System auf eine Anregung mit ganz bestimmten Frequenzen sehr kritisch reagiert
- dies kann im Extremfall sogar zu ernsthaften Schäden führen
- ein bekanntes Beispiel aus der Praxis ist der Einsturz der Tacoma-Narrows Brücke (USA, 1940), bei dem periodisch auftretende Seitenwinde derart extreme Schwingungsamplituden verursachten, dass große Teile der Brücke zerstört wurden
- deshalb ist es wichtig, dass man die Frequenzen, auf die ein System so empfindlich reagiert (**Eigenfrequenzen**), genau kennt
- für den Federschwinger von oben ist (unter der Annahme einer linearen Feder, d.h. Hookesches Gesetz) diese Frequenz gegeben durch

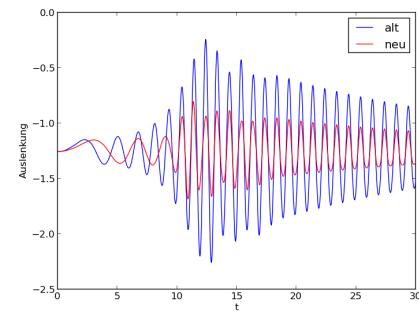
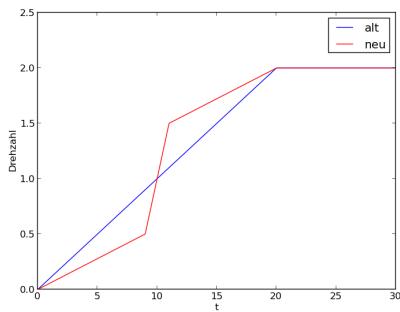
$$f = \frac{1}{2\pi} \sqrt{\frac{k}{m}},$$

wobei k die Federsteifigkeit und m die Masse der an der Feder befestigten Kugel ist

- kennen wir diese Frequenz, so können wir bei unserem Beispiel von oben den Anfahrprozess so abändern, dass die Schwingungsamplituden (und damit auch die Belastungen) während des Hochfahrens reduziert werden

Beispiel 123.

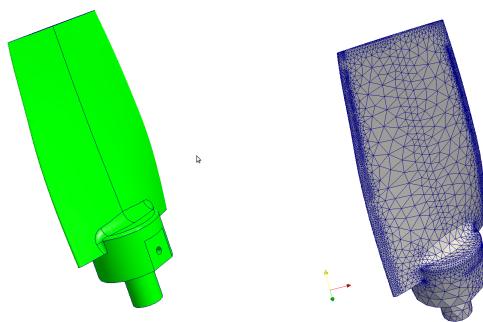
- ▶ wir betrachten den Federschwinger von oben, wobei m, k so gewählt wurden, dass die kritische Frequenz bei 1Hz liegt
- ▶ der Exzenter soll auf einen Nenndrehzahl von 2Hz gebracht werden
- ▶ in der ersten Variante wird die Drehzahl linear von 0Hz auf 2Hz erhöht
- ▶ in der zweiten Variante wird die Drehzahl zunächst langsam linear auf 0.5Hz gesteigert, dann wird relativ schnell auf 1.5Hz erhöht, damit die kritische Frequenz bei 1Hz möglichst schnell "überfahren" wird
- ▶ die dabei auftretenden Schwingungsamplituden sind deutlich kleiner als bei der ersten Variante



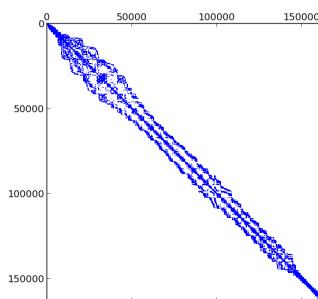
- bei komplexen Bauteile ist das Schwingungsverhalten sehr viel komplizierter
- sie besitzen nicht nur eine sondern sehr viele Eigenfrequenzen
- die zugehörigen Schwingungsformen (**Modes**) können ebenfalls sehr komplex sein (die Schwingungsformen beim Federschwinger war einfach nur auf-und-ab)
- sowohl Eigenfrequenzen als auch Schwingungsformen lassen sich nur für wenige Spezialfälle analytisch berechnen
- analog zu dem Beispiel aus der Strukturmechanik (Kurbel) aus dem ersten Kapitel betrachtet man deswegen wieder ein Ersatzproblem (Finite Elemente)
- das Ersatzmodell beschreibt letztlich das Bauteilverhalten über große, dünn besetzte Matrizen
- die Eigenfrequenzen und Eigenschwingungsformen ergeben sich aus den Eigenwerten und Eigenvektoren dieser Matrizen

Beispiel 124.

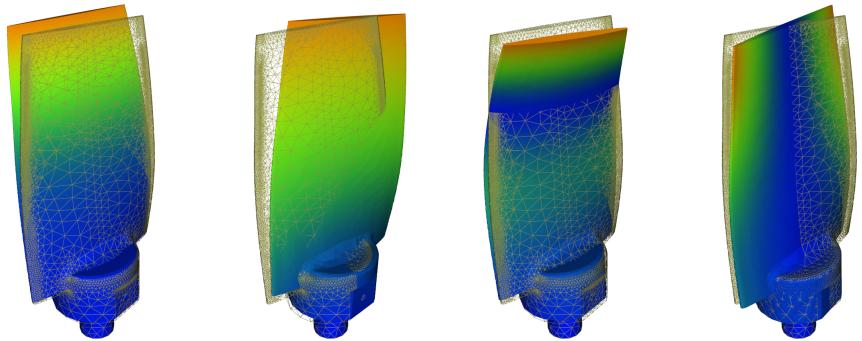
- ▶ wir untersuchen die Eigenschwingungen eines Turbinenblattes
- ▶ das CAD-Modell [6] wurde mit 28 864 Tetraedern vernetzt



- ▶ die damit erzeugten (symmetrischen) Matrizen haben das Format 166512×166512 , die Anzahl der Nichtnullelemente beträgt $n_{nz} = 11\,658\,690$, d.h. nur 0.042% der Einträge sind ungleich 0



- ▶ es wurden die 12 kleinsten Eigenwerte mit den zugehörigen Eigenvektoren berechnet und daraus die entsprechenden Eigenfrequenzen und Modes bestimmt

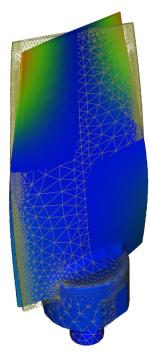


$f \approx 163\text{Hz}$

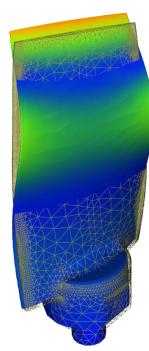
$f \approx 212\text{Hz}$

$f \approx 434\text{Hz}$

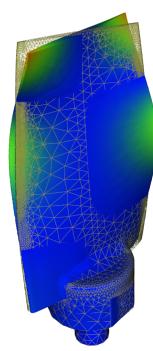
$f \approx 612\text{Hz}$



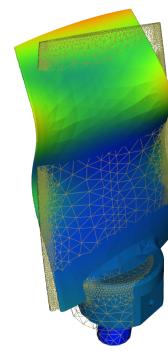
$f \approx 935\text{Hz}$



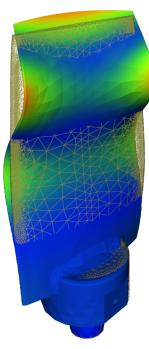
$f \approx 992\text{Hz}$



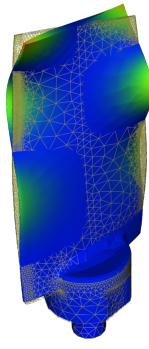
$f \approx 1674\text{Hz}$



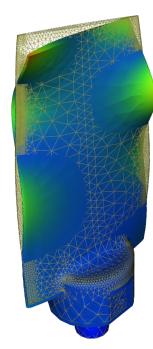
$f \approx 1781\text{Hz}$



$f \approx 2027\text{Hz}$



$f \approx 2636\text{Hz}$



$f \approx 2741\text{Hz}$



$f \approx 2826\text{Hz}$

- analog zum Federschwinger sollte man beim Betrieb der Turbine die den Frequenzen entsprechenden Drehzahlen besonders im Auge behalten

6.2 Theorie

- $A \in \mathbb{R}^{n \times n}$ hat **Eigenwerte** λ falls

$$0 = \det(A - \lambda I) = p_A(\lambda),$$

d.h. falls λ Nullstelle des **charakteristischen Polynoms** p_A ist

- p_A hat Grad n ,

$$p_A(\lambda) = c \cdot (\lambda - \lambda_1) \cdot \dots \cdot (\lambda - \lambda_n)$$

$\lambda_1, \dots, \lambda_n \in \mathbb{C}$ Eigenwerte

- ist $\lambda_i = \lambda_j$, $i \neq j$, so heißt λ_i **mehrfacher Eigenwert**

- $v_i \in \mathbb{R}^n$ ist **Eigenvektor** von A zu λ_i falls $v_i \neq 0$, $Av_i = \lambda_i v_i$

- ist v_i Eigenvektor zu λ_i , dann ist auch $c \cdot v_i$ Eigenvektor, $c \neq 0$

- bei mehrfachen Eigenwerten unterscheidet man die **algebraische Vielfachheit**, d.h. die Vielfachheit der Nullstelle im charakteristischen Polynom p_A , und die **geometrische Vielfachheit**, d.h. die Anzahl der linear unabhängigen Eigenvektoren zu dem mehrfachen Eigenwert

- für beide Werte gilt der Zusammenhang:

$$1 \leq \text{geometrische Vielfachheit} \leq \text{algebraische Vielfachheit}$$

- ist $A \in \mathbb{R}^{n \times n}$ und $T \in \mathbb{R}^{n \times n}$, T regulär, dann haben A und TAT^{-1} die selben Eigenwerte
- besitzt A n linear unabhängige Eigenvektoren v_i mit Eigenwerten λ_i , dann gilt

$$A = T \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} T^{-1}, \quad T = (v_1, \dots, v_n) \quad (6.1)$$

d.h. A ist **diagonalisierbar**

- der folgende Satz liefert eine erste Aussage, wo man die λ_i zu suchen hat

Satz 125 (Gerschgorin). Sei $A \in \mathbb{R}^{n \times n}$ und

$$K = \bigcup_{i=1}^n K_i, \quad K_i = \{z \mid z \in \mathbb{C}, |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}.$$

Dann gilt für die Eigenwerte λ_i von A dass $\lambda_i \in K \forall i$.

Beweis.

- sei λ ein beliebiger Eigenwert von A und $v \neq 0$ der zugehörige Eigenvektor, d.h. $Av - \lambda v = 0$ bzw. komponentenweise

$$(a_{ii} - \lambda)v_i + \sum_{j \neq i} a_{ij}v_j = 0, \quad i = 1, \dots, n \quad (6.2)$$

- betrachte jetzt die betragsgrößte Komponente v_k von v , d.h.

$$|v_k| \geq |v_l| \quad \forall l \neq k$$

- wegen $v \neq 0$ ist $v_k \neq 0$ und aus (6.2) folgt mit $i = k$

$$a_{kk} - \lambda = - \sum_{j \neq k} a_{kj} \frac{v_j}{v_k}$$

bzw.

$$|a_{kk} - \lambda| = \left| \sum_{j \neq k} a_{kj} \frac{v_j}{v_k} \right| \leq \sum_{j \neq k} |a_{kj}| \left| \frac{v_j}{v_k} \right| \leq \sum_{j \neq k} |a_{kj}|$$

□

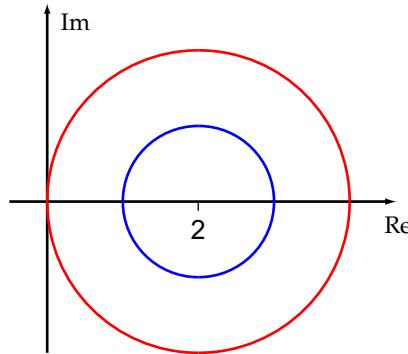
Beispiel 126. Für

$$A = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

erhalten wir

$$K_1 = K_n = \{z \mid |z - 2| \leq 1\}, \quad K_2 = \dots = K_{n-1} = \{z \mid |z - 2| \leq 2\}$$

und damit ist $K = K_2$, d.h. alle Eigenwerte liegen in einer abgeschlossenen Kreisscheibe mit Radius 2 um den Mittelpunkt $(2, 0)$



- ist A symmetrisch, so gelten zusätzlich folgende Aussagen:
 - $\lambda_i \in \mathbb{R} \forall i$
 - es gibt n Eigenvektoren v_i , die linear unabhängig sind, d.h. A ist diagonalisierbar
 - die v_i können so gewählt werden, dass sie eine Orthonormalbasis bilden:

$$\langle v_i, v_j \rangle = \delta_{ij} \quad i, j = 1, \dots, n$$

- $Q = (v_1, \dots, v_n)$ ist damit eine Orthonormalmatrix ($Q^{-1} = Q^T$) und nach (6.1) gilt:

$$A = Q\Lambda Q^{-1} = Q\Lambda Q^T, \quad \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

bzw.

$$\Lambda = Q^T A Q$$

- da die v_i eine Orthonormalbasis bilden, kann jeder Vektor $x \in \mathbb{R}^n$ geschrieben werden als

$$x = \sum_{i=1}^n x_i v_i, \quad x_i = \langle x, v_i \rangle$$

und

$$Ax = \sum_{i=1}^n x_i A v_i = \sum_{i=1}^n \lambda_i x_i v_i$$

- weiterhin gilt für $x = \sum_{i=1}^n x_i v_i$, $y = \sum_{i=1}^n y_i v_i$

$$\langle x, y \rangle = \sum_{i=1}^n x_i y_i, \quad \|x\|_2^2 = \sum_{i=1}^n x_i^2$$

- das Eigenwertproblem ist also ein Nullstellenproblem für das Polynom p_A
- aus der Algebra wissen wir, dass es ab Polynomgrad 5 *keinen* endlichen Algorithmus zur Lösung dieses Problems gibt
- numerische Algorithmen können deswegen nur iterative Verfahren sein

6.3 Vektoriteration

- wir betrachten zunächst ausschließlich symmetrische Matrizen A
- damit ist A diagonalisierbar, d.h. es existiert eine Orthonormalbasis von Eigenvektoren v_1, \dots, v_n
- jedes $x \in \mathbb{R}^n$ kann als $x = \sum_i x_i v_i$, $x_i = \langle x, v_i \rangle$ geschrieben werden und

$$Ax = \sum_i \lambda_i x_i v_i \quad \text{bzw.} \quad A^k x = \sum_i \lambda_i^k x_i v_i$$

- gilt $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$ so dominiert der Anteil $\lambda_1^k x_1 v_1$ (falls $x_1 \neq 0$) in $A^k x$
- wird x oft genug mit A multipliziert, so bleibt (im wesentlichen) ein Vielfaches von v_1 übrig
- ist $|\lambda_1| > 1$ so wird $\|A^k x\|_2$ immer größer
- um die damit verbundenen numerischen Probleme zu vermeiden normalisiert man nach jeder Multiplikation mit A den neu berechneten Vektor und erhält somit die **Vektoriteration**

- wähle $x^{(0)}$ mit $\|x^{(0)}\|_2 = 1$, $x_1^{(0)} = \langle x^{(0)}, v_1 \rangle \neq 0$
- wiederhole

$$\begin{aligned} y^{(k+1)} &= Ax^{(k)} \\ x^{(k+1)} &= \frac{y^{(k+1)}}{\|y^{(k+1)}\|_2} \\ \mu^{(k+1)} &= \langle x^{(k)}, y^{(k+1)} \rangle \end{aligned}$$

- $x^{(k)}$ ist eine Näherung von $c \cdot v_1$, $\mu^{(k)}$ eine Näherung von λ_1
- wir untersuchen zunächst den Fall $\lambda_1 > 0$ genauer
- $\lambda_1 < 0$ geht analog, weshalb nur die Ergebnisse angegeben werden

Satz 127. Ist A symmetrisch mit $\lambda_1 > |\lambda_2| \geq \dots \geq |\lambda_n|$. Dann gilt für die Vektoriteration

$$\mu^{(k)} \xrightarrow{k \rightarrow \infty} \lambda_1, \quad x^{(k)} \xrightarrow{k \rightarrow \infty} \text{sign}(x_1^{(0)})v_1$$

falls $x_1^{(0)} = \langle x^{(0)}, v_1 \rangle \neq 0$ ist.

Beweis.

- per Induktion zeigt man

$$x^{(k)} = \frac{A^k x^{(0)}}{\|A^k x^{(0)}\|_2} = \frac{\sum_{i=1}^n \lambda_i^k x_i^{(0)} v_i}{\sqrt{\sum_{i=1}^n (\lambda_i^k x_i^{(0)})^2}} = \frac{\sum_{i=1}^n (\frac{\lambda_i}{\lambda_1})^k x_i^{(0)} v_i}{\sqrt{\sum_{i=1}^n ((\frac{\lambda_i}{\lambda_1})^k x_i^{(0)})^2}}$$

- nach der Voraussetzung ist

$$\left| \frac{\lambda_i}{\lambda_1} \right| < 1 \quad \text{für } i > 1,$$

so dass $(\frac{\lambda_i}{\lambda_1})^k \xrightarrow{k \rightarrow \infty} 0$ für $i > 1$ und somit

$$x^{(k)} \xrightarrow{k \rightarrow \infty} \frac{x_1^{(0)} v_1}{\sqrt{x_1^{(0)2}}} = \frac{x_1^{(0)}}{|x_1^{(0)}|} v_1 = \text{sign}(x_1^{(0)}) v_1$$

- daraus folgt

$$\begin{aligned} \mu^{(k+1)} &= \langle x^{(k)}, y^{(k+1)} \rangle \\ &= \langle x^{(k)}, Ax^{(k)} \rangle \\ &\xrightarrow{k \rightarrow \infty} \langle \text{sign}(x_1^{(0)}) v_1, A(\text{sign}(x_1^{(0)}) v_1) \rangle \\ &= \text{sign}(x_1^{(0)})^2 \langle v_1, \lambda_1 v_1 \rangle \\ &= \lambda_1 \end{aligned}$$

□

Beispiel 128.

- wir wenden die Vektoriteration auf

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}, \quad x^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

an

- der betragsgrößte Eigenwert ist $\lambda_1 = 2(1 + \cos(\frac{\pi}{5})) \approx 3.6180$ mit zugehörigem Eigenvektor

$$v_1 = \frac{w}{\|w\|_2} \approx \begin{pmatrix} 0.3717 \\ 0.6015 \\ 0.6015 \\ 0.3717 \end{pmatrix}, \quad w = \begin{pmatrix} \sin(\frac{\pi}{5}) \\ \sin(\frac{2\pi}{5}) \\ \sin(\frac{3\pi}{5}) \\ \sin(\frac{4\pi}{5}) \end{pmatrix}$$

► $k = 1$:

$$y^{(1)} = Ax^{(0)} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad x^{(1)} = \frac{y^{(1)}}{\|y^{(1)}\|} = \frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} \approx \begin{pmatrix} 0.89 \\ 0.45 \\ 0 \\ 0 \end{pmatrix}$$

$$\mu^{(1)} = \langle x^{(0)}, y^{(1)} \rangle = \langle \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} \rangle = 2$$

► $k = 2$:

$$y^{(2)} = Ax^{(1)} = \frac{1}{\sqrt{5}} \begin{pmatrix} 5 \\ 4 \\ 1 \\ 0 \end{pmatrix}, \quad x^{(2)} \approx \begin{pmatrix} 0.77 \\ 0.62 \\ 0.15 \\ 0 \end{pmatrix}, \quad \mu^{(2)} \approx 2.8$$

► $k = 15$:

$$x^{(15)} \approx \begin{pmatrix} 0.3793 \\ 0.6061 \\ 0.5968 \\ 0.3641 \end{pmatrix}, \quad \mu^{(15)} \approx 3.6177$$

Beispiel 129.

► wir wenden die Vektoriteration auf die Matrix

$$\begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}$$

an und bestimmen die ersten 5 Iterierten zum Startvektor $x_0 = \frac{1}{\sqrt{3}}(1, 1, 1)^T$:

k	$\mu^{(k)}$	$x^{(k)T}$
0	—	(0.5774, 0.5774, 0.5774)
1	1.2333	(0.8080, 0.4775, 0.3452)
2	1.4068	(0.8254, 0.4614, 0.3252)
3	1.4083	(0.8269, 0.4600, 0.3235)
4	1.4083	(0.8270, 0.4599, 0.3233)
5	1.4083	(0.8270, 0.4599, 0.3233)

Bemerkung 130.

- ◊ $x_1^{(0)} \neq 0$ ist unkritisch, aufgrund von Rundungsfehlern stellt sich nach einigen Iterationen in der Regel ein Anteil in Richtung v_1 ein
- ◊ für $\lambda_1 < 0$, $|\lambda_1| > |\lambda_{p+1}| \geq \dots \geq |\lambda_n|$ zeigt man analog zum Beweis des letzten Satzes, dass für $x_1^{(0)} = \langle x^{(0)}, v_1 \rangle \neq 0$

$$\mu^{(k)} \xrightarrow{k \rightarrow \infty} \lambda_1, \quad (-1)^k x^{(k)} \xrightarrow{k \rightarrow \infty} \text{sign}(x_1^{(0)}) v_1$$

- ◊ ist $\lambda_1 = \dots = \lambda_p$, $|\lambda_1| > |\lambda_{p+1}| \geq \dots \geq |\lambda_n|$, so tritt der betragsgrößte Eigenwert mehrfach auf und die Aussagen von oben gelten analog:

$$\mu^{(k)} \rightarrow \lambda_1, \quad x^{(k)} \rightarrow v \quad \text{für } \lambda_1 > 0,$$

$$\mu^{(k)} \rightarrow \lambda_1, \quad (-1)^k x^{(k)} \rightarrow v \quad \text{für } \lambda_1 < 0$$

mit $Av = \lambda_1 v$, wobei v nicht notwendig der Eigenvektor v_1 zu λ_1 ist

- ◊ für $\lambda_1 = \dots = \lambda_q = -\lambda_{q+1} = \dots = -\lambda_p > |\lambda_{p+1}| \geq \dots \geq |\lambda_n|$ konvergiert die Vektoriteration nicht (man wendet sie dann auf A^2 an und kann λ_1^2 approximieren)
- ◊ unter den Voraussetzungen von Satz 127 kann man zeigen, dass

$$|\lambda_1 - \mu^{(k)}| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$$

Zusammenfassung.

- die Vektoriteration ist sehr einfach zu implementieren (ein Matrix-Vektor-Produkt und eine Norm pro Schritt)
- sie liefert nur Approximationen für den betragsgrößten Eigenwert und den zugehörigen Eigenvektor

6.4 Jacobi-Verfahren

- wir versuchen ein Verfahren zu konstruieren, das Näherungen für alle Eigenwerte und -vektoren liefert
- dazu beschränken wir uns auf symmetrische Matrizen A , d.h. es gibt orthonormale Matrizen Q mit

$$Q^T A Q = \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}, \quad Q = (v_1, \dots, v_n)$$

wobei λ_i, v_i die Eigenwerte und -vektoren von A sind

- wir wissen, dass es kein direktes Verfahren geben kann um die λ_i, v_i zu bestimmen und somit kann auch Q nicht in endlich vielen Schritten erzeugt werden
- deshalb versuchen wir, A iterativ mit einfachen orthonormalen Matrizen Q_k auf näherungsweise Diagonalgestalt zu transformieren

$$A^{(0)} = A, \quad A^{(k)} = Q_k^T A^{(k-1)} Q_k \tag{6.3}$$

- die "einfachsten" Q_k , die wir kennen, sind Givens-Rotationen

$$Q_k = \begin{pmatrix} 1 & & & & & \\ \ddots & \ddots & & & & \\ & 1 & c & & s & \\ & & \ddots & 1 & & \\ & & & 1 & c & \\ & & & & -s & \\ & & & & & 1 \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \quad (6.4)$$

- $A^{(k)}$ soll "näher" an Λ sein als $A^{(k-1)}$, d.h. die Nebendiagonalelemente sollten möglichst klein sein:
 - suche das betragsgrößte Nebendiagonalelement $a_{i_0 j_0}^{(k-1)}$ von $A^{(k-1)}$
 - dabei kann $j_0 > i_0$ angenommen werden, da alle Matrizen $A^{(k-1)}$ symmetrisch sind
 - bestimme Q_k so, dass $a_{i_0 j_0}^{(k)} = 0$
- das kann erreicht werden wenn Q_k wie in (6.4) gewählt wird mit

$$\begin{aligned} c &= \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{\frac{\alpha^2}{1+\alpha^2}}}, & s &= \frac{\text{sign}(\alpha)}{2c\sqrt{1+\alpha^2}}, \\ \alpha &= \frac{a_{j_0 j_0} - a_{i_0 i_0}}{2a_{i_0 j_0}}, & \text{sign}(t) &= \begin{cases} 1 & \text{für } t \geq 0 \\ -1 & t < 0 \end{cases} \end{aligned} \quad (6.5)$$

also

$$Q_k = \begin{pmatrix} 1 & & & & & \\ \ddots & \ddots & & & & \\ & 1 & c & & s & \\ & & \ddots & 1 & & \\ & & & 1 & c & \\ & & & & -s & \\ & & & & & 1 \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \quad \begin{matrix} i_0 & & j_0 \\ & i_0 & \\ & & j_0 \end{matrix}$$

- die Multiplikation $Q_k^T A^{(k-1)} Q_k$ verändert Einträge in den Spalten und Zeilen mit Index i_0 und j_0
- wurde dort im vorherigen Schritt eine 0 erzeugt, so kann diese jetzt wieder verschwinden, d.h. wir eliminieren *nicht* nach und nach die Nebendiagonalelemente
- trotzdem haben wir damit etwas gewonnen

Satz 131. Die Quadratsumme $N(A^{(k)})$ der Nebendiagonalelemente der Matrizen $A^{(k)}$ aus (6.3),(6.5) ist streng monoton fallend mit

$$N(A^{(k)}) = \sum_{i \neq j} (a_{ij}^{(k)})^2 \xrightarrow{k \rightarrow \infty} 0.$$

Satz 132. Sind $\lambda_i, a_{ii}^{(k)}$ aufsteigend sortiert, so gilt

$$|a_{ii}^{(k)} - \lambda_i| \leq \sqrt{N(A^{(k)})},$$

d.h. die sortierten Diagonalelemente von $A^{(k)}$ konvergieren gegen die Eigenwerte von A

- wir berechnen also $A^{(k)} = Q_k^T A^{(k-1)} Q_k$ und benutzen

$$\text{diag}(A^{(k)}) = \begin{pmatrix} a_{11}^{(k)} & & \\ & \ddots & \\ & & a_{nn}^{(k)} \end{pmatrix}$$

als Näherung für Λ , also als Näherung für alle Eigenwerte λ_i

- wie man eine Approximation der Eigenvektoren erhält, kann man sich durch folgende Überlegungen plausibel machen:

- nach (6.3) ist

$$A^{(k)} = Q_k^T \cdot \dots \cdot Q_1^T A \underbrace{Q_1 \cdot \dots \cdot Q_k}_{=: Q^{(k)}}$$

und deshalb

$$A = Q^{(k)} A^{(k)} Q^{(k)T}$$

- andererseits ist $A = Q \Lambda Q^T$, $Q = (v_1, \dots, v_n)$
- ist $A^{(k)}$ eine gute Näherung von Λ (d.h. sind die Nebendiagonalelemente klein), so liefern die Spalten von $Q^{(k)}$ eine Approximation der v_i
- zur Implementierung erweitert man die Iteration um

$$Q^{(0)} = I, \quad Q^{(k)} = Q^{(k-1)} Q_k$$

Beispiel 133.

- wir wenden das Jacobi-Verfahren auf die Matrix

$$\begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}$$

an und iterieren, bis $\sqrt{N(A^{(k)})} \leq 10^{-3}$ ist:

k	Q_k	$A^{(k)}$	$Q^{(k)}$
0	–	$\begin{pmatrix} 1.0000 & 0.5000 & 0.3333 \\ 0.5000 & 0.3333 & 0.2500 \\ 0.3333 & 0.2500 & 0.2000 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
1	$\begin{pmatrix} 0.8817 & -0.4719 & 0 \\ 0.4719 & 0.8817 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1.2676 & 0 & 0.4119 \\ 0 & 0.0657 & 0.0631 \\ 0.4119 & 0.0631 & 0.2000 \end{pmatrix}$	$\begin{pmatrix} 0.8817 & -0.4719 & 0 \\ 0.4719 & 0.8817 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
2	$\begin{pmatrix} 0.9465 & 0 & -0.3227 \\ 0 & 1 & 0 \\ 0.3227 & 0 & 0.9465 \end{pmatrix}$	$\begin{pmatrix} 1.4080 & 0.0204 & 0 \\ 0.0204 & 0.0657 & 0.0598 \\ 0 & 0.0598 & 0.0596 \end{pmatrix}$	$\begin{pmatrix} 0.8345 & -0.4719 & -0.2845 \\ 0.4466 & 0.8817 & -0.1523 \\ 0.3227 & 0 & 0.9465 \end{pmatrix}$
3	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.7251 & -0.6887 \\ 0 & 0.6887 & 0.7251 \end{pmatrix}$	$\begin{pmatrix} 1.4080 & 0.0148 & -0.0140 \\ 0.0148 & 0.1225 & 0 \\ -0.0140 & 0 & 0.0028 \end{pmatrix}$	$\begin{pmatrix} 0.8345 & -0.5381 & 0.1187 \\ 0.4466 & 0.5344 & -0.7176 \\ 0.3227 & 0.6518 & 0.6863 \end{pmatrix}$
4	$\begin{pmatrix} 0.9999 & -0.0115 & 0 \\ 0.0115 & 0.9999 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1.4082 & 0 & -0.0140 \\ 0 & 0.1223 & 0.0002 \\ -0.0140 & 0.0002 & 0.0028 \end{pmatrix}$	$\begin{pmatrix} 0.8283 & -0.5476 & 0.1187 \\ 0.4527 & 0.5293 & -0.7176 \\ 0.3302 & 0.6481 & 0.6863 \end{pmatrix}$
5	$\begin{pmatrix} 1.0000 & 0 & 0.0100 \\ 0 & 1 & 0 \\ -0.0100 & 0 & 1.0000 \end{pmatrix}$	$\begin{pmatrix} 1.4083 & 0.0000 & 0 \\ 0.0000 & 0.1223 & 0.0002 \\ 0 & 0.0002 & 0.0027 \end{pmatrix}$	$\begin{pmatrix} 0.8270 & -0.5476 & 0.1269 \\ 0.4599 & 0.5293 & -0.7130 \\ 0.3233 & 0.6481 & 0.6895 \end{pmatrix}$

► damit erhalten wir die Näherungen

$$\begin{aligned}\lambda_1 &\approx 1.4083, & v_1 &\approx (0.8270, 0.4599, 0.3233)^T, \\ \lambda_2 &\approx 0.1223, & v_2 &\approx (-0.5476, 0.5293, 0.6481)^T, \\ \lambda_3 &\approx 0.0027, & v_3 &\approx (0.1269, -0.7130, 0.6895)^T\end{aligned}$$

► die auf 6 Nachkommastellen gerundeten exakten Werte sind

$$\begin{aligned}\lambda_1 &\approx 1.408319, & v_1 &\approx (0.827045, 0.459864, 0.323298)^T, \\ \lambda_2 &\approx 0.122327, & v_2 &\approx (-0.547448, 0.528290, 0.649007)^T, \\ \lambda_3 &\approx 0.002687, & v_3 &\approx (0.127659, -0.713747, 0.688672)^T\end{aligned}$$

Zusammenfassung.

- das Jacobi-Verfahren liefert Approximationen für alle Eigenwerte und Eigenvektoren, $N(A^{(k)})$ nimmt streng monoton ab
- bei größerem n ist das Suchen von $a_{i_0 j_0}$ aufwändig und die Konvergenzgeschwindigkeit unter Umständen sehr niedrig

6.5 QR-Verfahren, Hessenberg-Transformation, Shift-Technik

- mit Givens- bzw. Householder-Transformationen können wir eine reguläre Matrix A in $A = QR$ zerlegen, wobei Q orthonormal und R eine obere Dreiecksmatrix ist
- eine analoge Zerlegung ist auch für singuläres A möglich:

- falls eine Spalte inklusive des Diagonalelements gleich 0 ist, benutzt man als Dreh- bzw. Spiegelmatrix $Q_k = I$
- auf der Diagonalen der oberen Dreiecksmatrix R trägt man entsprechend eine 0 ein
- wir zerlegen eine beliebige quadratische Matrix A in $A = QR$ und betrachten

$$RQ = Q^T QRQ = Q^T A Q$$

- da $Q^T = Q^{-1}$ ist

$$RQ = Q^{-1} A Q,$$

d.h. RQ und $A = QR$ haben die selben Eigenwerte

- wir bauen daraus wie üblich eine Iteration auf und erhalten das **QR-Verfahren**:

- starte mit $A^{(0)} = A$
- wiederhole:
 - zerlege $A^{(k)} = Q_k R_k$
 - berechne $A^{(k+1)} = R_k Q_k$

Satz 134. Sei $A \in \mathbb{R}^{n \times n}$, $\lambda_i \in \mathbb{R}$ die Eigenwerte von A mit $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$, dann gilt für die mit dem QR-Verfahren erzeugten Matrizen $A^{(k)}$

$$A^{(k)} \xrightarrow{k \rightarrow \infty} \begin{pmatrix} \lambda_1 & * & \dots & * \\ & \ddots & \ddots & \vdots \\ & & \ddots & * \\ & & & \lambda_n \end{pmatrix}$$

Ist A symmetrisch, so folgt

$$A^{(k)} \xrightarrow{k \rightarrow \infty} \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

- bei symmetrischem A konvergieren die $A^{(k)}$ wie beim Jacobi-Verfahren gegen die Diagonalmatrix Λ
- wegen

$$A^{(k+1)} = R_k Q_k = Q_k^T Q_k R_k Q_k = Q_k^T A^{(k)} Q_k$$

ist

$$A^{(k)} = Q_{k-1}^T \cdot \dots \cdot Q_0^T A \underbrace{Q_0 \cdot \dots \cdot Q_{k-1}}_{=: Q^{(k)}},$$

also

$$A = Q^{(k)} A^{(k)} Q^{(k)T},$$

so dass die Spalten von $Q^{(k)}$ wieder als Approximation der Eigenvektoren benutzt werden können

- der Aufwand pro Schritt entspricht im wesentlichen einer Householder-Zerlegung ($\sim \frac{4}{3}n^3$) und ist damit extrem hoch
- ist es möglich, A vorab in eine Form zu bringen, die den Aufwand der QR-Zerlegungen senkt?
- dazu benutzen wir Householder-Matrizen auf verkürzten Spalten von A :

- wir betrachten die erste Spalte s_1 und die um das Diagonalelement verkürzte erste Spalte \tilde{s}_1

$$s_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}, \quad \tilde{s}_1 = \begin{pmatrix} a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}$$

und konstruieren die Householder-Matrix \tilde{Q}_1 die \tilde{s}_1 eliminiert, d.h.

$$\tilde{Q}_1 \tilde{s}_1 = \alpha_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- erweitern wir $\tilde{Q}_1 \in \mathbb{R}^{n-1 \times n-1}$ zu

$$Q_1 := \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \tilde{Q}_1 & \\ 0 & & & \end{pmatrix}$$

dann ist $Q_1 \in R^{n \times n}$ orthonormal mit

$$Q_1 A = \begin{pmatrix} a_{11} & * & \dots & \dots & * \\ \alpha_1 & * & \dots & \dots & * \\ 0 & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & * & \dots & \dots & * \end{pmatrix}, \quad Q_1 A Q_1^T = Q_1 A Q_1 = \begin{pmatrix} a_{11} & * & \dots & \dots & * \\ \alpha_1 & * & \dots & \dots & * \\ 0 & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & * & \dots & \dots & * \end{pmatrix}$$

- analoges Vorgehen für Spalte 2 bis $n - 2$ liefert

$$H = \underbrace{Q_{n-2} \cdots Q_1}_Q A \underbrace{Q_1^T \cdots Q_{n-2}^T}_{Q^T = Q^{-1}} = \begin{pmatrix} * & \dots & & * \\ * & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & * & * \end{pmatrix}$$

- $H = QAQ^{-1}$ hat dieselben Eigenwerte wie A und besitzt **Hessenberg-Form**, d.h. das um die erste Nebendiagonale reduzierte untere Dreieck ist identisch 0

Beispiel 135.

- für die Hilbertmatrix A der Dimension 5 erhalten wir

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix},$$

$$H = \begin{pmatrix} 1 & -0.680890 & 0 & 0 & 0 \\ -0.680890 & 0.743987 & 0.091410 & 0 & 0 \\ 0 & 0.091410 & 0.042413 & 0.002695 & 0 \\ 0 & 0 & 0.002695 & 0.000893 & -0.000041 \\ 0 & 0 & 0 & -0.000041 & 0.000009 \end{pmatrix}$$

- für das QR-Verfahren bringt die Hessenberg-Form folgenden Vorteil:
 - bei der QR-Zerlegung einer Hessenberg-Matrix muss pro Spalte nur das Subdiagonalelement eliminiert werden, was jeweils mit *einer einzigen* Givens-Rotation erledigt werden kann, so dass der Gesamtaufwand nur noch $\sim n^2$ statt $\sim \frac{4}{3}n^3$ ist
 - hat $A^{(k)}$ Hessenberg-Form und wurde mit Givens-Matrizen in $A^{(k)} = Q_k R_k$ zerlegt, so hat sowohl Q_k als auch $A^{(k+1)} = R_k Q_k$ wieder Hessenberg-Form
- startet man das QR-Verfahren also mit $A^{(0)}$ in Hessenberg-Form, so haben alle $A^{(k)}$ Hessenberg-Form, d.h. die QR-Zerlegung kann mit Givens-Rotationen mit $\sim n^2$ Operationen durchgeführt werden
- ist A symmetrisch, so vereinfacht sich das ganze nochmals:
 - die Hessenberg-Transformierte H ist wegen $H = QAQ^{-1} = QAQ^T$ ebenfalls symmetrisch und muss deshalb tridiagonal sein:

$$H = \begin{pmatrix} * & * & 0 & \dots & 0 \\ * & * & * & \ddots & \vdots \\ 0 & * & * & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \dots & 0 & * & * \end{pmatrix}$$

- wird das QR-Verfahren mit tridiagonalem $A^{(0)}$ gestartet, so sind alle $A^{(k)}$ tridiagonal und der Aufwand für die Zerlegung $A^{(k)} = Q_k R_k$ und die Matrixmultiplikation $R_k Q_k$ ist jeweils $\sim n$
- mit der Hessenberg-Transformation haben wir den Aufwand reduziert, aber die Konvergenzgeschwindigkeit kann noch verbessert werden
- dazu benutzen wir den folgenden Zusammenhang:
 - zerlege statt A die Matrix $A - \mu I = QR$
 - dann hat $\tilde{A} = RQ + \mu I$

wegen $Q^T = Q^{-1}$ dieselben Eigenwerte wie A , denn

$$\tilde{A} = RQ + \mu I$$

$$\begin{aligned}
&= Q^T QRQ + \mu I \\
&= Q^T(A - \mu I)Q + \mu I \\
&= Q^T A Q - \mu Q^T Q + \mu I \\
&= Q^T A Q
\end{aligned}$$

- damit bauen wir in das QR-Verfahren einen **Shift** ein, d.h. wir ändern die Iteration in

$$\begin{aligned}
A^{(k)} - \mu^{(k)} I &= Q_k R_k \\
A^{(k+1)} &= R_k Q_k + \mu^{(k)} I
\end{aligned}$$

wobei $\mu^{(k)}$ geeignet zu bestimmen ist

- der zusätzliche Aufwand für den Shift ist gering ($\sim 2n$)
- wie wählt man $\mu^{(k)}$ geschickt?
- wir untersuchen ausschließlich den Fall, dass A nur reelle Eigenwerte besitzt (zur Behandlung komplexer Eigenwerte sind zusätzliche Überlegungen notwendig):
 - betrachte den 2×2 Block

$$B^{(k)} := \begin{pmatrix} a_{n-1 n-1}^{(k)} & a_{n-1 n}^{(k)} \\ a_{n n-1}^{(k)} & a_{n n}^{(k)} \end{pmatrix}$$

und bestimme dessen Eigenwerte γ_1, γ_2

- als $\mu^{(k)}$ benutzen wir den Eigenwert γ_i , für den $|\gamma_i - a_{nn}^{(k)}|$ minimal wird
- mit dieser Wahl konvergiert $a_{nn-1}^{(k)}$ schnell gegen 0 und $a_{nn}^{(k)}$ gegen einen Eigenwert λ_n von A , d.h. ist $l > k$ groß genug, so ist

$$A^{(l)} = \begin{pmatrix} * & \dots & \dots & \dots & * \\ * & \ddots & & & \vdots \\ & \ddots & \ddots & & \vdots \\ & & * & * & * \\ & & & \varepsilon & \tilde{\lambda}_n \end{pmatrix}$$

wobei ε klein und $\tilde{\lambda}_n$ eine gute Näherung für λ_n ist

- damit haben wir einen Eigenwert approximiert und die restlichen Eigenwerte sind ausschließlich durch die ersten $n - 1$ Spalten und Zeilen der Matrix bestimmt
- deshalb streichen wir die letzte Zeile und Spalte und wiederholen das Verfahren auf der verbliebenen Restmatrix
- diese Reduktionstechnik nennt sich **Deflation**
- insgesamt erhalten wir folgendes Verfahren:

- setze $k = 0$
- bestimme aus A die Hessenbergmatrix $A^{(0)}$ und setze $Q^{(0)} = I$
- wiederhole bis $n = 1$:
 - wiederhole bis $|a_{nn-1}^{(k)}| \leq \varepsilon$:

- bestimme $\mu^{(k)}$ aus $B^{(k)}$
- zerlege $A^{(k)} - \mu^{(k)} I$ in $Q_k R_k$ mit Givens-Rotationen
- $A^{(k+1)} = R_k Q_k + \mu^{(k)} I$
- $Q^{(k+1)} = Q^{(k)} Q_k$
- $k := k + 1$
- $n := n - 1$
- die Diagonalelemente von $A^{(k)}$ sind Approximationen der Eigenwerte
- die Spalten von $Q^{(k)}$ sind Approximationen der zugehörigen Eigenvektoren
- die Implementierung in dieser Form zählt zu den Standardverfahren und findet sich so auch in den einschlägigen Bibliotheken (z.B. LAPACK)

Beispiel 136.

- wir bestimmen die Eigenwerte der Hilbertmatrix mit Dimension 5 (siehe Beispiel 135):

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix}$$

mit dem Jacobi-Verfahren, dem QR-Verfahren mit vorheriger Hessenberg-Transformation bzw. dem QR-Verfahren mit vorheriger Hessenberg-Transformation und Shift

- als Abbruchkriterium wird jeweils $N(A) < 10^{-6}$ benutzt

Verfahren	Jacobi	QR ohne Shift	QR mit Shift
Iterationen	25	8	5
λ_1	1.56705069109823	1.56705069109822	1.56705069109823
λ_2	0.20853421861008	0.20853421861101	0.20853421861101
λ_3	0.01140749162378	0.01140749162320	0.01140749162342
λ_4	0.00030589801324	0.00030589804030	0.00030589804015
λ_5	0.00000328795626	0.00000328792885	0.00000328792877

Kapitel 7

Nichtlineare Gleichungen

7.1 Einleitung

- bisher haben wir immer lineare Gleichungssysteme $Ax = b$ gelöst, d.h. Nullstellen

$$f(x) = 0, \quad f : \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad f(x) = Ax - b$$

gesucht

- jetzt betrachten wir $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ nichtlinear
- Problem:
 - Eindeutigkeit ($f(x) = x^2 - 1$)
 - keine direkten Verfahren möglich (Nullstellen von Polynomen mit Grad ≥ 5)
- betrachten deshalb ausschließlich iterative Verfahren
- Konvergenz:
 - f linear: $\forall x_0$ muss $x_k \xrightarrow{k \rightarrow \infty} x$ gelten (global)
 - für f nichtlinear können wir dies nicht mehr erwarten und müssen uns mit Konvergenz für $x_0 \in X \subset \mathbb{R}^n$ begnügen (**lokale Konvergenz**)
- Vorgehensweise in den nächsten Abschnitten
 - betrachten vorwiegend den eindimensionalen Fall $f : \mathbb{R} \rightarrow \mathbb{R}$
 - starten mit intuitiver geometrischer Herleitung einiger Verfahren
 - definieren allgemeinen Iterationsprozess für $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ (Erweiterung des linearen Falls), beweisen ein hinreichendes Konvergenzkriterium inklusive Fehlerabschätzungen und untersuchen die Konvergenzgeschwindigkeit
 - wenden die allgemeinen Aussagen auf das wichtigste Verfahren (Newton-Verfahren) an, um den eindimensionalen Fall detailliert zu untersuchen
 - zum Abschluss gehen wir noch auf eine Verallgemeinerung für $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ein

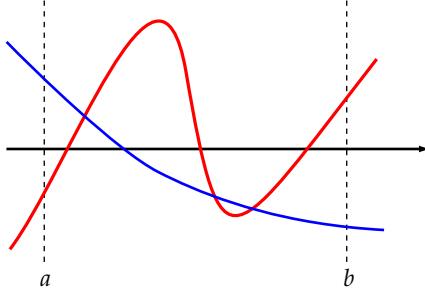
7.2 Einfache Verfahren und ihre geometrische Interpretation

7.2.1 Bisektions-Verfahren

- wir setzen $f : \mathbb{R} \rightarrow \mathbb{R}$ stetig voraus
- ist $a_0, b_0 \in \mathbb{R}$ mit $f(a_0) < 0, f(b_0) > 0$ oder $f(a_0) > 0, f(b_0) < 0$, also

$$f(a_0) \cdot f(b_0) < 0,$$

dann muss in $[a_0, b_0]$ mindestens eine Nullstelle von f liegen



- betrachte $x_0 = \frac{a_0+b_0}{2}$, d.h. die Intervallmitte
- ist $f(x_0) \cdot f(a_0) < 0$ dann liegt sicher eine Nullstelle in $[a_0, x_0]$ sonst in $[x_0, b_0]$
- setze also

$$\begin{aligned} a_1 &= a_0, & b_1 &= x_0 && \text{falls } f(x_0) \cdot f(a_0) < 0, \\ a_1 &= x_0, & b_1 &= b_0 && \text{sonst} \end{aligned}$$

und wiederhole dieselben Überlegungen für $[a_1, b_1]$

- halbiere Intervalle fortlaufend
- erhalten Folge x_k die gegen eine Nullstelle konvergiert

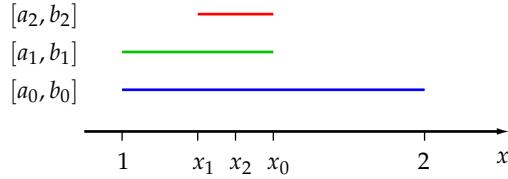
Bemerkung 137. Verfahren, die untere und obere Schranken a_k und b_k für eine gesuchte Nullstelle erzeugen, nennt man **Einschließungsverfahren**

Beispiel 138.

- $f(x) = x^2 - 2$
- $f(1) = -1, f(2) = 2$, weshalb wir $[a_0, b_0] = [1, 2]$ wählen:

$$\begin{aligned} x_0 &= \frac{a_0 + b_0}{2} = \frac{3}{2} & f(x_0) &= \left(\frac{3}{2}\right)^2 - 2 = \frac{1}{4} & \Rightarrow & [a_1, b_1] = \left[1, \frac{3}{2}\right] \\ x_1 &= \frac{a_1 + b_1}{2} = \frac{5}{4} & f(x_1) &= \left(\frac{5}{4}\right)^2 - 2 = -\frac{7}{16} & \Rightarrow & [a_2, b_2] = \left[\frac{5}{4}, \frac{3}{2}\right] \\ x_2 &= \frac{11}{8} & f(x_2) &= \left(\frac{11}{8}\right)^2 - 2 = -\frac{7}{64} & \Rightarrow & [a_3, b_3] = \left[\frac{11}{8}, \frac{3}{2}\right] \end{aligned}$$

- geometrisch sieht das wie folgt aus:



- iteriere so lange, bis die Intervalllänge $|b_k - a_k|$ oder $|f(x_k)|$ hinreichend klein ist
- Konvergenzgeschwindigkeit:

- $x \in [a_0, b_0]$, $x_0 = \frac{a_0 + b_0}{2}$ und somit

$$|e_0| = |x - x_0| \leq \frac{b_0 - a_0}{2}$$

- analog gilt

$$|e_k| = |x - x_k| \leq \frac{b_k - a_k}{2}$$

- wegen der Intervallhalbierung haben wir

$$b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_{k-2} - a_{k-2}}{4} = \dots = \frac{b_0 - a_0}{2^k}$$

und damit

$$|e_k| \leq \frac{b_0 - a_0}{2^{k+1}} \quad (7.1)$$

bzw. für den relativen Fehler

$$|e_{k,r}| \leq \frac{b_0 - a_0}{|x|} \cdot \frac{1}{2^{k+1}}$$

- Vorteil:

- f muss nur stetig sein
- Fehler einfach abschätzbar
- einfach zu implementieren

- Nachteil:

- ein Anfangsintervall $[a_0, b_0]$ mit $f(a_0) \cdot f(b_0) < 0$ muss bekannt sein
- die Konvergenz ist relativ langsam, 10 Schritte liefern etwa 3 Dezimalstellen, denn

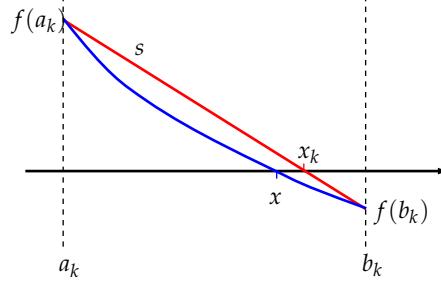
$$\frac{|e_{k+10,r}|}{|e_{k,r}|} \approx \frac{\frac{b_0 - a_0}{|x|} \cdot \frac{1}{2^{k+11}}}{\frac{b_0 - a_0}{|x|} \cdot \frac{1}{2^{k+1}}} = \frac{1}{2^{10}} = \frac{1}{1024}$$

7.2.2 Regula-Falsi-Verfahren

- wir modifizieren das Bisektions-Verfahren, um die Konvergenzgeschwindigkeit zu erhöhen:

- benutze für x_k nicht einfach Intervallmitte $x_k = \frac{a_k + b_k}{2}$, sondern zusätzliche Information:

- verbinde $(a_k, f(a_k)), (b_k, f(b_k))$ durch eine Gerade s



- x_k sei jetzt die Nullstelle der Geraden s :

$$s(x) = f(a_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k} \cdot (x - a_k)$$

$$s(x_k) = 0 \Leftrightarrow x_k = a_k - \frac{b_k - a_k}{f(b_k) - f(a_k)} f(a_k) = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$$

- zusammen erhalten wir das **Regula-Falsi-Verfahren**:

- wähle a_0, b_0 so, dass $f(a_0) \cdot f(b_0) < 0$
- berechne je Schritt:

$$x_k = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$$

und setze

$$\begin{aligned} a_{k+1} &= a_k, & b_{k+1} &= x_k && \text{falls } f(x_k) \cdot f(a_k) < 0, \\ a_{k+1} &= x_k, & b_{k+1} &= b_k && \text{sonst} \end{aligned}$$

Bemerkung 139. Auch das Regula-Falsi-Verfahren ist ein **Einschließungsverfahren**

Beispiel 140.

- wir betrachten wie in Beispiel 138 die Funktion $f(x) = x^2 - 2$ mit $[a_0, b_0] = [1, 2]$, d.h. die exakte Lösung ist $\sqrt{2} \approx 1.41421$

$$\begin{aligned} [a_0, b_0] &= [1, 2] & \Rightarrow & x_0 = \frac{4}{3}, & f(x_0) &= -\frac{2}{9} \\ [a_1, b_1] &= [\frac{4}{3}, 2] & \Rightarrow & x_1 = \frac{7}{5}, & f(x_1) &= -\frac{1}{25} \\ [a_2, b_2] &= [\frac{7}{5}, 2] & \Rightarrow & x_2 = \frac{24}{17}, & f(x_2) &= -\frac{2}{289} \\ [a_3, b_3] &= [\frac{24}{17}, 2] & \Rightarrow & x_3 = \frac{41}{29}, & f(x_3) &= -\frac{1}{841} \end{aligned}$$

so dass $x_3 \approx 1.413793$ drei gültige Stellen besitzt

- zum Vergleich liefert das Bisektions-Verfahren

$$x_0 = \frac{3}{2} \quad x_1 = \frac{5}{4} \quad x_2 = \frac{11}{8} \quad x_3 = \frac{23}{16} \approx 1.4375$$

wobei x_3 einen deutlich größeren Fehler aufweist

Bemerkung 141.

- ◊ im Beispiel ist das Regula-Falsi-Verfahren deutlich genauer als das Bisektions-Verfahren, obwohl das Intervall $[a_k, b_k]$ noch recht groß ist
- ◊ deshalb benutzt man in der Regel die Größe von $|f(x_k)|$ oder $|x_{k+1} - x_k|$ als Abbruchkriterium und nicht die Intervalllänge $|b_k - a_k|$
- Konvergenzgeschwindigkeit:
 - unter einigen zusätzlichen Voraussetzungen an f kann man zeigen, dass

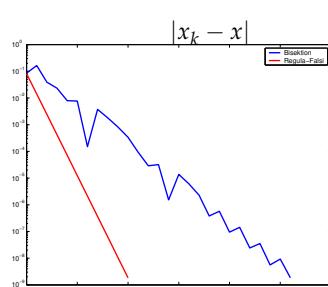
$$|e_{k+1}| \leq \alpha |e_k|, \quad \alpha < 1 \quad (7.2)$$

gilt

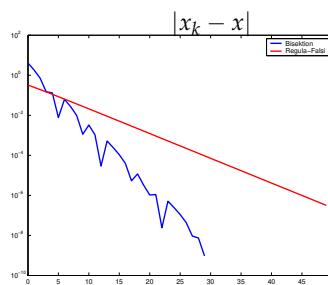
- für $\alpha < \frac{1}{2}$ wäre das Regula-Falsi-Verfahren schneller als das Bisektions-Verfahren
- dies trifft nicht immer zu, wie das folgende Beispiel zeigt

Beispiel 142.

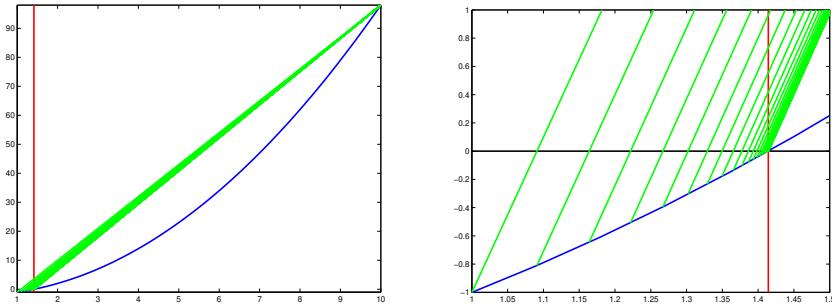
- wir betrachten nochmals $f(x) = x^2 - 2$ mit $[a_0, b_0] = [1, 2]$ (also Nullstelle $x = \sqrt{2}$) und vergleichen den Fehler $|x_k - x|$ für das Bisektions- bzw. Regula-Falsi-Verfahren:



- in diesem Fall ist das Regula-Falsi-Verfahren deutlich schneller
- die Fehlerkurve ist in dem halblogarithmischen Koordinatensystem eine Gerade mit Steigung α aus Gleichung (7.2)
- benutzen wir statt $[a_0, b_0] = [1, 2]$ nun $[a_0, b_0] = [1, 10]$ so erhalten wir



- das Regula-Falsi-Verfahren ist deutlich langsamer
- Grund dafür ist eine Stagnation der a_k



7.2.3 Sekanten-Verfahren

- wir ändern das Regula-Falsi-Verfahren geringfügig ab:
 - seien x_{-2}, x_{-1} gegeben
 - bestimme für $k = 0, 1, \dots$ die neue Näherung x_k als Nullstelle der Geraden durch die Punkte

$$(x_{k-2}, f(x_{k-2})), \quad (x_{k-1}, f(x_{k-1})),$$

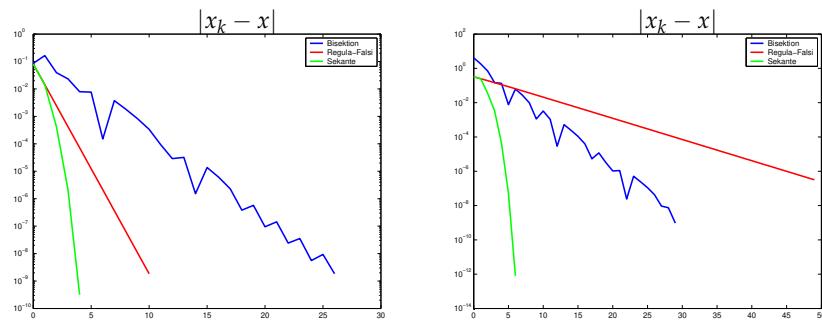
also

$$x_k = x_{k-2} - \frac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})} f(x_{k-2}) = x_{k-1} - \frac{x_{k-1} - x_{k-2}}{f(x_{k-1}) - f(x_{k-2})} f(x_{k-1})$$

- damit erhalten wir das **Sekanten-Verfahren**
- das Sekanten-Verfahren ist *kein* Einschließungsverfahren

Beispiel 143.

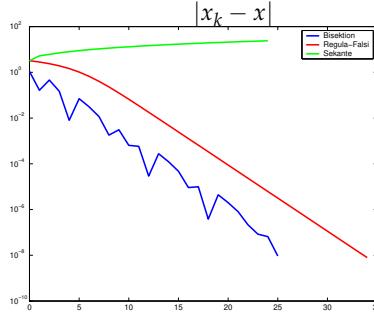
- wir betrachten wieder $f(x) = x^2 - 2$ mit Startintervall $[a_0, b_0]$ für das Bisektions- bzw. Regula-Falsi-Verfahren und $x_{-2} = a_0, x_{-1} = b_0$ für das Sekanten-Verfahren
- für $[a_0, b_0] = [1, 2]$ bzw. $[a_0, b_0] = [1, 10]$ erhalten wir



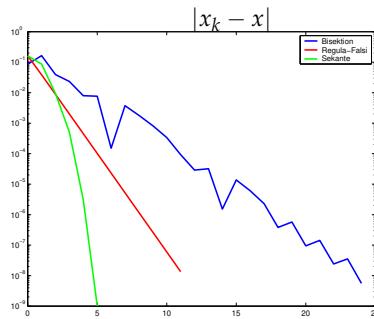
- in beiden Fällen konvergiert das Sekanten-Verfahren am schnellsten
- die Fehler-Kurve fällt stärker als linear im halblogarithmischen Koordinatensystem

Beispiel 144.

- wir betrachten $f(x) = (x^2 - 2)e^{-x}$ mit Startintervall $[0, 5]$ für das Bisektions- bzw. Regula-Falsi-Verfahren und $x_{-2} = 0, x_{-1} = 5$ für das Sekanten-Verfahren



- Bisektions- und Regula-Falsi-Verfahren konvergieren relativ schnell gegen die Nullstelle $x = \sqrt{2}$
- das Sekanten-Verfahren konvergiert nicht gegen eine Nullstelle von f , obwohl die Nullstelle $x = \sqrt{2}$ in $[x_{-2}, x_{-1}]$ liegt (man kann zeigen, dass für $k \geq 1$ alle $x_k > 5$ sind)
- mit Startintervall $[1, 2]$ und $x_{-2} = 1, x_{-1} = 2$ erhalten wir dagegen



- konvergiert das Sekanten-Verfahren, so ist es in der Regel schneller als das Bisektions- und Regula-Falsi-Verfahren
- das Sekanten-Verfahren ist nur lokal konvergent, d.h. es konvergiert nur, wenn die Startwerte genügend nahe an der gesuchten Nullstelle liegen
- für die Konvergenzgeschwindigkeit des Sekanten-Verfahrens erhalten wir folgende Aussage

Satz 145. Ist f zweimal stetig differenzierbar in einer hinreichend kleinen Umgebung X der gesuchten Nullstelle x . Dann konvergiert das Sekanten-Verfahren für alle $x_{-2}, x_{-1} \in X$ und es gibt eine Konstante $c > 0$ so dass

$$|e_k| \leq c |e_{k-1}|^{\frac{1+\sqrt{5}}{2}} \quad (7.3)$$

gilt für $k \rightarrow \infty$

Bemerkung 146.

- ◊ für das Sekantenverfahren gilt

$$|e_k| \leq c |e_{k-1}|^{\frac{1+\sqrt{5}}{2}} = \underbrace{c |e_{k-1}|^{\frac{\sqrt{5}-1}{2}}}_{=\alpha_k} |e_{k-1}|$$

- ◊ konvergiert das Sekantenverfahren ($e_{k-1} \xrightarrow{k \rightarrow \infty} 0$), so folgt wegen $\frac{\sqrt{5}-1}{2} > 0$

$$|e_k| \leq \alpha_k |e_{k-1}|, \quad \alpha_k \xrightarrow{k \rightarrow \infty} 0$$

- ◊ nach (7.1),(7.2) gilt für das Bisektions- bzw. Regula-Falsi-Verfahren

$$|e_{k,r}| \leq \alpha |e_{k-1}|, \quad 0 < \alpha < 1$$

- ◊ deshalb ist das Sekanten-Verfahren (falls es konvergiert) in der Regel schneller als das Bisektions- bzw. Regula-Falsi-Verfahren

7.2.4 Taylor-Reihen-basierte Verfahren, Newton-Verfahren

- ist $f(x) = 0$ und f hinreichend oft differenzierbar, x_k gegeben, so liefert eine Taylor-Entwicklung von f um x_k

$$f(y) = f(x_k) + (y - x_k)f'(x_k) + \frac{(y - x_k)^2}{2}f''(x_k) + \dots$$

bzw. für $y = x$

$$0 = f(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{(x - x_k)^2}{2}f''(x_k) + \dots$$

- Idee:

- breche Taylor-Reihe nach einigen Termen ab
- die abgebrochene Reihe ist ein Polynom, welches die Funktion f in einer Umgebung von x_k approximiert
- bestimme eine Nullstelle \tilde{x} dieses Polynoms und setze

$$x_{k+1} = \tilde{x}$$

Beispiel 147.

- brechen wir nach dem linearen Term ab, dann ist

$$0 = f(x_k) + (\tilde{x} - x_k)f'(x_k) \Rightarrow \tilde{x} = x_k - \frac{f(x_k)}{f'(x_k)}$$

und wir erhalten das **Newton-Verfahren**

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

- brechen wir nach dem quadratischen Term ab so folgt

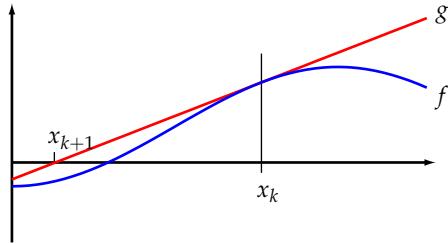
$$0 = f(x_k) + (\tilde{x} - x_k)f'(x_k) + \frac{(\tilde{x} - x_k)^2}{2}f''(x_k)$$

und damit

$$x_{k+1} = x_k - \frac{f'(x_k) \pm \sqrt{(f'(x_k))^2 - 2 \cdot f(x_k)f''(x_k)}}{f''(x_k)}$$

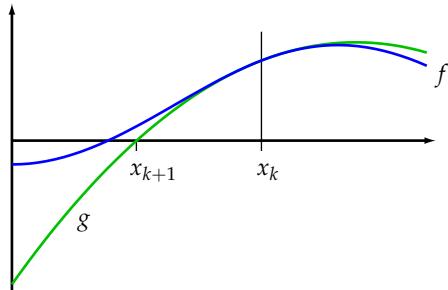
- geometrische Interpretation:

- das Newton-Verfahren approximiert f durch die Tangente g durch den Punkt $(x_k, f(x_k))$ und benutzt die Nullstelle von g als x_{k+1}



- das zweite Verfahren verwendet statt einer Tangente eine Parabel g mit

$$g(x_k) = f(x_k), \quad g'(x_k) = f'(x_k), \quad g''(x_k) = f''(x_k)$$



Bemerkung 148.

- ◊ alle per abgebrochener Taylor-Reihe erzeugten Verfahren sind stationäre Iterations-Verfahren

$$x_{k+1} = \Phi(x_k)$$

- ◊ für das Newton-Verfahren gilt

$$\Phi(x) = x - \frac{f(x)}{f'(x)}$$

Beispiel 149.

► $f(x) = x^2 - 2, x_0 = 1$

► für das Newton-Verfahren ergibt sich die Rekursion

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - 2}{2x_k} = \frac{x_k^2 + 2}{2x_k}$$

und damit

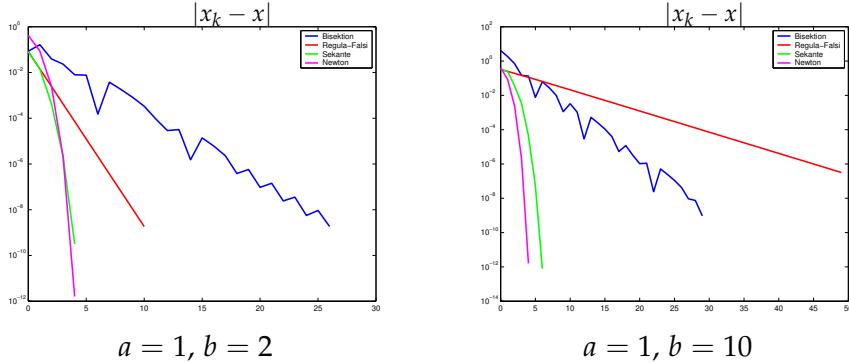
$$\begin{aligned} x_1 &= \frac{1+2}{2} = \frac{3}{2} = 1.5 \\ x_2 &= \frac{(\frac{3}{2})^2 + 2}{3} = \frac{17}{12} \approx 1.417 \\ x_3 &= \frac{577}{408} \approx 1.4142156 \end{aligned}$$

so dass bei x_3 sechs führende Stellen stimmen

- zweites Verfahren mit " $\sqrt{-}$ " vor Wurzel liefert $x_1 = \sqrt{2}$, d.h. nach einem Schritt ist die exakte Lösung erreicht, da die Parabel g mit der Funktion f übereinstimmt

Beispiel 150.

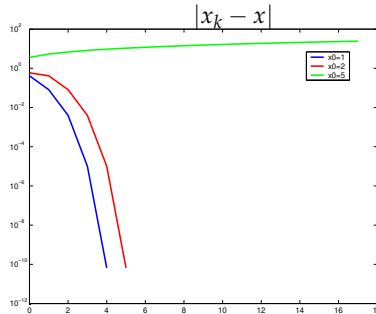
- wir betrachten wieder $f(x) = x^2 - 2$ und vergleichen Bisektions- und Regula-Falsi-Verfahren mit Startintervall $[a, b]$ mit dem Sekanten-Verfahren mit $x_{-2} = a$, $x_{-1} = b$ bzw. dem Newton-Verfahren mit $x_0 = a$
- für $a = 1, b = 2$ bzw. $a = 1, b = 10$ erhalten wir



- das Newton-Verfahren konvergiert am schnellsten

Beispiel 151.

- wir betrachten das Newton-Verfahren für $f(x) = (x^2 - 2)e^{-x}$ mit Startwert $x_0 = 1$, $x_0 = 2$, $x_0 = 5$



- das Newton-Verfahren konvergiert für $x_0 = 5$ nicht
- das Newton-Verfahren ist nur lokal konvergent, d.h. es konvergiert nur, wenn die Startwerte genügend nahe an der gesuchten Nullstelle liegen
- das Newton-Verfahren ist das gebräuchlichste Verfahren für nichtlineare Nullstellenprobleme
- genauere Aussagen über das Konvergenzverhalten leiten wir aus den allgemeinen Betrachtungen des nächsten Kapitels her

Zusammenfassung.

- aus abgebrochener Taylor-Reihe erhalten wir immer eine stationäre Iteration $x_{k+1} = \Phi(x_k)$
- höhere Ordnung \Rightarrow Probleme beim Auflösen (mehrere Nullstellen), müssen höhere Ableitungen von f berechnen
- in der Praxis verwendet man fast ausschließlich das Newton-Verfahren
- Konvergenz, Konvergenzgeschwindigkeit werden mit Hilfe der allgemeinen Betrachtungen des nächsten Kapitels untersucht

7.3 Stationäre Iterationen, Banachscher Fixpunktsatz

- wir betrachten allgemeine stationäre Iterationen $x_{k+1} = \Phi(x_k)$ und leiten Aussagen über deren Konvergenz und Konvergenzgeschwindigkeit her
- wir hatten bei linearen Gleichungssystemen schon einmal solche Verfahren betrachtet
 - wollten $f(x) = b - Ax = 0$ lösen
 - haben es in Fixpunktproblem

$$x = \Phi(x) = Bx + d$$

umgewandelt so dass

$$x = \Phi(x) \Leftrightarrow b - Ax = 0$$

- $x = \Phi(x)$ haben wir dann iterativ mit $x_{k+1} = \Phi(x_k)$ gelöst
- Konvergenz $\Leftrightarrow \rho(B) < 1$
- Geschwindigkeit wird durch $\rho(B)$ bestimmt
- damit haben wir also Nullstellen einer linear-affinen Abbildung

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad f(y) = b - Ax$$

bestimmt

- wir wollen jetzt allgemeiner $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ nichtlinear betrachten und analog vorgehen
- wandeln Nullstellenproblem so in Fixpunktproblem um, dass gilt

$$x = \Phi(x) \Rightarrow f(x) = 0$$

- wegen Mehrdeutigkeit von $f(x) = 0$ muss Umkehrung (\Leftrightarrow) nicht notwendig gelten, d.h. es kann auch nur *eine* Nullstelle Fixpunkt sein
- da f nichtlinear ist, hat Φ nicht mehr die einfache Gestalt $Bx + d$ wie im linearen Fall
- wir betrachten wieder zu gegebenem x_0 die **Picard-Iteration**

$$x_{k+1} = \Phi(x_k)$$

Beispiel 152.

- ▶ Newton-Verfahren $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$, $\Phi(y) = y - \frac{f(y)}{f'(y)}$ ist im allgemeinen nichtlinear
- ▶ Fixpunkte erhalten wir falls

$$x = \Phi(x) = x - \frac{f(x)}{f'(x)} \Leftrightarrow \frac{f(x)}{f'(x)} = 0,$$

also nur wenn wie gewünscht $f(x) = 0$

- aus der Fixpunkteigenschaft folgt für den Fehler

$$e_{k+1} = x - x_{k+1} = \Phi(x) - \Phi(x_k)$$

- wie verhält sich e_k für $k \rightarrow \infty$?
- wir betrachten zunächst wieder den linearen Fall $\Phi(y) = By + d$

- wegen

$$e_{k+1} = Bx + d - (Bx_k + d) = B(x - x_k) = Be_k$$

ist

$$\|e_{k+1}\| \leq \|B\| \|e_k\|$$

und $\|B\| < 1$ ist ein hinreichendes Kriterium um Konvergenz des Verfahrens $\|e_k\| \xrightarrow{k \rightarrow \infty} 0$ für alle Startwerte $x_0 \in \mathbb{R}^n$ zu erhalten

- im eindimensionalen Fall ist

$$\Phi : \mathbb{R} \rightarrow \mathbb{R}, \quad \Phi(y) = By + d, \quad B, d \in \mathbb{R}$$

und die Bedingung $\|B\| < 1$ bedeutet

$$\|B\| = |B| = |\Phi'(\xi)| < 1 \quad \forall \xi \in \mathbb{R}$$

- ist f nichtlinear, so ist Φ in der Regel auch nichtlinear

- wir betrachten den eindimensionalen Fall $\Phi : \mathbb{R} \rightarrow \mathbb{R}$
- ist Φ differenzierbar, dann folgt aus dem Mittelwertsatz

$$e_{k+1} = \Phi(x) - \Phi(x_k) = \Phi'(\xi_k)(x - x_k) = \Phi'(\xi_k)e_k$$

mit $\xi_k \in \text{co}(x, x_k)$ und somit

$$|e_{k+1}| \leq |\Phi'(\xi_k)| \cdot |e_k|$$

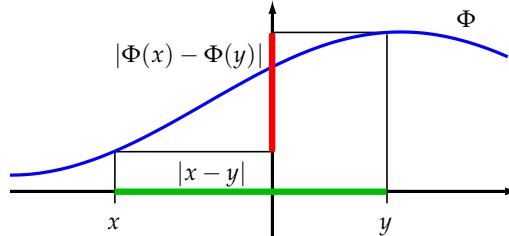
- für $|\Phi'(\xi_k)| \leq \alpha < 1 \ \forall k$, α unabhängig von k , folgt

$$|e_k| \xrightarrow{k \rightarrow \infty} 0$$

- geometrisch folgt aus $|\Phi'| < 1$

$$|\Phi(x) - \Phi(y)| < |x - y|,$$

d.h. Φ kontrahiert



- im Gegensatz zum linearen Fall ist hier $\Phi'(\xi)$ nicht konstant für alle $\xi \in \mathbb{R}$, d.h. Φ kann in einer Teilmenge des \mathbb{R} kontrahieren, in einer anderen Teilmenge dagegen nicht
- jetzt verallgemeinern wir diese Überlegungen auf \mathbb{R}^n

Definition 153.

- $x_k, x \in \mathbb{R}^n$, x_k konvergiert gegen x in $\|\cdot\|$ falls $\lim_{k \rightarrow \infty} \|x_k - x\| = 0$
- Φ heißt **Kontraktion** bezüglich $\|\cdot\|$ auf $X \subset \mathbb{R}^n$ falls
 - (1) $\Phi : X \rightarrow X$
 - (2) $\|\Phi(x) - \Phi(y)\| \leq \alpha \|x - y\| \quad \forall x, y \in X$ und $\alpha < 1$ unabhängig von x, y
- die Iteration $x_{k+1} = \Phi(x_k)$ heißt **lokal konvergent** gegen $x \in X \subset \mathbb{R}^n$ falls

$$\lim_{k \rightarrow \infty} x_k = x \quad \forall x_0 \in X$$

Bemerkung 154. Ob Φ eine Kontraktion ist, hängt auch von der benutzten Norm ab

- der folgende Satz liefert ein hinreichendes Konvergenzkriterium

Satz 155 (Banachscher Fixpunktsatz). $X \subset \mathbb{R}^n$ abgeschlossen, $\Phi : X \rightarrow X$ eine Kontraktion auf X mit

$$\|\Phi(x) - \Phi(y)\| \leq \alpha \|x - y\| \quad \forall x, y \in X,$$

$\alpha < 1$ unabhängig von x, y . Dann hat Φ genau einen Fixpunkt $x = \Phi(x)$ in X . Die Picard-Iteration

$$x_{k+1} = \Phi(x_k)$$

konvergiert $\forall x_0 \in X$ gegen x . Es gelten die Abschätzungen

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

Bemerkung 156.

- ◊ ist Φ eine Kontraktion, so existiert genau ein Fixpunkt
- ◊ x_k konvergiert gegen x
- ◊ die Kontraktionseigenschaft ist hinreichend für Konvergenz, es kann aber auch konvergente Iterationen geben, die *keine* Kontraktionen sind
- ◊ oft ist $\Phi(X) \subset X$ die kritische Voraussetzung, nicht so sehr $\alpha < 1$
- ◊ in der Regel ist $X \neq \mathbb{R}^n$, d.h. man erhält nur lokale Konvergenz
- ◊ die beiden Abschätzungen liefern Fehlerschranken, ohne dass man x kennt
- ◊ die Fehlerschranken gelten bezüglich der Norm $\|\cdot\|$ für die Φ eine Kontraktion ist
- ◊ a-priori Abschätzung:
 - ◊ brauche nur α, x_0, x_1 und kann dann $\|e_k\|$ abschätzen, ohne x_k ausrechnen zu müssen
 - ◊ wird benutzt um die maximale Zahl der Iterationen für eine gegebene Genauigkeit zu bestimmen
 - ◊ oft sehr grob
- ◊ a-posteriori Abschätzung:
 - ◊ schätzt $\|e_k\|$ ab über α, x_k, x_{k-1} , d.h. erst wenn x_k ausgerechnet wurde
 - ◊ liefert in der Regel genauere Abschätzung für $\|e_k\|$
 - ◊ wird oft als Abbruchkriterium benutzt

Beweis.

- (1) Konvergenz der Iteration:

- o es gilt

$$\|x_{k+1} - x_k\| = \|\Phi(x_k) - \Phi(x_{k-1})\| \leq \alpha \|x_k - x_{k-1}\| \leq \alpha^k \|x_1 - x_0\| \quad (7.4)$$

- o für $l > k$ folgt

$$\begin{aligned} \|x_l - x_k\| &= \|x_l - x_{l-1} + x_{l-1} - x_{l-2} + \dots + x_{k+1} - x_k\| \\ &\leq \|x_l - x_{l-1}\| + \dots + \|x_{k+1} - x_k\| \\ &\stackrel{(7.4)}{\leq} \alpha^{l-1} \|x_1 - x_0\| + \dots + \alpha^k \|x_1 - x_0\| \\ &= \alpha^k (1 + \alpha + \alpha^2 + \dots + \alpha^{l-k-1}) \|x_1 - x_0\| \\ &= \alpha^k \frac{1 - \alpha^{l-k}}{1 - \alpha} \|x_1 - x_0\| \\ &\leq \frac{\alpha^k}{1 - \alpha} \|x_1 - x_0\| \end{aligned} \quad (7.5)$$

- o ist k groß genug, so wird $\|x_l - x_k\| \forall l \geq k$ beliebig klein, so dass x_k Cauchyfolge in $X \subset \mathbb{R}^n$ ist
- o \mathbb{R}^n ist vollständig, weshalb $x_k \xrightarrow{k \rightarrow \infty} x \in \mathbb{R}^n$
- o da $x_k \in X$, X abgeschlossen, folgt $x \in X$

(2) Grenzwert x ist Fixpunkt:

- o mit der Dreiecksungleichung und der Kontraktionseigenschaft folgt

$$\begin{aligned} \|x - \Phi(x)\| &= \|x - x_k + x_k - \Phi(x)\| \\ &\leq \|x - x_k\| + \|\Phi(x_{k-1}) - \Phi(x)\| \\ &\leq \|x - x_k\| + \alpha \|x_{k-1} - x\| \end{aligned}$$

- o wegen $x_k \xrightarrow{k \rightarrow \infty} x$ ist

$$\|x - x_k\| \xrightarrow{k \rightarrow \infty} 0, \quad \|x_{k-1} - x\| \xrightarrow{k \rightarrow \infty} 0,$$

- so dass die rechte Seite der letzten Ungleichung für $k \rightarrow \infty$ gegen 0 konvergiert
- damit ist $\|x - \Phi(x)\| = 0$ und $x = \Phi(x)$

(3) x ist der einzige Fixpunkt:

- o Annahme: es gibt einen zweiten Fixpunkt $y \in X$ mit $y = \Phi(y)$
- o dann ist

$$\begin{aligned} \|x - y\| &= \|\Phi(x) - \Phi(y)\| \leq \alpha \|x - y\| \\ \Rightarrow \quad (1 - \alpha) \|x - y\| &\leq 0 \\ \stackrel{1-\alpha>0}{\Rightarrow} \quad \|x - y\| &= 0 \end{aligned}$$

und damit $x = y$

(4) a-priori Abschätzung:

- o aus (7.5) wissen wir:

$$\|x_l - x_k\| \leq \frac{\alpha^k}{1 - \alpha} \|x_1 - x_0\|$$

- o mit $l \rightarrow \infty$ folgt wegen $x_l \xrightarrow{l \rightarrow \infty} x$

$$\|x - x_k\| \leq \frac{\alpha^k}{1 - \alpha} \|x_1 - x_0\|$$

(5) a-posteriori Abschätzung:

- nach (7.4) ist

$$\|x_{k+1} - x_k\| \leq \alpha \|x_k - x_{k-1}\|$$

und

$$\begin{aligned} \|x_l - x_k\| &\leq \|x_l - x_{l-1}\| + \dots + \|x_{k+2} - x_{k+1}\| + \|x_{k+1} - x_k\| \\ &\leq \alpha^{l-k} \|x_k - x_{k-1}\| + \dots + \alpha^2 \|x_k - x_{k-1}\| + \alpha \|x_k - x_{k-1}\| \\ &= \alpha(1 + \alpha + \alpha^2 + \dots + \alpha^{l-k-1}) \|x_k - x_{k-1}\| \\ &= \alpha \frac{1 - \alpha^{l-k}}{1 - \alpha} \|x_k - x_{k-1}\| \\ &\leq \frac{\alpha}{1 - \alpha} \|x_k - x_{k-1}\| \end{aligned}$$

- mit $l \rightarrow \infty$ folgt wegen $x_l \xrightarrow{l \rightarrow \infty} x$ die Behauptung

□

Beispiel 157.

- wir betrachten das Newton-Verfahren für $f(x) = x^2 - 2$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - 2}{2x_k} = \frac{x_k}{2} + \frac{1}{x_k}$$

also

$$\Phi(y) = \frac{y}{2} + \frac{1}{y}$$

- Fixpunkte:

$$\begin{aligned} \bullet \quad x = \Phi(x) &\Leftrightarrow x = \frac{x}{2} + \frac{1}{x} \Leftrightarrow \frac{x}{2} = \frac{1}{x} \Leftrightarrow x^2 = 2 \Leftrightarrow x^2 - 2 = 0 \\ \bullet \quad \text{in diesem Fall stimmen die Fixpunkte von } \Phi \text{ mit den Nullstellen von } f \text{ überein} \end{aligned}$$

- Φ ist Kontraktion auf $X = [1, 2]$:

- $\Phi(X) \subset X$, denn für $y \in [1, 2]$ gilt

$$\Phi(y) = \frac{y}{2} + \frac{1}{y} \geq \frac{1}{2} + \frac{1}{2} = 1, \quad \Phi(y) \leq \frac{2}{2} + \frac{1}{1} = 2$$

- Φ ist auf X differenzierbar mit $\Phi'(y) = \frac{1}{2} - \frac{1}{y^2}$, so dass $\forall \xi \in X$ gilt

$$\Phi'(\xi) = \frac{1}{2} - \frac{1}{\xi^2} \geq \frac{1}{2} - \frac{1}{1} \geq -\frac{1}{2}, \quad \Phi'(\xi) \leq \frac{1}{2} - \frac{1}{4} \leq \frac{1}{4}$$

- mit dem Mittelwertsatz folgt mit $\xi \in \text{co}(x, y)$

$$|\Phi(x) - \Phi(y)| = |\Phi'(\xi)| \cdot |x - y| \leq \frac{1}{2} |x - y| \quad \forall x, y \in X = [1, 2]$$

- damit ist Φ eine Kontraktion auf X mit $\alpha = \frac{1}{2}$ und Φ hat genau einen Fixpunkt in X
- nach dem Banachschen Fixpunktsatz haben wir damit für $x_0 \in [1, 2]$ garantierte (lokale) Konvergenz

- Fehlerabschätzungen

► a-priori Abschätzung

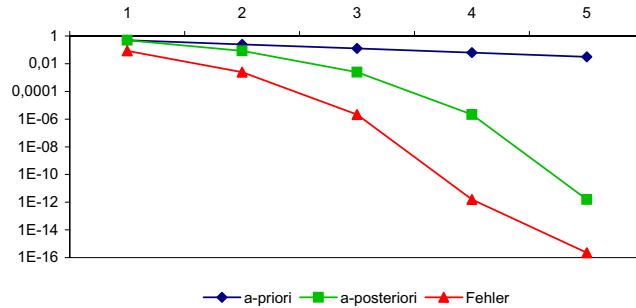
$$|e_k| = |x - x_k| \leq \frac{\alpha^k}{1-\alpha} |x_1 - x_0| = \left(\frac{1}{2}\right)^{k-1} |x_1 - x_0|$$

► a-posteriori Abschätzung

$$|e_k| \leq \frac{\alpha}{1-\alpha} |x_k - x_{k-1}| = |x_k - x_{k-1}|$$

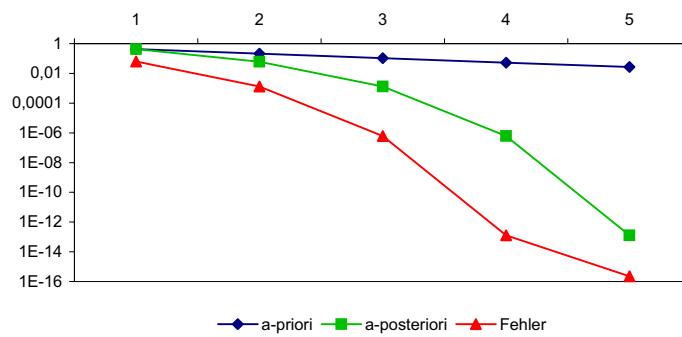
► für $x_0 = 1$ ist $x_1 = \frac{3}{2}$ und wir erhalten

k	xk	a-priori	a-posteriori	Fehler
0	1,000000			4,142E-01
1	1,500000	5,000E-01	5,000E-01	8,579E-02
2	1,416667	2,500E-01	8,333E-02	2,453E-03
3	1,414216	1,250E-01	2,451E-03	2,124E-06
4	1,414214	6,250E-02	2,124E-06	1,595E-12
5	1,414214	3,125E-02	1,595E-12	2,220E-16



► analog ergibt sich für $x_0 = 1.9$

k	xk	a-priori	a-posteriori	Fehler
0	1,900000			4,858E-01
1	1,476316	4,237E-01	4,237E-01	6,210E-02
2	1,415520	2,118E-01	6,080E-02	1,306E-03
3	1,414214	1,059E-01	1,306E-03	6,026E-07
4	1,414214	5,296E-02	6,026E-07	1,283E-13
5	1,414214	2,648E-02	1,286E-13	2,220E-16



► a-posteriori Ergebnisse sind sehr viel genauer

► Konvergenzverhalten:

- haben bis jetzt lokale Konvergenz für $x_0 \in [1, 2]$
- eine genauere Analyse (ohne Banach) liefert für den untersuchten Spezialfall

$$x_0 > 0 \quad \Rightarrow \quad x_k \xrightarrow{k \rightarrow \infty} \sqrt{2}, \quad x_0 < 0 \quad \Rightarrow \quad x_k \xrightarrow{k \rightarrow \infty} -\sqrt{2}$$

- auf $(-\sqrt{\frac{3}{2}}, 0)$ und $(0, \sqrt{\frac{3}{2}})$ ist Φ keine Kontraktion, konvergiert aber trotzdem
- Banach liefert also nur hinreichende, keine notwendigen Konvergenzkriterien

Beispiel 158.

- wir betrachten jetzt $f(x) = x^2 - 2$ und als Iteration $\Phi(x) = x - \frac{1}{2}f(x)$

- Fixpunkte:

- $x = \Phi(x) \Leftrightarrow x = x - \frac{1}{2}f(x) \Leftrightarrow f(x) = 0$

- Φ Kontraktion auf $[1, \frac{3}{2}]$:

- es gilt

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

und damit

$$\Phi(x) \leq \frac{3}{2} \quad \forall x \in [1, \frac{3}{2}]$$

- für $x \in [1, \frac{3}{2}]$ ist

$$\Phi(x) \geq \Phi\left(\frac{3}{2}\right) = \frac{1}{2}\left(3 - \frac{1}{4}\right) = \frac{3}{2} - \frac{1}{8} > 1$$

- außerdem gilt

$$|\Phi(x) - \Phi(y)| = \left|x - y - \frac{1}{2}(x^2 - y^2)\right| = \left|(1 - \frac{x+y}{2})(x-y)\right| = \left|1 - \frac{x+y}{2}\right| |x-y|$$

- für $x \in [1, \frac{3}{2}]$ ist also

$$1 - \frac{x+y}{2} \geq 1 - \frac{\frac{3}{2} + \frac{3}{2}}{2} = -\frac{1}{2}, \quad 1 - \frac{x+y}{2} \leq 1 - \frac{1+1}{2} = 0$$

und damit

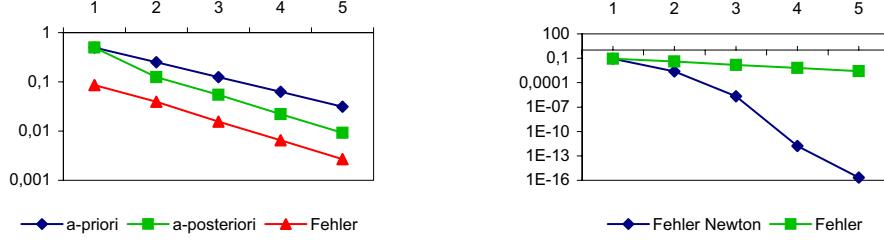
$$\alpha = \max_{x,y \in [1, \frac{3}{2}]} \left|1 - \frac{x+y}{2}\right| = \frac{1}{2},$$

d.h. Φ ist eine Kontraktion und die Iteration konvergiert $\forall x_0 \in [1, \frac{3}{2}]$

- Fehlerabschätzungen:

- die Fehlerabschätzungen verlaufen analog zum letzten Beispiel

k	xk	a-priori	a-posteriori	Fehler
0	1,000000			4,142E-01
1	1,500000	5,000E-01	5,000E-01	8,579E-02
2	1,375000	2,500E-01	1,250E-01	3,921E-02
3	1,429688	1,250E-01	5,469E-02	1,547E-02
4	1,407684	6,250E-02	2,200E-02	6,529E-03
5	1,416897	3,125E-02	9,212E-03	2,683E-03



► Konvergenzverhalten:

- ein Vergleich zu Newton für $x_0 = 1$ zeigt, dass das neue Verfahren trotz gleicher Kontraktionszahl α viel langsamer konvergiert
- wir erhalten lokale Konvergenz auf $[1, \frac{3}{2}]$ nach dem Banachschen Fixpunktsatz
- für $x_k \leq -2$ folgt

$$x_{k+1} = \Phi(x_k) = x_k - \underbrace{\frac{x_k^2 - 2}{2}}_{\geq 0} \leq x_k \leq -2$$

womit x_k nicht gegen eine Nullstelle von $f(x) = x^2 - 2$ konvergieren kann

- eine analoge Aussage folgt für $x_k \geq 4$
- damit ist die Konvergenz *echt* lokal

Beispiel 159.

- der Banachsche Fixpunktsatz kann auch auf die linearen stationären Iterationen aus Kapitel 5 angewandt werden
- wir betrachten das Jacobi-Verfahren für das System $Ax = b$,

$$A = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 6 \\ 5 \end{pmatrix},$$

d.h. $x = (1, 2)^T$ ist die exakte Lösung

- für die Jacobi-Iteration erhalten wir

$$x_{k+1} = \Phi(x_k) = Bx_k + d$$

mit

$$B = \begin{pmatrix} 0 & -\frac{1}{4} \\ -\frac{1}{2} & 0 \end{pmatrix}$$

- wir wissen, dass $x = \Phi(x) \Leftrightarrow Ax = b$

- Φ ist bzgl. der Norm $\|\cdot\|_\infty$ eine Kontraktion in ganz \mathbb{R}^n , da
$$\|\Phi(y) - \Phi(x)\|_\infty = \|By + d - (Bx + d)\|_\infty = \|B(y - x)\|_\infty \leq \|B\|_\infty \|y - x\|_\infty \quad \forall x, y \in \mathbb{R}^n$$
und damit $\alpha = \|B\|_\infty = \frac{1}{2}$

- das Verfahren konvergiert also für alle $x_0 \in \mathbb{R}^n$ und wir erhalten folgende Fehlerabschätzungen

- a-priori Abschätzung

$$\|e_k\|_\infty = \|x - x_k\|_\infty \leq \frac{\alpha^k}{1-\alpha} \|x_1 - x_0\|_\infty = \left(\frac{1}{2}\right)^{k-1} \|x_1 - x_0\|_\infty$$

- a-posteriori Abschätzung

$$\|e_k\|_\infty \leq \frac{\alpha}{1-\alpha} \|x_k - x_{k-1}\|_\infty = \|x_k - x_{k-1}\|_\infty$$

- aus der a-priori Abschätzung folgt mit $x_0 = (0, 0)^T$, $x_1 = (\frac{3}{2}, \frac{5}{2})^T$

$$\|e_k\|_\infty \leq \left(\frac{1}{2}\right)^{k-1} \|x_1 - x_0\|_\infty = \left(\frac{1}{2}\right)^{k-1} \frac{5}{2},$$

für $k = 10$, also

$$e_{10} \leq \frac{5}{1024}$$

- aus der a-posteriori Abschätzung für $k = 10$ folgt mit $x_9 = (\frac{8193}{8192}, \frac{16385}{8192})^T$, $x_{10} = (\frac{32767}{32768}, \frac{32767}{16384})^T$

$$\|e_{10}\|_\infty \leq \|x_{10} - x_9\|_\infty = \frac{3}{16384}$$

- der exakte Wert ist

$$\|e_{10}\|_\infty = \|x_{10} - x\|_\infty = \left\| \begin{pmatrix} \frac{32767}{32768} \\ \frac{32767}{16384} \end{pmatrix} - \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\|_\infty = \frac{1}{16384}$$

Bemerkung 160.

- ◊ wir haben für das selbe Nullstellenproblem zwei stationäre Iterationen mit dem selben Kontraktionsfaktor α , aber sehr unterschiedlicher Konvergenzgeschwindigkeit
- ◊ auch das Konvergenzverhalten ist völlig verschieden (global/lokal)
- jetzt betrachten wir die Konvergenzgeschwindigkeit genauer

Definition 161. Die Iteration $x_{k+1} = \Phi(x_k)$ heißt

- **linear konvergent** falls es eine Konstante $0 \leq c < 1$ unabhängig von k gibt mit

$$\|e_{k+1}\| \leq c \|e_k\| \quad \forall k$$

- **konvergent mit Ordnung m** falls es eine Konstante $0 \leq c$ unabhängig von k gibt mit

$$\lim_{k \rightarrow \infty} \|e_k\| = 0 \quad \text{und} \quad \|e_{k+1}\| \leq c \|e_k\|^m \quad \forall k$$

Bemerkung 162.

- ◊ ist $m > 1$ so nennt man Φ **superlinear konvergent**
- ◊ für $m = 2$ erhalten wir **quadratische Konvergenz**
- ◊ wegen (7.1) ist das Bisektionsverfahren linear konvergent mit $c = \frac{1}{2}$
- ◊ das Regula-Falsi-Verfahren ist ebenfalls linear konvergent (siehe (7.2))
- ◊ aus (7.3) folgt, dass das Sekanten-Verfahren Konvergenzordnung

$$m = \frac{1 + \sqrt{5}}{2} \approx 1.618$$

hat

- ist Φ stetig differenzierbar, so kann man allgemeine Aussagen über die Konvergenzordnung treffen
- wir beschränken uns hier auf den eindimensionalen Fall $\Phi : \mathbb{R} \rightarrow \mathbb{R}$
- die Ergebnisse lassen sich entsprechend auf den höherdimensionalen Fall verallgemeinern

Satz 163. Ist $m \geq 2$, x ein Fixpunkt von $\Phi : \mathbb{R} \rightarrow \mathbb{R}$ und Φ m -mal stetig differenzierbar in x mit

$$\Phi'(x) = \Phi''(x) = \dots = \Phi^{(m-1)}(x) = 0,$$

dann gibt es ein abgeschlossenes Intervall $X = [x - \delta, x + \delta]$, $\delta > 0$, so dass die Iteration

$$x_{k+1} = \Phi(x_k)$$

für alle $x_0 \in X$ mindestens mit Ordnung m konvergiert

Beweis.

- ◊ da Φ stetig differenzierbar ist, gibt es wegen $\Phi'(x) = 0$ ein Intervall $X = [x - \delta, x + \delta]$ mit $\delta > 0$ und

$$|\Phi'(\xi)| \leq \alpha < 1 \quad \forall \xi \in X,$$

wobei α unabhängig von ξ ist

- ◊ aus dem Mittelwertsatz folgt für ein beliebiges $y \in X$

$$|\Phi(y) - \Phi(x)| = |\Phi'(\xi)| |y - x| \leq \alpha |y - x|, \quad \xi \in \text{co}(x, y) \quad (7.6)$$

- ◊ wegen $x = \Phi(x)$ erhalten wir aus (7.6) auch

$$|\Phi(y) - x| = |\Phi(y) - \Phi(x)| \leq \alpha |y - x| \leq |y - x| \leq \delta,$$

d.h. liegt y in X dann gilt dies auch für $\Phi(y)$

- ◊ damit ist Φ eine Kontraktion in X und die Folge der x_k konvergiert nach dem Banachschen Fixpunktsatz gegen x

- wir entwickeln nun Φ in eine Taylor-Reihe um den Fixpunkt x

$$\begin{aligned}\Phi(y) &= \Phi(x) + \Phi'(x)(y-x) + \frac{1}{2!}\Phi''(x)(y-x)^2 + \dots \\ &\quad + \frac{1}{(m-1)!}\Phi^{(m-1)}(x)(y-x)^{m-1} + \frac{1}{m!}\Phi^{(m)}(\xi)(y-x)^m\end{aligned}$$

mit $\xi \in \text{co}(x, y)$

- nach den Voraussetzungen des Satzes sollen die ersten $m-1$ Ableitungen von Φ an der Stelle x verschwinden, d.h.

$$\Phi(y) = \Phi(x) + \frac{1}{m!}\Phi^{(m)}(\xi)(y-x)^m$$

- setzen wir $y = x_k$ und benutzen $x = \Phi(x)$ so folgt

$$x_{k+1} = \Phi(x_k) = x + \frac{1}{m!}\Phi^{(m)}(\xi_k)(x_k - x)^m, \quad \xi_k \in \text{co}(x, x_k)$$

bzw.

$$|e_{k+1}| = |x_{k+1} - x| \leq \frac{1}{m!}|\Phi^{(m)}(\xi_k)| |x_k - x|^m = \frac{1}{m!}|\Phi^{(m)}(\xi_k)| |e_k|^m \quad (7.7)$$

- Φ ist eine Kontraktion in X , d.h. für $x_0 \in X$ gilt $x_k \in X$
- damit gilt auch $\xi_k \in \text{co}(x, x_k) \subset X$
- $\Phi^{(m)}$ ist nach Voraussetzung stetig und damit auf der abgeschlossenen Menge X beschränkt, weshalb

$$|\Phi^{(m)}(\xi_k)| \leq \tilde{c} \quad \forall k,$$

\tilde{c} unabhängig von k

- mit $c = \frac{1}{m!}\tilde{c}$ erhalten wir damit aus (7.7)

$$|e_{k+1}| \leq c |e_k|^m \quad \forall k,$$

d.h. die Folge der x_k konvergiert mindestens mit Ordnung m

□

- analog zeigt man das folgende Ergebnis

Satz 164. Ist $m \geq 2$, x ein Fixpunkt von $\Phi : \mathbb{R} \rightarrow \mathbb{R}$ und Φ $(m+1)$ -mal stetig differenzierbar in x mit

$$\Phi'(x) = \Phi''(x) = \dots = \Phi^{(m-1)}(x) = 0, \quad \Phi^{(m)}(x) \neq 0,$$

dann gibt es ein abgeschlossenes Intervall $X = [x - \delta, x + \delta]$, $\delta > 0$, so dass die Iteration

$$x_{k+1} = \Phi(x_k)$$

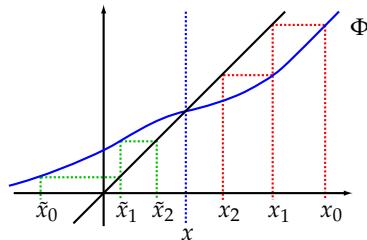
für alle $x_0 \in X$ genau mit Ordnung m konvergiert

- geometrisches Konvergenzverhalten

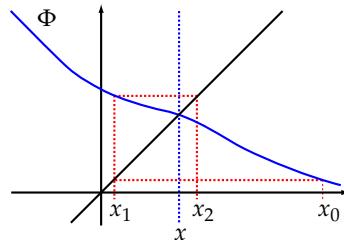
- x sei ein Fixpunkt von Φ , Φ sei in einem abgeschlossenen Intervall $X = [x - \delta, x + \delta]$, $\delta > 0$, stetig differenzierbar mit

$$|\Phi'(\xi)| \leq \alpha < 1 \quad \forall \xi \in X$$

- wie im Beweis von Satz 163 folgt daraus, dass Φ eine Kontraktion auf X ist und die Iteration $x_{k+1} = \Phi(x_k)$ für alle $x_0 \in X$ gegen den Fixpunkt x konvergiert
- hat Φ' in ganz X dasselbe Vorzeichen, so können wir Aussagen über das Konvergenzverhalten der x_k treffen
- für $0 < \Phi'(\xi) < 1 \forall \xi \in X$ erhalten wir monoton fallende oder monoton wachsende Folgen x_k



- für $-1 < \Phi'(\xi) < 0 \forall \xi \in X$ erhalten wir Folgen x_k , die um den Fixpunkt x alternieren



- schließlich brauchen wir noch ein Abbruchkriterium für die Picard-Iteration:

$$(1) \quad |x_{k+1} - x_k| < \varepsilon$$

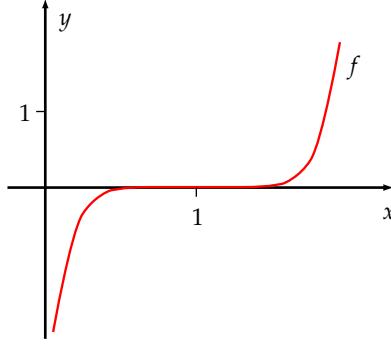
- die a-posteriori Abschätzung des Satzes von Banach liefert

$$|e_{k+1}| \leq \frac{\alpha}{1-\alpha} |x_{k+1} - x_k|$$

- ist α nicht zu nahe bei 1, so liefert $|x_{k+1} - x_k|$ eine brauchbare Abschätzung für $|e_{k+1}|$

$$(2) \quad |f(x_k)| < \varepsilon$$

- problematisch bei schlecht konditionierten Problemen



- selbst wenn $|x_k - x|$ "groß" ist, kann $|f(x_k)|$ schon sehr klein sein

(3) Schätzung $\hat{\alpha}$ von α

- nach dem Satz von Banach gilt $|e_{k+1}| \leq \frac{\alpha}{1-\alpha} |x_{k+1} - x_k|$
- α ist in der Regel unbekannt bzw. mühsam zu bestimmen
- versuche α durch $\hat{\alpha}$ zu schätzen
- ist Φ differenzierbar, so gilt $|\Phi'(\xi)| \leq \alpha$
- setze

$$\alpha \approx |\Phi'(x_k)| \approx \left| \frac{\Phi(x_k) - \Phi(x_{k-1})}{x_k - x_{k-1}} \right| = \left| \frac{x_{k+1} - x_k}{x_k - x_{k-1}} \right| =: \hat{\alpha}$$

- benutze dann

$$|\hat{e}_{k+1}| := \frac{\hat{\alpha}}{1 - \hat{\alpha}} |x_{k+1} - x_k|$$

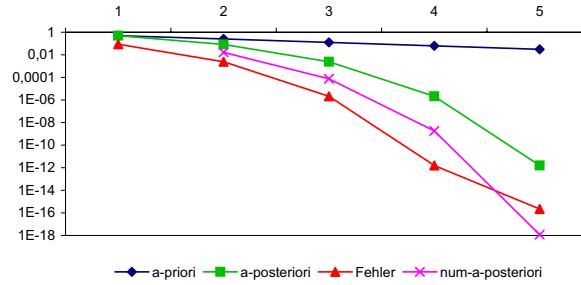
als Schätzer für $|e_{k+1}|$

- breche ab, wenn $|\hat{e}_k| \leq \varepsilon$
- $\hat{\alpha}$ wird in jedem Schritt neu berechnet, weshalb diese Art der Fehlerschätzung oft genauere Ergebnisse liefert als die a-posteriori Formel mit festem α

Beispiel 165.

- wir betrachten das Newton-Verfahren für $f(x) = x^2 - 2$, $x_0 = 1$

k	x _k	a-priori	a-posteriori	Fehler	num-a-post.
0	1,00000				0,000E+00
1	1,50000	5,000E-01	5,000E-01	8,579E-02	0,000E+00
2	1,416667	2,500E-01	8,333E-02	2,453E-03	1,667E-02
3	1,414216	1,250E-01	2,451E-03	2,124E-06	7,427E-05
4	1,414214	6,250E-02	2,124E-06	1,595E-12	1,842E-09
5	1,414214	3,125E-02	1,595E-12	2,220E-16	1,198E-18



- $|\hat{e}_k|$ besser als Schranke mit festem α
- Genauigkeitsprobleme durch Division, wenn $|x_k - x_{k-1}|$ sehr klein ist (siehe $k = 5$)

7.4 Newton-Verfahren

- wir benutzen die Ergebnisse von Kapitel 7.3 um das Newton-Verfahren

$$x_{k+1} = \Phi(x_k), \quad \Phi(y) = y - \frac{f(y)}{f'(y)}$$

genauer zu untersuchen

- für den Fixpunkt $x = \Phi(x)$ gilt $f(x) = 0$
- um eine möglichst hohe Konvergenzgeschwindigkeit zu erhalten, sollen nach Satz 163 möglichst viele Ableitungen von Φ im Fixpunkt $x = \Phi(x)$ verschwinden:

$$\Phi'(x) = \left(x - \frac{f(x)}{f'(x)} \right)' = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2} \quad (7.8)$$

$$\Phi''(x) = \frac{(f'(x))^3 f''(x) + f(x)(f'(x))^2 f'''(x) - 2f(x)f'(x)(f''(x))^2}{(f'(x))^4} \quad (7.9)$$

- ist $f'(x) \neq 0$ (damit Division keine Probleme macht), so ist $\Phi'(x) = 0$ und $\Phi''(x) = \frac{f''(x)}{f'(x)}$ im allgemeinen ungleich 0
- damit können wir Satz 163 mit $m = 2$ anwenden und erhalten

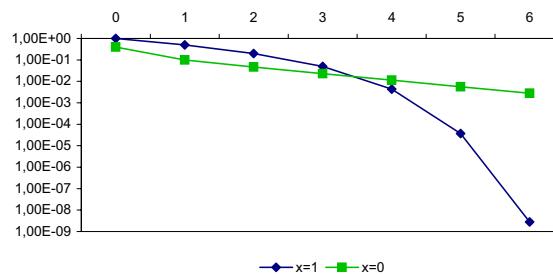
Satz 166. $f : \mathbb{R} \rightarrow \mathbb{R}$ sei zweimal stetig differenzierbar mit $f(x) = 0, f'(x) \neq 0$, d.h. x ist eine **einfache Nullstelle** von f . Liegt x_0 nahe genug an x , dann konvergiert das Newton-Verfahren **quadratisch** gegen x .

- was passiert bei **mehrfachen Nullstellen**, d.h. wenn $f(x) = 0, f'(x) = 0$?
- $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$, weshalb für x_k nahe bei x die Quotientenbildung Probleme macht

Beispiel 167.

- $f(x) = x^2(x - 1)$ hat $x = 1$ als einfache, $x = 0$ als doppelte Nullstelle
- direkte Anwendung liefert

k	xk	Fehler	xk	Fehler
0	2,000000	1,000E+00	0,400000	4,000E-01
1	1,500000	5,000E-01	0,100000	1,000E-01
2	1,200000	2,000E-01	0,047059	4,706E-02
3	1,050000	5,000E-02	0,022934	2,293E-02
4	1,004348	4,348E-03	0,011331	1,133E-02
5	1,000037	3,732E-05	0,005633	5,633E-03
6	1,000000	2,785E-09	0,002808	2,808E-03



- das Verfahren konvergiert auch für die doppelte Nullstelle $x = 0$, aber viel langsamer als für die einfache $x = 1$

- betrachten jetzt den allgemeinen Fall, dass x eine **q -fache Nullstelle**, $q > 1$, von f ist:

$$f(x) = 0, \quad f'(x) = 0, \quad \dots, \quad f^{(q-1)}(x) = 0, \quad f^{(q)}(x) \neq 0$$

- damit kann f geschrieben werden (Taylorentwicklung um x) als

$$f(y) = (y - x)^q g(y), \quad g(x) \neq 0$$

- für die Ableitung von f erhalten wir

$$f'(y) = q(y - x)^{q-1} g(y) + (y - x)^q g'(y)$$

und damit

$$\frac{f(y)}{f'(y)} = \frac{(y - x)^q g(y)}{q(y - x)^{q-1} g(y) + (y - x)^q g'(y)} = \frac{(y - x)g(y)}{qg(y) + (y - x)g'(y)}$$

- da $g(x) \neq 0$ ist folgt

$$\lim_{y \rightarrow x} \frac{f(y)}{f'(y)} = 0,$$

so dass der Quotient im Newton-Verfahren auch im Grenzwert existiert

- berechnet man mit der Darstellung von f auch

$$\Phi'(y) = \left(y - \frac{f(y)}{f'(y)} \right)' = \frac{f(y)f''(y)}{\left(f'(y) \right)^2}$$

so erhält man

$$\Phi'(x) = \lim_{y \rightarrow x} \Phi'(y) = 1 - \frac{1}{q}, \quad q > 1$$

- damit ist $\Phi'(x) > 0$ und wir können Satz 163 nicht anwenden

Satz 168. Für mehrfache Nullstellen ist das Newton-Verfahren wohldefiniert und konvergiert nur linear in einer Umgebung von x

Beweis.

- x sei q -fache Nullstelle von f , $q > 1$

- wegen

$$\Phi'(x) = 1 - \frac{1}{q}$$

und $q > 1$ gilt

$$0 < \Phi'(x) < 1$$

- deshalb gibt es ein Intervall $X = [x - \delta, x + \delta]$ mit $\delta > 0$ und

$$0 < \beta \leq \Phi'(\xi) \leq \alpha < 1 \quad \forall \xi \in X, \tag{7.10}$$

wobei α, β unabhängig von ξ sind

- damit ist $|\Phi'(\xi)| \leq \alpha \forall \xi \in X$

- wie im Beweis von Satz 163 folgt daraus, dass Φ eine Kontraktion auf X ist und die Iteration $x_{k+1} = \Phi(x_k)$ für alle $x_0 \in X$ gegen den Fixpunkt x konvergiert

- nach dem Mittelwertsatz ist

$$|x_{k+1} - x| = |\Phi(x_k) - \Phi(x)| = |\Phi'(\xi)| |x_k - x|, \quad \xi \in \text{co}(x_k, x),$$

so dass wegen (7.10)

$$|x_{k+1} - x| \leq \alpha |x_k - x|$$

bzw.

$$|x_{k+1} - x| \geq \beta |x_k - x|$$

folgt, d.h. es liegt nur lineare Konvergenz vor

□

- wie kann man bei mehrfachen Nullstellen die Konvergenz beschleunigen
- sei x eine q -fache Nullstelle von f , d.h.

$$f(x) = f'(x) = \dots = f^{(q-1)}(x) = 0, \quad f^{(q)}(x) \neq 0$$

- Variante 1:

- setze $\tilde{\Phi}(y) = y - q \frac{f(y)}{f'(y)}$ und iteriere $x_{k+1} = \tilde{\Phi}(x_k)$
- durch Ausrechnen erhält man $\tilde{\Phi}'(x) = 0$ und damit also (mindestens) quadratische Konvergenz
- der Nachteil dieser Variante ist, dass die Vielfachheit q bekannt sein muss

- Variante 2

- da x eine q -fache Nullstelle ist, erhalten wir aus einer Taylorentwicklung von f

$$f(y) = \frac{1}{q!} (y-x)^q f^{(q)}(\xi), \quad f'(y) = \frac{1}{(q-1)!} (y-x)^{q-1} f^{(q)}(\eta)$$

mit $\xi, \eta \in \text{co}(x, y)$

- da $f^{(q)}(x) \neq 0$ ist folgt für y nahe x , dass $f^{(q)}(\xi) \neq 0$ und $f^{(q)}(\eta) \neq 0$ und damit

$$\tilde{f}(y) = \frac{f(y)}{f'(y)} = \frac{1}{q} (y-x) \frac{f^{(q)}(\xi)}{f^{(q)}(\eta)},$$

d.h. x ist einfache Nullstelle von $\tilde{f} = \frac{f}{f'}$

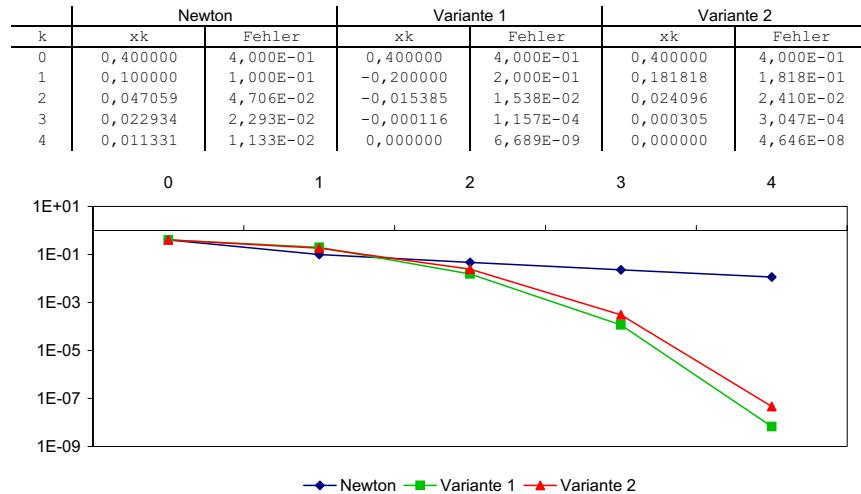
- wird das Newton-Verfahren auf \tilde{f} angewandt, so konvergiert es quadratisch
- die Vielfachheit q wird hier also nicht explizit benutzt
- Nachteil ist die kompliziertere Iterationsvorschrift

$$\Phi(y) = y - \frac{\tilde{f}(y)}{\tilde{f}'(y)} = y - \frac{\frac{f(y)}{f'(y)}}{\frac{(f'(y))^2 - f(y)f''(y)}{(f'(y))^2}} = y - \frac{f(y)f'(y)}{(f'(y))^2 - f(y)f''(y)}$$

in der pro Schritt neben f und f' auch noch f'' auszuwerten ist

Beispiel 169.

- für $f(x) = x^2(x - 1)$ erhalten wir folgende Resultate



- bisher haben wir das Newton-Verfahren nur auf skalare Nullstellenprobleme $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = 0$ angewandt
- was ändert sich für Systeme, d.h. für $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$?
- auch hier leiten wir das Verfahren über die Taylorentwicklung

$$f(y) = f(x_k) + (Df)(x_k)(y - x_k) + R$$

her, wobei

$$(Df)(x_k) = \begin{pmatrix} \partial_1 f_1(x_k) & \dots & \partial_n f_1(x_k) \\ \vdots & & \vdots \\ \partial_1 f_n(x_k) & \dots & \partial_n f_n(x_k) \end{pmatrix}$$

die **Jacobi-Matrix** von f in x_k ist

- durch Weglassen des Restgliedes R erhält man

$$0 = f(x_k) + (Df)(x_k)(x_{k+1} - x_k)$$

und damit für reguläres $Df(x_k)$

$$x_{k+1} = x_k - ((Df)(x_k))^{-1} f(x_k)$$

- analog zum skalaren Fall zeigt man

Satz 170. Sei $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $f(x) = 0$ und $(Df)(x)$ regulär (was einer einfachen Nullstelle im eindimensionalen entspricht), so konvergiert das Newton-Verfahren lokal quadratisch

- das Analogon zur mehrfachen Nullstelle ist der Fall $Df(x)$ singulär
- die Behandlung ist etwas komplizierter als im skalaren Fall

Zusammenfassung.

- das Newton-Verfahren ist nur lokal konvergent, d.h. x_0 muss "nahe" genug bei x liegen
- das Newton-Verfahren konvergiert mit Ordnung 2 für $f'(x) \neq 0$, mit Ordnung 1 bei $f'(x) = 0$
- in der Praxis kombiniert man ein global konvergentes (aber in der Regel langsames) Verfahren um in die Nähe von x zu kommen und schaltet dann auf das schnellere Newton-Verfahren um
- es gibt (insbesondere für Systeme) unzählige Varianten

Kapitel 8

Interpolation

8.1 Einleitung

- gegeben sind Punkte $(x_0, y_0), \dots, (x_n, y_n)$, z.B. Messwerte
- gesucht ist ein funktionaler Zusammenhang zwischen x_i und y_i , d.h. eine Funktion so dass

$$f(x_i) = y_i \quad \forall i = 0, \dots, n,$$

d.h. die Funktion f **interpoliert** die Werte (x_i, y_i)

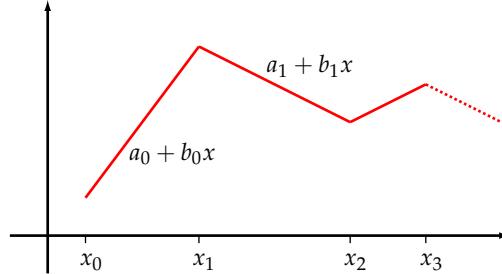
- ist f gefunden, so wird es in der Regel an **Zwischenstellen** $x \neq x_i$ ausgewertet
- f sollte möglichst einfach handhabbar sein, z.B.

- ein Polynom

$$f(x) = a_0 + a_1 x + \dots + a_m x^m$$

- ein stückweises Polynom wie

$$f|_{[x_i, x_{i+1}]}(x) = a_i + b_i x$$



- eine Überlagerung trigonometrischer Funktionen

$$f(x) = c_0 + c_1 \cos(x) + c_2 \cos(2x) + \dots + c_{n-1} \cos((n-1)x)$$

- in allen Fällen sind Koeffizienten so zu bestimmen, dass $f(x_i) = y_i$
- Grundsätzliche Fragen:
 - geht das immer eindeutig
 - wie muss f aussehen (z.B. Polynomgrad, Anzahl der cos-Terme)
 - wie wertet man f effizient an Zwischenstellen aus

8.2 Polynominterpolation

- gegeben seien die $n+1$ Punkte (x_i, y_i) , $i = 0, \dots, n$, d.h. $n+1$ Interpolationsbedingungen sind zu erfüllen
- damit brauchen wir ein Polynom mit mindestens $n+1$ Koeffizienten, also mit Grad n :

$$f(x) = p_n(x) = a_0 + a_1 x + \dots + a_n x^n \quad (8.1)$$

- kann man a_0, \dots, a_n so bestimmen, dass $p_n(x_i) = y_i \forall i$
- aus der Interpolationsbedingung $p_n(x_i) = y_i$ erhalten wir

$$\begin{aligned} y_0 &= p_n(x_0) = a_0 + a_1 x_0 + \dots + a_n x_0^n \\ &\vdots \qquad \vdots \\ y_n &= p_n(x_n) = a_0 + a_1 x_n + \dots + a_n x_n^n \end{aligned}$$

also ein lineares Gleichungssystem für die Koeffizienten a_0, \dots, a_n in der Form

$$\underbrace{\begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \dots & x_n^n \end{pmatrix}}_A \underbrace{\begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix}}_a = \underbrace{\begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}}_y \quad (8.2)$$

- ist $A \in \mathbb{R}^{(n+1) \times (n+1)}$ regulär, so können wir für beliebige y_i -Werte immer ein interpolierendes Polynom vom Grad $\leq n$ finden

Bemerkung 171. A heißt **Vandermonde-Matrix** zu x_i , $i = 0, \dots, n$

Satz 172. Ist $x_i \neq x_j \forall i \neq j$ so gibt es genau ein Polynom $p_n(x)$ mit $\text{Grad} \leq n$, das $(x_0, y_0), \dots, (x_n, y_n)$ interpoliert

Beweis.

- wegen (8.2) interpoliert $p_n(x) = a_0 + a_1 x + \dots + a_n x^n$ die Punkte (x_i, y_i) genau dann, wenn $a = (a_0, \dots, a_n)^T$ Lösung von $Aa = y$ ist
- $Aa = y$ ist eindeutig lösbar (d.h. es existiert genau ein interpolierendes Polynom) genau dann wenn A regulär ist, was äquivalent zu $\det(A) \neq 0$ ist
- deswegen berechnen wir jetzt $\det(A)$:

$$\begin{aligned} \det(A) &= \det \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \\ &= \det \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 - x_0 & x_1^2 - x_0 x_1 & \dots & x_1^n - x_0 x_1^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n - x_0 & x_n^2 - x_0 x_n & \dots & x_n^n - x_0 x_n^{n-1} \end{pmatrix} \end{aligned}$$

$$\begin{aligned}
&= \det \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 - x_0 & x_1(x_1 - x_0) & \dots & x_1^{n-1}(x_1 - x_0) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n - x_0 & x_n(x_n - x_0) & \dots & x_n^{n-1}(x_n - x_0) \end{pmatrix} \\
&= (x_1 - x_0) \cdot (x_2 - x_0) \cdot \dots \cdot (x_n - x_0) \cdot \det \begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{pmatrix}
\end{aligned}$$

◦ per Induktion folgt

$$\begin{aligned}
\det(A) &= (x_1 - x_0) \cdot (x_2 - x_0) \cdot \dots \cdot (x_n - x_0) \\
&\quad \cdot (x_2 - x_1) \cdot \dots \cdot (x_n - x_1) \\
&\quad \quad \quad \ddots \quad \vdots \\
&\quad \cdot (x_n - x_{n-1}) \\
&= \prod_{0 \leq i < j \leq n} (x_j - x_i)
\end{aligned}$$

◦ damit ist $\det(A) \neq 0$ genau dann wenn $x_i \neq x_j \forall i \neq j$

□

- praktische Durchführung:

- stelle das lineare Gleichungssystem $Aa = y$ auf und löse es
- damit sind die Polynomkoeffizienten a_0, \dots, a_n bekannt und $p_n(x) = a_0 + a_1x + \dots + a_nx^n$ erfüllt $p_n(x_i) = y_i, i = 0, \dots, n$
- die Auswertung an Zwischenstellen erfolgt
 - direkt über $p_n(x) = a_0 + a_1x + \dots + a_nx^n$, d.h. berechne $x^i, a_i x^i$ und summiere, was teuer und rundungsfehleranfällig ist
 - mit dem **Horner-Schema**:

- $p_n(x)$ wird wie folgt umgeformt

$$\begin{aligned}
p_n(x) &= a_0 + a_1x + \dots + a_nx^n \\
&= a_0 + x(a_1 + a_2x + \dots + a_nx^{n-1}) \\
&= a_0 + x(a_1 + x(a_2 + \dots + a_nx^{n-2})) \\
&\quad \vdots \\
&= a_0 + x(a_1 + x(a_2 + x(\dots + x(a_{n-1} + xa_n)\dots)))
\end{aligned}$$

- berechne schrittweise von innen nach außen

$$\begin{aligned}
q_0 &= a_n \\
q_1 &= a_{n-1} + x \cdot q_0 = a_{n-1} + xa_n \\
q_2 &= a_{n-2} + x \cdot q_1 = a_{n-2} + x(a_{n-1} + xa_n) = a_{n-2} + xa_{n-1} + x^2a_n
\end{aligned}$$

also

$$q_0 = a_n, \quad q_k = a_{n-k} + xq_{k-1}, \quad k = 1, \dots, n$$

und damit $p_n(x) = q_n$

- bestimmt man das Interpolationspolynom wie oben beschrieben so ist

- der Aufwand zum Erstellen und Lösen von $Aa = y$ relativ hoch
- die Kondition von A oft schlecht
- geht das auch einfacher?
- wir betrachten x_0, \dots, x_n und definieren das j -te **Lagrange-Polynom** als

$$L_j(x) = \frac{x - x_0}{x_j - x_0} \cdot \dots \cdot \frac{x - x_{j-1}}{x_j - x_{j-1}} \cdot \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdot \dots \cdot \frac{x - x_n}{x_j - x_n}$$

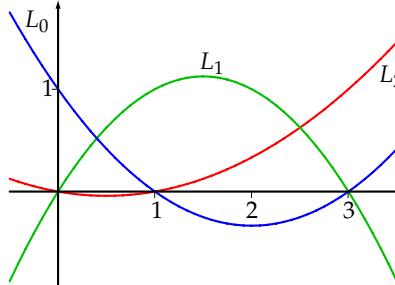
- L_j ist wohldefiniert falls $x_i \neq x_j \forall i \neq j$ und hat Grad n
- außerdem gilt

$$L_j(x_i) = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{sonst} \end{cases} \quad (8.3)$$

Beispiel 173.

- gegeben ist $x_0 = 0, x_1 = 1, x_2 = 3$
- damit erhalten wir

$$\begin{aligned} L_0(x) &= \frac{x - x_1}{x_0 - x_1} \frac{x - x_2}{x_0 - x_2} = \frac{x - 1}{0 - 1} \frac{x - 3}{0 - 3} = \frac{1}{3}(x - 1)(x - 3) \\ L_1(x) &= \frac{x - x_0}{x_1 - x_0} \frac{x - x_2}{x_1 - x_2} = \frac{x - 0}{1 - 0} \frac{x - 3}{1 - 3} = -\frac{1}{2}x(x - 3) \\ L_2(x) &= \frac{x - x_0}{x_2 - x_0} \frac{x - x_1}{x_2 - x_1} = \frac{x - 0}{3 - 0} \frac{x - 1}{3 - 1} = \frac{1}{6}x(x - 1) \end{aligned}$$



- mit den Lagrange-Polynomen L_j können wir das Interpolationspolynom nun ganz einfach ermitteln
- setzt man

$$p_n(x) = \sum_{j=0}^n y_j L_j(x)$$

dann ist der Grad von $p_n(x)$ höchstens n und wegen (8.3) gilt

$$p_n(x_i) = \sum_{j=0}^n y_j L_j(x_i) = \sum_{j=0}^n y_j \delta_{ij} = y_i \quad \forall i = 0, \dots, n,$$

d.h. $p_n(x)$ ist das gesuchte Interpolationspolynom

- damit erhalten wir folgendes Verfahren:

- berechne aus den x_i die Lagrange Polynome L_0, \dots, L_n
- berechne mit den y_i dann

$$p_n(x) = \sum_{j=0}^n y_j L_j(x) \quad (8.4)$$

Beispiel 174.

- gegeben seien die Punkte $(0, 3), (1, 2), (3, 6)$, d.h. $n = 2$,

$$\begin{aligned} x_0 &= 0, & x_1 &= 1, & x_2 &= 3 \\ y_0 &= 3, & y_1 &= 2, & y_2 &= 6 \end{aligned}$$

- die Lagrange-Polynome

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

wurden bereits in Beispiel 173 berechnet

- als Interpolationspolynom erhalten wir schließlich

$$\begin{aligned} p_2(x) &= 3L_0(x) + 2L_1(x) + 6L_2(x) \\ &= (x-1)(x-3) - x(x-3) + x(x-1) \\ &= x^2 - 2x + 3 \end{aligned}$$

- bei Benutzung der Lagrange-Polynome müssen wir kein Gleichungssystem lösen
- kommt ein weiterer Datenpunkt (x_{n+1}, y_{n+1}) hinzu, müssen sämtliche L_j verändert werden
- gibt es eine Möglichkeit, p_n schrittweise aufzubauen, d.h.
 - mit dem ersten Punkt (x_0, y_0) wird zunächst $p_0(x)$ mit Grad 0 erzeugt
 - mit dem zweiten Punkt (x_1, y_1) wird p_0 auf $p_1(x)$ mit Grad 1 erweitert
 - mit dem dritten Punkt (x_2, y_2) wird p_1 auf $p_2(x)$ mit Grad 2 erweitert
 - etc.
- dazu untersuchen wir zunächst, wie sich $p_n(x)$ und $p_{n+1}(x)$ unterscheiden
 - die Interpolationseigenschaften von p_n und p_{n+1} bedeuten

$$\begin{aligned} p_n(x_i) &= y_i, & i &= 0, \dots, n, \\ p_{n+1}(x_i) &= y_i, & i &= 0, \dots, n, n+1 \end{aligned}$$

- das Polynom $q_{n+1}(x) = p_{n+1}(x) - p_n(x)$ hat Grad $n+1$ und

$$\begin{aligned} q_{n+1}(x_i) &= 0 & i &= 0, \dots, n \\ q_{n+1}(x_{n+1}) &= y_{n+1} - p_n(x_{n+1}) =: \hat{a}_{n+1}, \end{aligned}$$

d.h. q_{n+1} muss ein Vielfaches des $(n+1)$ -ten Lagrange-Polynoms L_{n+1} zu x_0, \dots, x_{n+1} sein:

$$q_{n+1}(x) = \hat{a}_{n+1} L_{n+1}(x)$$

- damit ist

$$p_{n+1}(x) = p_n(x) + q_{n+1}(x) = p_n(x) + \hat{a}_{n+1} L_{n+1}(x)$$

mit $\hat{a}_{n+1} = y_{n+1} - p_n(x_{n+1})$

- wegen

$$L_{n+1}(x) = \frac{(x - x_0) \cdot \dots \cdot (x - x_n)}{(x_{n+1} - x_0) \cdot \dots \cdot (x_{n+1} - x_n)}$$

folgt

$$p_{n+1}(x) = p_n(x) + a_{n+1}(x - x_0) \cdot \dots \cdot (x - x_n),$$

$$a_{n+1} = \frac{\hat{a}_{n+1}}{(x_{n+1} - x_0) \cdot \dots \cdot (x_{n+1} - x_n)}$$

$$= \frac{y_{n+1} - p_n(x_{n+1})}{(x_{n+1} - x_0) \cdot \dots \cdot (x_{n+1} - x_n)}$$

- damit können wir Punkt für Punkt das Interpolationspolynom aufbauen:

$$p_0(x) = a_0 = y_0,$$

$$a_0 = y_0$$

$$p_1(x) = p_0(x) + a_1(x - x_0)$$

$$a_1 = \frac{y_1 - p_0(x_1)}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$p_2(x) = p_1(x) + a_2(x - x_0)(x - x_1)$$

$$= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1),$$

$$a_2 = \frac{y_2 - p_1(x_2)}{(x_2 - x_0)(x_2 - x_1)}$$

⋮

$$p_n(x) = a_0 + a_1(x - x_0) + \dots + a_n(x - x_0) \cdot \dots \cdot (x - x_{n-1})$$

Bemerkung 175. Diese Darstellung von p_n heißt **Newton-Interpolationspolynom**

- die a_n können mit Hilfe **dividierter Differenzen** sehr einfach berechnet werden

$$d_{ii} = y_i \quad i = 0, 1, \dots \quad d_{nm} = \frac{d_{n,m+1} - d_{n-1,m}}{x_n - x_m} \quad n > m \quad (8.5)$$

Beispiel 176.

$$d_{00} = y_0$$

$$d_{11} = y_1 \quad d_{10} = \frac{d_{11} - d_{00}}{x_1 - x_0}$$

$$= \frac{y_1 - y_0}{x_1 - x_0}$$

$$d_{22} = y_2 \quad d_{21} = \frac{d_{22} - d_{11}}{x_2 - x_1}$$

$$= \frac{y_2 - y_1}{x_2 - x_1}$$

$$d_{20} = \frac{d_{21} - d_{10}}{x_2 - x_0}$$

$$= \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0}$$

- bei gegebenen Punkten (x_i, y_i) kann damit punktweise (Zeile für Zeile) folgendes Schema aufgebaut werden

$$\begin{array}{rcl}
 y_0 & = & d_{00} \\
 & & \searrow \\
 y_1 & = & d_{11} \rightarrow d_{10} \\
 & & \searrow \quad \searrow \\
 y_2 & = & d_{22} \rightarrow d_{21} \rightarrow d_{20} \\
 & \vdots & \vdots \quad \vdots \quad \vdots \quad \ddots
 \end{array}$$

- wird ein Punkt hinzugefügt, wird das Schema lediglich um eine neue Zeile erweitert
- durch Induktion zeigt man für den Koeffizienten a_n des Newton-Interpolationspolynoms

$$a_n = d_{n0}, \quad \forall n$$

- damit erhalten wir insgesamt das folgende Ergebnis

Satz 177. Das Interpolationspolynom in Newton-Darstellung lautet

$$p_n(x) = d_{00} + d_{10}(x - x_0) + \dots + d_{n0}(x - x_0) \cdot \dots \cdot (x - x_{n-1}), \quad (8.6)$$

wobei d_{k0} dividierte Differenzen sind:

$$d_{ii} = y_i, \quad d_{nm} = \frac{d_{n\,m+1} - d_{n-1\,m}}{x_n - x_m}.$$

- zur Auswertung schreiben wir $p_n(x)$ als

$$p_n(x) = d_{00} + (x - x_0)(d_{10} + (x - x_1)(d_{20} + \dots (x - x_{n-1})d_{n0}) \dots)$$

und erhalten analog zum Horner-Schema

$$q_0 = d_{n0}, \quad q_k = d_{n-k\,0} + (x - x_{n-k})q_{k-1} \quad k = 1, \dots, n$$

und $p_n(x) = q_n$

Beispiel 178.

- wir betrachten die drei Punkte $(0, 3), (1, 2), (3, 6)$ und wollen das zugehörige Interpolationspolynom an der Stelle $x = 2$ auswerten
- mit $x_0 = 0, x_1 = 1, x_2 = 3$ erhalten wir als dividierte Differenzen

$$\begin{array}{cc}
 x_i & y_i \\
 0 & 3 \\
 & \searrow \\
 1 & 2 \rightarrow \frac{2-3}{1-0} = -1 \\
 & \searrow \quad \searrow \\
 3 & 6 \rightarrow \frac{6-2}{3-1} = 2 \rightarrow \frac{2-(-1)}{3-0} = 1
 \end{array}$$

und damit die Interpolationspolynome

$$\begin{aligned} p_0(x) &= 3 \\ p_1(x) &= p_0(x) + (-1)(x - x_0) \\ &= 3 - x \\ p_2(x) &= p_1(x) + (x - x_0)(x - x_1) \\ &= 3 - x + x(x - 1) \end{aligned}$$

- die Auswertung bei $x = 2$ liefert

$$\begin{aligned} q_0 &= d_{20} = 1 \\ q_1 &= d_{10} + (x - x_1)q_0 = -1 + (2 - 1)1 = 0 \\ p_2(2) &= q_2 = d_{00} + (x - x_0)q_1 = 3 \end{aligned}$$

8.3 Splines

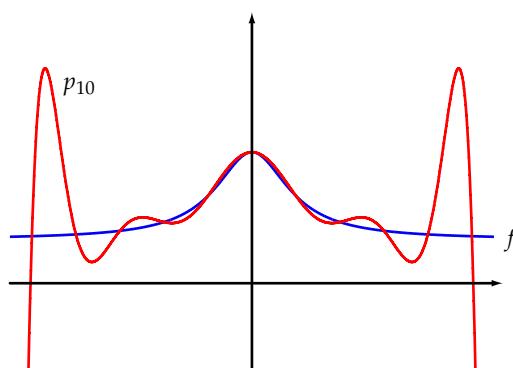
- Probleme bei Interpolationspolynom:
 - bei steigender Zahl von Punkten wächst auch der Polynomgrad
 - bei höherem Polynomgrad kann es zu sehr eigenartigem Verhalten des Interpolationspolynoms an Zwischenstellen kommen (Überschwinger, Instabilitäten)

Beispiel 179.

- betrachte $x_0 = -5, x_1 = -4, \dots, x_{10} = 5$ und $y_i = f(x_i)$ mit

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

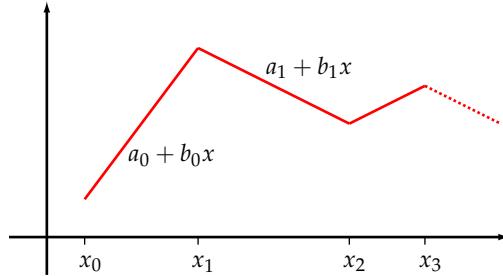
- berechnet man das Interpolationspolynom vom Grad 10 durch $(x_i, y_i), i = 0, \dots, 10$ und vergleicht es mit f so erhalten wir



- in konkreten Anwendungen können tausende von Punkten auftreten, so dass die Polynominterpolation unbrauchbar ist
- Lösung:

- begrenze Polynomgrad (≤ 3)
- stückle mehrere Polynome aneinander
- fordere glatte Übergänge zwischen einzelnen Polynomstücken

Beispiel 180. Bei Polynomgrad ≤ 1 auf $[x_i, x_{i+1}]$ und stetigen Übergängen erhalten wir einen interpolierenden **Polygonzug**



Definition 181. Seien $x_0 < x_1 < \dots < x_n$ gegeben. Eine Funktion s_k mit

- $s_k|_{[x_i, x_{i+1}]}$ ist Polynom vom Grad $\leq k$
- s_k ist C^{k-1} auf $[x_0, x_n]$

heißt **Polynomspline** der Ordnung k zu x_i , $i = 0, \dots, n$.

Beispiel 182.

- ein Polygonzug ist ein Polynomspline der Ordnung 1
- in der Praxis werden kubische Polynomsplines ($k = 3$) am häufigsten benutzt:
 - $s_3|_{[x_i, x_{i+1}]} = a_i + b_i x + c_i x^2 + d_i x^3$
 - s_3 ist zweimal stetig differenzierbar auf $[x_0, x_n]$, d.h. insbesondere an den Nahtstellen x_i , $i = 1, \dots, n - 1$

- wir betrachten jetzt nur noch kubische Splines und versuchen die a_i, b_i, c_i, d_i so zu bestimmen, dass s_3 die Punkte (x_i, y_i) interpoliert, d.h. $s_3(x_i) = y_i$, $i = 0, \dots, n$
- insgesamt haben wir n Intervalle $[x_i, x_{i+1}]$, $i = 0, \dots, n - 1$, so dass $4n$ Koeffizienten a_i, b_i, c_i, d_i zu bestimmen sind
- als Bedingungsgleichungen erhalten wir:
 - Interpolation:

$$s_3(x_i) = y_i, \quad i = 0, \dots, n \quad \Rightarrow \quad n + 1 \text{ Stück}$$

- die Glattheitsbedingungen an den inneren Punkten x_i , $i = 1, \dots, n - 1$ liefern

$$\begin{aligned} C^0 : \quad s_3(x_i)_- &= s_3(x_i)_+ \quad \Rightarrow \quad n - 1 \text{ Stück} \\ C^1 : \quad s'_3(x_i)_- &= s'_3(x_i)_+ \quad \Rightarrow \quad n - 1 \text{ Stück} \\ C^2 : \quad s''_3(x_i)_- &= s''_3(x_i)_+ \quad \Rightarrow \quad n - 1 \text{ Stück} \end{aligned}$$

- damit haben wir $4n - 2$ Gleichungen für $4n$ Unbekannte und brauchen zwei zusätzliche **Abschlussbedingungen**
- die gängigsten Abschlussbedingungen sind:

- **natürlicher Spline:**

$$s_3''(x_0) = s_3''(x_n) = 0,$$

d.h. die Krümmung am Rand verschwindet

- **periodischer Spline:**

$$s_3'(x_0) = s_3'(x_n), \quad s_3''(x_0) = s_3''(x_n)$$

- **Hermite Spline:**

$$s_3'(x_0) = u, \quad s_3'(x_n) = v,$$

u, v gegeben

- wir untersuchen den natürlichen Spline genauer, die anderen Fälle lassen sich analog bearbeiten
- wir schreiben zunächst die Bedingungen in Gleichungsform
 - auf $[x_i, x_{i+1}], i = 0, \dots, n-1$ gilt

$$\begin{aligned} s_3(x) &= a_i + b_i x + c_i x^2 + d_i x^3 \\ s_3'(x) &= b_i + 2c_i x + 3d_i x^2 \\ s_3''(x) &= 2c_i + 6d_i x \end{aligned}$$

- Interpolationseigenschaft und Stetigkeit sind erfüllt, falls

$$\begin{aligned} a_i + b_i x_i + c_i x_i^2 + d_i x_i^3 &= y_i & i = 0, \dots, n-1 \\ a_i + b_i x_{i+1} + c_i x_{i+1}^2 + d_i x_{i+1}^3 &= y_{i+1} \end{aligned}$$

- Stetigkeit der ersten Ableitung liefert

$$b_{i-1} + 2c_{i-1} x_i + 3d_{i-1} x_i^2 = b_i + 2c_i x_i + 3d_i x_i^2, \quad i = 1, \dots, n-1$$

und Stetigkeit der zweiten Ableitung bedeutet

$$2c_{i-1} + 6d_{i-1} x_i = 2c_i + 6d_i x_i, \quad i = 1, \dots, n-1$$

- schließlich erhalten wir aus der natürlichen Abschlussbedingung

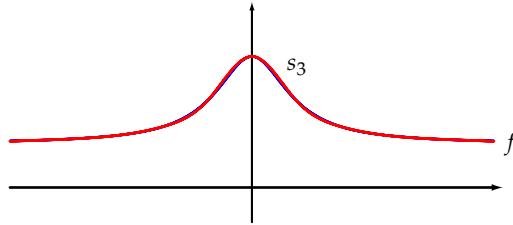
$$\begin{aligned} 2c_0 + 6d_0 x_0 &= 0 \\ 2c_{n-1} + 6d_{n-1} x_n &= 0 \end{aligned}$$

- alle Gleichungen sind linear in a_i, b_i, c_i, d_i , so dass wir insgesamt ein lineares Gleichungssystem der Dimension $4n$ erhalten
- für $x_i \neq x_j \forall i \neq j$ ist dies stets eindeutig lösbar (auch bei periodischen bzw. Hermite Abschlussbedingungen)

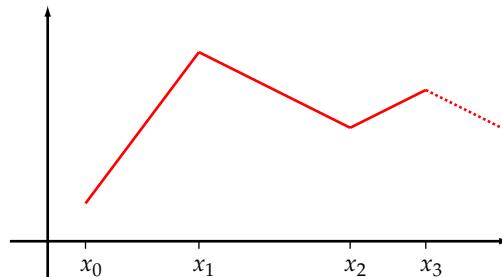
Satz 183. $(x_i, y_i), i = 0, \dots, n$ gegeben, $x_0 < x_1 < \dots < x_n$. Dann gibt es genau einen kubischen Spline s_3 mit natürlicher (periodischer/Hermite) Abschlussbedingung.

Beispiel 184.

- Die Punkte (x_i, y_i) seien wie in Beispiel 179 gegeben
- ein Vergleich der Graphen zeigt, dass beim natürlichen kubischen Spline keine Stabilitätsprobleme auftreten



- kann man s_3 effizienter berechnen?
 - die Dimension des linearen Gleichungssystems ist $4n$
 - die Systemmatrix hat wenig Struktur
- Idee:
 - s_3 ist C^2 und ist stückweise polynomial mit Grad ≤ 3
 - s_3'' ist damit stetig und stückweise linear mit $s_3''(x_0) = s_3''(x_n) = 0$ bei natürlichen Randbedingungen, also ein Polygonzug



- ein Polygonzug ist bestimmt durch die Funktionswerte an den Stützstellen, d.h. durch

$$\beta_i = s_3''(x_i), \quad i = 0, \dots, n$$
- wir versuchen also zunächst die **Momente** β_i (und damit den Polygonzug s_3'') zu bestimmen und integrieren dann zweimal um schließlich s_3 zu erhalten
- wir betrachten s_3 auf $[x_i, x_{i+1}], i \in \{0, \dots, n-1\}$ fest
 - s_3 ist ein Polynom mit Grad ≤ 3

$$s_3(x_i) = y_i, \quad s_3(x_{i+1}) = y_{i+1}$$

- s_3' ist ein Polynom mit Grad ≤ 2

$$s_3'(x_i) =: \alpha_i, \quad s_3'(x_{i+1}) =: \alpha_{i+1} \quad (8.7)$$

- s_3'' ist ein Polynom vom Grad ≤ 1 (also eine Gerade), mit

$$s_3''(x_i) = \beta_i, \quad s_3''(x_{i+1}) = \beta_{i+1}$$

so dass

$$s_3''(x) = \beta_i + \frac{x - x_i}{x_{i+1} - x_i}(\beta_{i+1} - \beta_i) = \beta_i + \frac{\beta_{i+1} - \beta_i}{h_i}(x - x_i)$$

mit $h_i = x_{i+1} - x_i$

- Integration von $s_3''(z)$ über $[x_i, x]$ liefert

$$\begin{aligned} \int_{x_i}^x s_3''(z) dz &= \int_{x_i}^x \beta_i + \frac{\beta_{i+1} - \beta_i}{h_i}(z - x_i) dz \\ \Leftrightarrow s_3'(x) - s_3'(x_i) &= \beta_i(x - x_i) + \frac{\beta_{i+1} - \beta_i}{2h_i}(x - x_i)^2 \end{aligned}$$

und wegen (8.7)

$$s_3'(x) = \alpha_i + \beta_i(x - x_i) + \frac{\beta_{i+1} - \beta_i}{2h_i}(x - x_i)^2 \quad (8.8)$$

- integrieren wir $s_3'(z)$ über $[x_i, x]$ so folgt

$$s_3(x) = y_i + \alpha_i(x - x_i) + \frac{\beta_i}{2}(x - x_i)^2 + \frac{\beta_{i+1} - \beta_i}{6h_i}(x - x_i)^3 \quad (8.9)$$

- mit $x = x_{i+1}$ erhalten wir aus (8.8)

$$\begin{aligned} \alpha_{i+1} &= s_3'(x_{i+1}) \\ &= \alpha_i + \beta_i(x_{i+1} - x_i) + \frac{\beta_{i+1} - \beta_i}{2h_i}(x_{i+1} - x_i)^2 \\ &= \alpha_i + \beta_i h_i + \frac{\beta_{i+1} - \beta_i}{2h_i} h_i^2 \end{aligned}$$

und damit

$$\alpha_{i+1} - \alpha_i = \frac{\beta_i + \beta_{i+1}}{2} h_i \quad (8.10)$$

- mit $x = x_{i+1}$ liefert (8.9)

$$\begin{aligned} y_{i+1} &= s_3(x_{i+1}) \\ &= y_i + \alpha_i(x_{i+1} - x_i) + \frac{\beta_i}{2}(x_{i+1} - x_i)^2 + \frac{\beta_{i+1} - \beta_i}{6h_i}(x_{i+1} - x_i)^3 \\ &= y_i + \alpha_i h_i + \frac{\beta_i}{2} h_i^2 + \frac{\beta_{i+1} - \beta_i}{6} h_i^2 \end{aligned}$$

so dass

$$\frac{y_{i+1} - y_i}{h_i} = \alpha_i + \frac{\beta_i}{3} h_i + \frac{\beta_{i+1}}{6} h_i \quad (8.11)$$

- betrachten wir jetzt einen inneren Punkt x_i , $i \in \{1, \dots, n-1\}$ und das Intervall $[x_i, x_{i+1}]$ bzw. $[x_{i-1}, x_i]$, so folgt aus (8.11)

$$\begin{aligned} \frac{y_{i+1} - y_i}{h_i} &= \alpha_i + \frac{\beta_i}{3} h_i + \frac{\beta_{i+1}}{6} h_i \\ \frac{y_i - y_{i-1}}{h_{i-1}} &= \alpha_{i-1} + \frac{\beta_{i-1}}{3} h_{i-1} + \frac{\beta_i}{6} h_{i-1} \end{aligned}$$

und damit

$$\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} = \alpha_i - \alpha_{i-1} + \frac{\beta_i}{3} h_i + \frac{\beta_i + 1}{6} h_i - \frac{\beta_{i-1}}{3} h_{i-1} - \frac{\beta_i}{6} h_{i-1}$$

$$\begin{aligned}
&\stackrel{(8.10)}{=} \frac{\beta_{i-1} + \beta_i}{2} h_{i-1} + \frac{\beta_i}{3} h_i + \frac{\beta_{i+1}}{6} h_i - \frac{\beta_{i-1}}{3} h_{i-1} - \frac{\beta_i}{6} h_{i-1} \\
&= \frac{h_{i-1}}{6} \beta_{i-1} + \frac{h_{i-1} + h_i}{3} \beta_i + \frac{h_i}{6} \beta_{i+1}
\end{aligned}$$

bzw.

$$h_{i-1} \beta_{i-1} + 2(h_{i-1} + h_i) \beta_i + h_i \beta_{i+1} = 6 \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right) \quad (8.12)$$

für $i = 1, \dots, n-1$

- Gleichung (8.12) liefert also $n-1$ lineare Gleichungen für die $n+1$ Unbekannten $\beta_0, \beta_1, \dots, \beta_{n-1}, \beta_n$
- aus den natürlichen Abschlussbedingung wissen wir aber, dass $\beta_0 = \beta_n = 0$
- zusammen bleibt also ein lineares Gleichungssystem der Dimension $n-1$ übrig

$$\underbrace{\begin{pmatrix} 2(h_0 + h_1) & h_1 \\ h_1 & 2(h_1 + h_2) & h_2 \\ & \ddots & \ddots & \ddots \\ & \ddots & \ddots & h_{n-2} \\ h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} \beta_1 \\ \vdots \\ \beta_{n-1} \end{pmatrix}}_{\beta} = \underbrace{\begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_{n-1} \end{pmatrix}}_{\gamma}$$

mit

$$\gamma_i = 6 \left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right), \quad h_i = x_{i+1} - x_i$$

- A ist streng diagonaldominant und damit regulär, so dass die β_i eindeutig bestimmt sind
- damit kennen wir die Momente β_i , d.h. s''_3
- wie erhält man s_3 ?
- nach (8.9) gilt:

$$s_3|_{[x_i, x_{i+1}]}(x) = y_i + \alpha_i(x - x_i) + \frac{\beta_i}{2}(x - x_i)^2 + \frac{\beta_{i+1} - \beta_i}{6h_i}(x - x_i)^3$$

- bis auf die α_i sind alle Koeffizienten bekannt
- die α_i können mit Hilfe von (8.11) berechnet werden:

$$\alpha_i = \frac{y_{i+1} - y_i}{h_i} - \frac{1}{3}\beta_i h_i - \frac{1}{6}\beta_{i+1} h_i$$

- zusammengefasst erhalten wir folgendes Verfahren zur Bestimmung eines natürlichen, kubischen Splines s_3 :

- berechne aus x_i die $h_i = x_{i+1} - x_i$ und stelle

$$A = \begin{pmatrix} 2(h_0 + h_1) & h_1 \\ h_1 & 2(h_1 + h_2) & h_2 \\ & \ddots & \ddots & \ddots \\ & \ddots & \ddots & h_{n-2} \\ h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{pmatrix} \in \mathbb{R}^{(n-1) \times (n-1)}$$

auf

- A ist symmetrisch, tridiagonal und streng diagonaldominant, also sehr einfach mit direkten oder iterativen Lösern zu bearbeiten
- berechne aus y_i, h_i die rechte Seite $\gamma = (\gamma_1, \dots, \gamma_{n-1})^T$,

$$\gamma_i = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$

- löse $A\beta = \gamma$ und setze $\beta_0 = \beta_n = 0$

- berechne α_i durch

$$\alpha_i = \frac{y_{i+1} - y_i}{h_i} - \frac{1}{3}\beta_i h_i - \frac{1}{6}\beta_{i+1} h_i, \quad i = 0, \dots, n-1$$

- auf $[x_i, x_{i+1}]$ ist $s_3(x)$ dann gegeben durch

$$y_i + \alpha_i(x - x_i) + \frac{\beta_i}{2}(x - x_i)^2 + \frac{\beta_{i+1} - \beta_i}{6h_i}(x - x_i)^3$$

- Auswertung erfolgt dann z.B. mit dem Horner-Schema (analog Newton-Polynom)

Beispiel 185.

► betrachte $f(x) = \frac{1}{2} + \frac{1}{1+x^2}$, $x_i = -3, -1, 0, 1, 3$, $y_i = f(x_i)$

► daraus folgt

i	0	1	2	3	4
x_i	-3	-1	0	1	3
y_i	$\frac{3}{5}$	1	$\frac{3}{2}$	1	$\frac{3}{5}$
h_i	2	1	1	2	-

und

$$A = \begin{pmatrix} 2(2+1) & 1 & 0 \\ 1 & 2(1+1) & 1 \\ 0 & 1 & 2(1+2) \end{pmatrix} = \begin{pmatrix} 6 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 6 \end{pmatrix},$$

$$\gamma = \begin{pmatrix} 6\left(\frac{\frac{3}{2}-1}{1} - \frac{1-\frac{3}{5}}{2}\right) \\ 6\left(\frac{1-\frac{3}{2}}{1} - \frac{\frac{3}{2}-1}{1}\right) \\ 6\left(\frac{\frac{3}{5}-1}{2} - \frac{1-\frac{3}{2}}{1}\right) \end{pmatrix} = \begin{pmatrix} \frac{9}{5} \\ -6 \\ \frac{9}{5} \end{pmatrix}$$

► löst man $A\beta = \gamma$ und benutzt $\beta_0 = 0 = \beta_n$, so erhält man

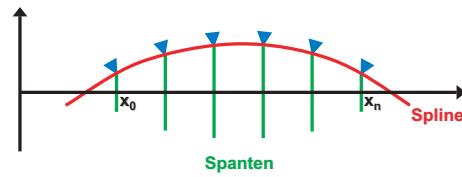
i	0	1	2	3	4
β_i	0	$\frac{3}{5}$	$-\frac{9}{5}$	$\frac{3}{5}$	0
α_i	0	$\frac{3}{5}$	0	$-\frac{3}{5}$	-

und als Spline

$$s_3(x) = \begin{cases} \frac{39}{20} + \frac{27}{20}x + \frac{9}{20}x^2 + \frac{1}{20}x^3 & \text{für } x < -1 \\ \frac{3}{2} - \frac{9}{10}x^2 - \frac{2}{5}x^3 & \text{für } -1 \leq x < 0 \\ \frac{3}{2} - \frac{9}{10}x^2 + \frac{2}{5}x^3 & \text{für } 0 \leq x < 1 \\ \frac{39}{20} - \frac{27}{20}x + \frac{9}{20}x^2 - \frac{1}{20}x^3 & \text{sonst} \end{cases}$$

Bemerkung 186.

- ◊ "Spline" ist der englische Begriff für ein stabförmiges Kurvenlineal aus Holz oder Metall
- ◊ es wurde im Schiffsbau benutzt um die Form der Längsbepflankung des Bootsrumpfs zu bestimmen



- ◊ die Form des gebogenen Lineals $s(x)$ ergibt sich aus den Randbedingungen und der Minimierung der Biegeenergie

$$E = \int \left(\frac{s''(x)}{(1 + (s'(x))^2)^{\frac{3}{2}}} \right)^2 dx$$

- ◊ ein kubischer Spline s ist unter allen interpolierenden C^2 -Funktionen diejenige, die die linearisierte Biegeenergie

$$\tilde{E} = \int (s''(x))^2 dx$$

minimiert

8.4 B-Splines

- die im letzten Kapitel hergeleitete Methode zur Bestimmung kubischer Splines ist sehr effizient, lässt sich aber nur schwer verallgemeinern
- zur Implementierung benutzt man daher in der Regel einen anderen Zugang über sogenannte **B-Splines** (Basis-Splines oder auch Bell-Splines)
- zur Motivation der folgenden Überlegungen betrachten wir nochmal die Polynominterpolation:

- durch die Punkte $(x_i, y_i), i = 0, \dots, n, x_0 < x_1 < \dots < x_n$ können wir genau ein Polynom

$$p_n(x) = a_0 + a_1 x + \dots + a_n x^n$$

vom Grad n legen

- das Polynom p_n ist durch seine Koeffizienten a_0, \dots, a_n bestimmt, d.h. durch einen Vektor

$$(a_0, \dots, a_n)^T \in \mathbb{R}^{n+1}$$

- die Menge der Polynome vom Grad n ist also ein $(n+1)$ -dimensionaler Vektorraum
- damit kann jedes Polynom p_n vom Grad n geschrieben werden als

$$p_n(x) = c_0 e_0(x) + \dots + c_n e_n(x)$$

wobei $e_0(x), \dots, e_n(x)$ linear unabhängige Basis-Polynome sind und $c_0, \dots, c_n \in \mathbb{R}$ Koeffizienten

- durch Benutzung verschiedener Basen erhalten wir die unterschiedlichen Darstellungen des Interpolationspolynoms aus Abschnitt 8.2:

- beim direkten Zugang (8.1) werden Monome als Basis verwendet, d.h.

$$e_0(x) = x^0 = 1, \quad e_1(x) = x^1, \quad \dots, \quad e_n(x) = x^n$$

und $c_i = a_i, i = 0, \dots, n$

- beim Lagrange-Verfahren (8.4) ist

$$e_i(x) = L_i(x), \quad i = 0, \dots, n$$

und $c_i = y_i, i = 0, \dots, n$

- in der Newton-Darstellung (8.6) ist

$$e_0(x) = 1, \quad e_1(x) = x - x_0, \quad \dots, \quad e_n(x) = (x - x_0) \cdot \dots \cdot (x - x_{n-1})$$

und $c_i = d_{i0}, i = 0, \dots, n$, wobei d_{i0} die entsprechenden dividierten Differenzen (8.5) sind

- je nach Basis erhält man sehr unterschiedliche Algorithmen für dieselbe Aufgabenstellung
- die Überlegungen bezüglich Basen werden wir jetzt auf Splines übertragen
- dazu untersuchen wir allgemeine Polynom-Splines wie in Definition 181 und ermitteln zunächst, wie viele linear unabhängige Spline-Funktionen existieren

Definition 187.

- ist $x_0 < x_1 < \dots < x_n$ so heißt

$$\Omega_n = \{x_0, \dots, x_n\}$$

Unterteilung des Intervalls $[x_0, x_n]$

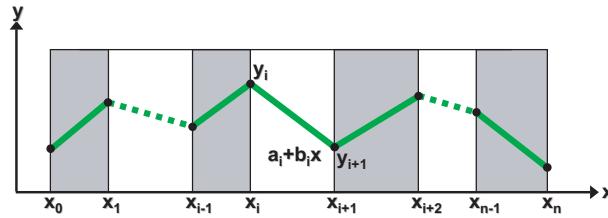
- $S_k(\Omega_n)$ ist die Menge aller Polynom-Splines der Ordnung k zur Unterteilung $\Omega_n = \{x_0, \dots, x_n\}$

Satz 188. In $S_k(\Omega_n)$ gibt es genau $k + n$ linear unabhängige Splines, d.h.

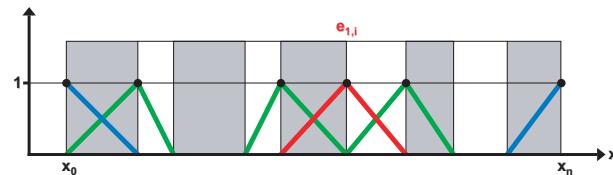
$$\dim(S_k(\Omega_n)) = k + n$$

Beispiel 189.

- für $k = 1$ erhalten wir lineare Splines, d.h. Polygonzüge



- der Raum der linearen Splines $S_1(\Omega_n)$ zur Unterteilung $\Omega_n = \{x_0, \dots, x_n\}$ hat Dimension $n + 1$
- wir betrachten die $n + 1$ Spline-Funktionen $e_{1,i}$ von $S_1(\Omega_n)$



- die $e_{1,i}$ haben folgenden Eigenschaften:
 - die $e_{1,i}$ sind einfache, lokale Funktionen, d.h.
- $$e_{1,i}(x) \geq 0 \quad \forall x, \quad e_{1,i}(x) = 0 \quad \forall x \notin [x_i, x_{i+2}]$$
- sie unterscheiden sich gegenseitig im wesentlichen durch eine einfache Verschiebung entlang der x -Achse
 - es gilt
- $$\sum_{i=0}^n e_{1,i}(x) = 1 \quad \forall x \in [x_0, x_n]$$
- die Spline-Funktionen $e_{1,i}, i = 0, \dots, n$ bilden eine Basis von $S_1(\Omega_n)$
- auch für Splines höherer Ordnung kann man entsprechende Basen finden

Satz 190. Es sei $\Omega_n = \{x_0, \dots, x_n\}$ eine Unterteilung von $[x_0, x_n]$. Dann gibt es genau $k + n$ **Basis-Splines** $e_{k,i}(x), i = -k, \dots, n - 1$, der Ordnung k mit folgenden Eigenschaften:

- es ist

$$e_{k,i}(x) \geq 0 \quad \text{auf } [x_i, x_{i+k+1}], \quad e_{k,i}(x) = 0 \quad \text{sonst} \quad (8.13)$$

- es gilt

$$\sum_{i=-k}^{n-1} e_{k,i}(x) = 1 \quad \forall x \in [x_0, x_n]$$

- die Spline-Funktionen $e_{k,i}$ bilden eine Basis von $S_k(\Omega_n)$

- Basis-Splines mit diesen Eigenschaften lassen sich leicht berechnen

Satz 191. Es sei $\Omega_n = \{x_0, \dots, x_n\}$ eine Unterteilung von $[x_0, x_n]$ und

$$x_{-k} < \dots < x_{-1} < x_0, \quad x_n < x_{n+1} < \dots < x_{n+k}.$$

Dann sind die rekursiv definierten Funktionen $e_{k,i}(x)$, $k \geq 1$, $i = -k, \dots, n-1$

$$e_{0,i}(x) = \begin{cases} 1 & \text{für } x \in [x_i, x_{i+1}], \\ 0 & \text{sonst} \end{cases}, \quad (8.14)$$

$$e_{k,i}(x) = \frac{x - x_i}{x_{i+k} - x_i} e_{k-1,i}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} e_{k-1,i+1}(x) \quad (8.15)$$

Basis-Splines im Sinne von Satz 190 auf dem Intervall $[x_0, x_n]$.

Bemerkung 192.

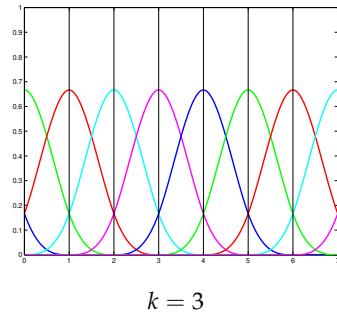
- ◊ die Rekursionsformel ist einfach implementierbar
- ◊ analog kann man auch Ableitungen der $e_{k,i}(x)$ berechnen
- ◊ die Hilfspunkte

$$x_{k-1} < \dots < x_{-1}, \quad x_{n-1} < \dots < x_{n+k}$$

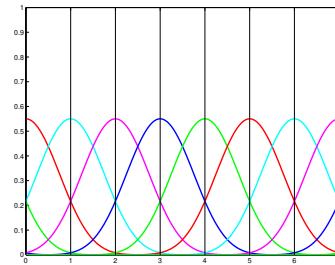
sind frei wählbar

Beispiel 193.

- wir betrachten die B-Splines zur Unterteilung $\Omega_7 = \{0, 1, \dots, 7\}$ für die Hilfspunkte $x_i = i$

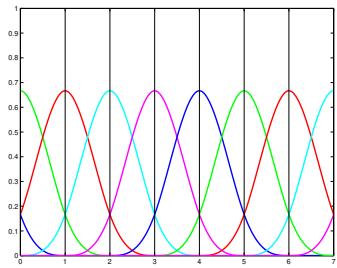


$$k = 3$$

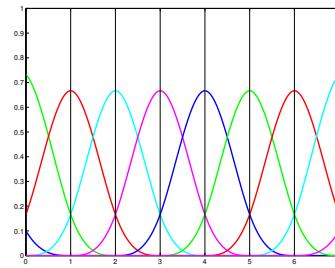


$$k = 5$$

- eine andere Wahl der Hilfspunkte beeinflusst die Form der B-Splines am Rand, wie wir an den folgenden kubischen Splines zur Unterteilung $\Omega_7 = \{0, 1, \dots, 7\}$ sehen:



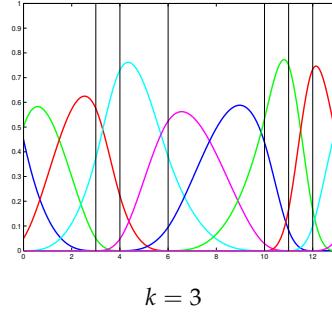
$$x_i = i$$



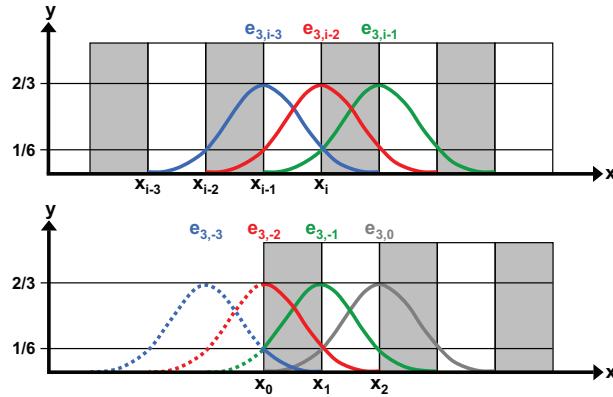
$$x_{-3} = -9, x_{-2} = -4, x_{-1} = -1$$

$$x_8 = 8, x_9 = 11, x_{10} = 16$$

- auf unregelmäßigen Unterteilungen wie $\Omega_7 = \{0, 3, 4, 6, 10, 11, 12, 13\}$ (hier mit Hilfsknoten $x_{-3} = -3, x_{-2} = -2, x_{-1} = -1, x_8 = 14, x_9 = 15, x_{10} = 16$) erhalten wir weniger regelmäßige B-Splines



- wir benutzen nun die kubischen Basissplines $e_{3,i}, i = -3, \dots, n-1$ um einen natürlichen kubischen Spline durch die Punkte $(x_i, y_i), i = 0, \dots, n$ zu interpolieren
- der Übersichtlichkeit halber beschränken wir uns auf äquidistante Unterteilungen (mit äquidistanten Hilfspunkten) $x_i = i \cdot h, h > 0$ gegeben
- die kubischen Basis-Splines auf $[x_0, x_n]$ sehen wie folgt aus



- man beachte, dass für alle x_i gilt

$$e_{3,i-3}(x_i) = e_{3,i-1}(x_i) = \frac{1}{6}, \quad e_{3,i-2}(x_i) = \frac{2}{3}, \quad e_{3,j}(x_i) = 0 \quad \text{für alle anderen } j \quad (8.16)$$

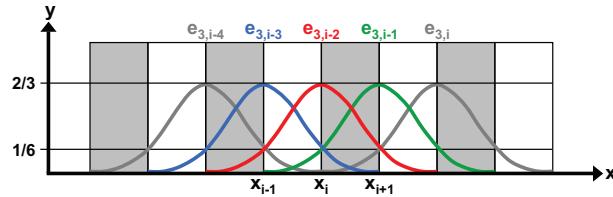
- der gesuchte interpolierende Spline s ist eine Linearkombination der kubischen Basis-Splines $e_{3,i}, i = -3, \dots, n-1$, d.h.

$$s(x) = \alpha_{-3}e_{3,-3}(x) + \alpha_{-2}e_{3,-2}(x) + \dots + \alpha_{n-1}e_{3,n-1}(x), \quad (8.17)$$

wobei die $n+3$ Koeffizienten α_i noch zu bestimmen sind

- dazu benutzen wir zunächst die $n+1$ Interpolationsbedingungen

$$s(x_i) = y_i, \quad i = 0, \dots, n$$



- für die Stützstelle x_i erhalten wir wegen (8.16)

$$\frac{1}{6}\alpha_{i-3} + \frac{2}{3}\alpha_{i-2} + \frac{1}{6}\alpha_{i-1} = y_i,$$

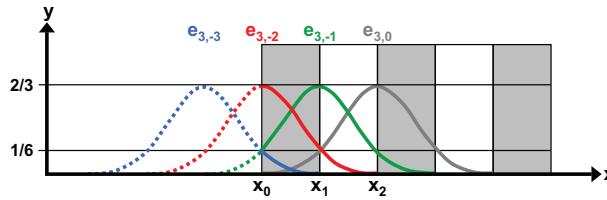
also eine lineare Gleichung, die drei benachbarte Koeffizienten verknüpft

- damit haben wir $n+1$ Gleichungen für $n+3$ Unbekannte
- die fehlenden beiden Gleichungen erhalten wir wieder aus den Abschlussbedingungen
- wir betrachten hier nur den Fall natürlicher Abschlussbedingungen

$$s''(x_0) = s''(x_n) = 0, \quad (8.18)$$

die anderen Abschlussbedingungen werden analog behandelt

- wir setzen (8.17) in (8.18) ein und betrachten x_0



- es gilt also

$$e''_{3,-3}(x_0), e''_{3,-2}(x_0), e''_{3,-1}(x_0) \neq 0, \quad e''_{3,i}(x_j) = 0 \quad \text{für alle anderen } j$$

- damit erhalten wir aus $s''(x_0) = 0$ die lineare Gleichung

$$e''_{3,-3}(x_0)\alpha_{-3} + e''_{3,-2}(x_0)\alpha_{-2} + e''_{3,-1}(x_0)\alpha_{-1} = 0$$

- für x_n erhalten wir analog

$$e''_{3,n-3}(x_n)\alpha_{n-3} + e''_{3,n-2}(x_n)\alpha_{n-2} + e''_{3,n-1}(x_n)\alpha_{n-1} = 0$$

- damit sind die gesuchten Koeffizienten α_i Lösung des linearen Gleichungssystems

$$\begin{pmatrix} e''_{3,-3}(x_0) & e''_{3,-2}(x_0) & e''_{3,-1}(x_0) \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ e''_{3,n-3}(x_n) & e''_{3,n-2}(x_n) & e''_{3,n-1}(x_n) \end{pmatrix} \begin{pmatrix} \alpha_{-3} \\ \alpha_{-2} \\ \vdots \\ \vdots \\ \alpha_{n-2} \\ \alpha_{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ y_0 \\ \vdots \\ \vdots \\ y_n \\ 0 \end{pmatrix}$$

- man kann zeigen, dass das System regulär ist
- die zweiten Ableitungen der $e_{3,i}$ können ähnlich wie die Funktionswerte (vergleiche Satz 191) rekursiv berechnet werden
- diese Vorgehensweise kann leicht auf Interpolationsaufgaben mit Splines beliebiger Ordnung übertragen werden:

- berechne rekursiv die Funktionswerte der Splines (sowie ggf. deren Ableitungen für die Abschlussbedingungen)
- stelle aus den Interpolationsbedingungen und den Abschlussbedingungen ein lineares Gleichungssystem auf
- löse das System und erhalte daraus die Linearfaktoren der einzelnen Basis-Splines

8.5 Interpolierende Kurven

- bisher haben wir zu Datenpunkten

$$(x_i, y_i), \quad x_i, y_i \in \mathbb{R}$$

Funktionen $f : \mathbb{R} \rightarrow \mathbb{R}$ gesucht (z.B. Polynome oder Splines) mit

$$f(x_i) = y_i \quad \forall i$$

- damit das funktioniert, haben wir immer angenommen, dass die x_i paarweise verschieden sind
- was können wir tun, wenn das nicht mehr der Fall ist?

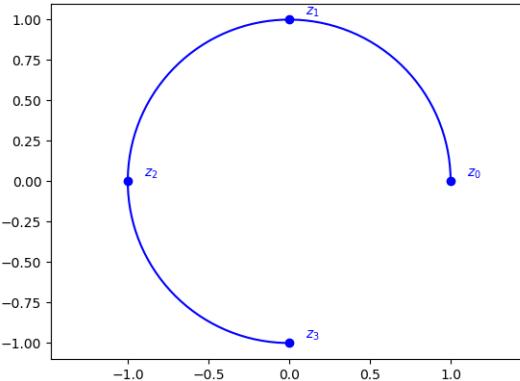
Beispiel 194.

- wir betrachten den Graph der Funktion

$$\gamma : [0, 1] \rightarrow \mathbb{R}^2, \quad \gamma(t) = \begin{pmatrix} \cos(\frac{3}{2}\pi t) \\ \sin(\frac{3}{2}\pi t) \end{pmatrix}$$

und die Punkte $z_i = \gamma(t_i)$ für

$$t_i = \frac{i}{3}, \quad i = 0, 1, 2, 3$$



- die Punkte $z_i = (x_i, y_i)$ können nicht mit einer einfachen Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ interpoliert werden, da zu $x_1 = x_3 = 0$ zwei unterschiedliche Werte $y_1 = 1$ bzw. $y_3 = -1$ vorliegen
- statt einfacher Funktionen benutzen wir jetzt Kurven zur Interpolation

Definition 195.

- der Graph einer stetige Abbildung

$$\gamma : \mathbb{R} \supset [a, b] \rightarrow \mathbb{R}^m, \quad t \rightarrow \gamma(t)$$

heißt **Kurve** in \mathbb{R}^m

- $t \in [a, b]$ ist der **Kurvenparameter**

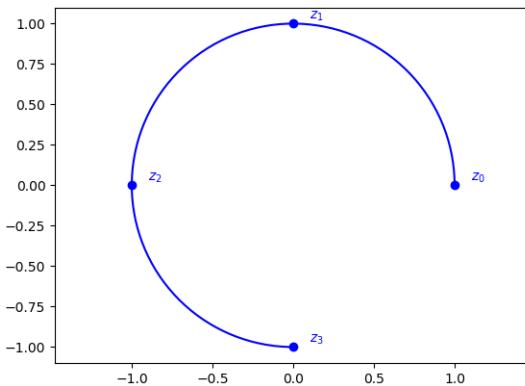
Beispiel 196.

- wir betrachten die Kurven $\gamma_k : [0, 1] \rightarrow \mathbb{R}^2$

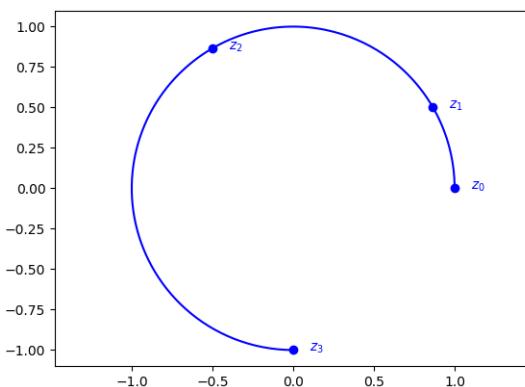
$$\gamma_1(t) = \begin{pmatrix} \cos(\frac{3}{2}\pi t) \\ \sin(\frac{3}{2}\pi t) \end{pmatrix}, \quad \gamma_2(t) = \begin{pmatrix} \cos(\frac{3}{2}\pi t^2) \\ \sin(\frac{3}{2}\pi t^2) \end{pmatrix}, \quad \gamma_2(t) = \begin{pmatrix} \cos(\frac{3}{2}\pi(1-t)) \\ \sin(\frac{3}{2}\pi(1-t)) \end{pmatrix}$$

- die folgenden Abbildungen zeigen jeweils die Kurve sowie die Punkte $z_i = \gamma(t_i)$ für

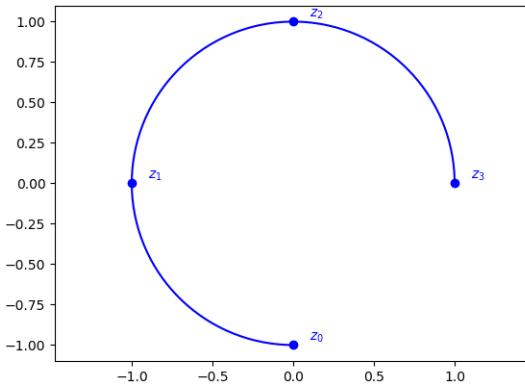
$$t_i = \frac{i}{3}, \quad i = 0, 1, 2, 3$$



γ_1



γ_2



γ_3

- die Graphen sind identisch, die Parametrierung der Kurven ist unterschiedlich

- unsere neue Aufgabenstellung lautet jetzt:

- gegeben (t_i, z_i) , $t_i \in [a, b]$, $z_i \in \mathbb{R}^m$, $i = 0, \dots, n$
- finde $f : [a, b] \rightarrow \mathbb{R}^m$ mit $f(t_i) = z_i \forall i$

- für f werden wir die "einfachen" Polynome bzw. Splines von oben verwenden
- interpolierende polynomiale Kurven:

- wir benutzen

$$f(t) = \begin{pmatrix} p_1(t) \\ \vdots \\ p_m(t) \end{pmatrix}$$

wobei die $p_j(t)$ Polynome vom Grad n sind

- wegen

$$\mathbb{R}^m \ni z_i = \begin{pmatrix} z_{i1} \\ \vdots \\ z_{im} \end{pmatrix}$$

folgt aus $f(t_i) = z_i$

$$p_j(t_i) = z_{ij}, \quad i = 0, \dots, n, \quad j = 1, \dots, m,$$

d.h. wir haben m "einfache" Interpolationspolynome p_j zu den Daten (t_i, z_{ij}) , $i = 0, \dots, n$, zu bestimmen

- interpolierende Spline-Kurven:

- wir benutzen

$$f(t) = \begin{pmatrix} s_1(t) \\ \vdots \\ s_m(t) \end{pmatrix}$$

wobei die $s_j(t)$ Splines mit der selben Ordnung k sind

- aus $f(t_i) = z_i$ folgt

$$s_j(t_i) = z_{ij}, \quad i = 0, \dots, n, \quad j = 1, \dots, m,$$

d.h. wir haben wieder m "einfache" interpolierende Splines s_j zu den Daten (t_i, z_{ij}) , $i = 0, \dots, n$, zu berechnen

Beispiel 197.

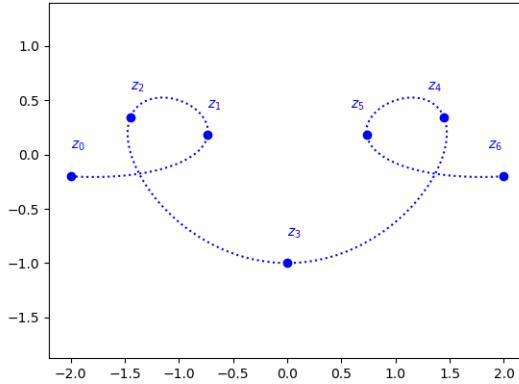
- wir betrachten die Kurve

$$\gamma : [-1, 1] \rightarrow \mathbb{R}^2, \quad \gamma(t) = \begin{pmatrix} 2t + \frac{\sin(2\pi t)}{t^2+1} \\ -\frac{\cos(2\pi t)}{4t^2+1} \end{pmatrix}$$

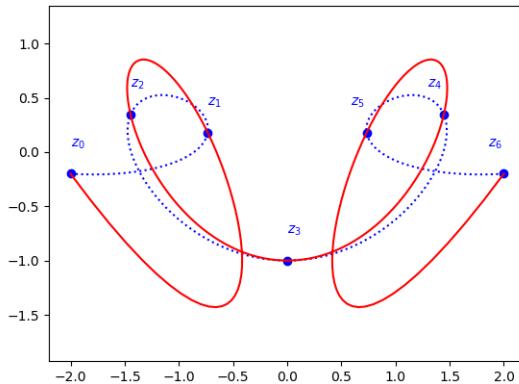
und benutzen die Kurvenpunkte

$$z_i = \gamma(t_i), \quad t_i = \frac{i-3}{3}, \quad i = 0, \dots, 6$$

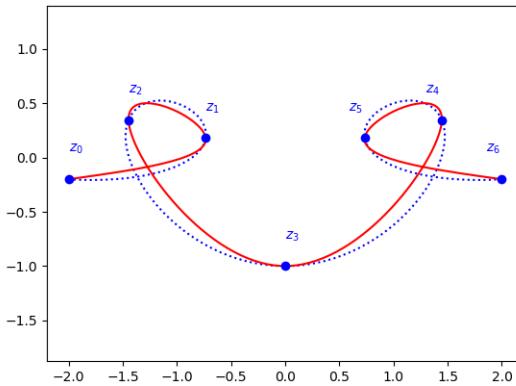
als Input für die Kurveninterpolation



- als polynomiale Interpolierende (vom Grad 6) erhalten wir



- die Spline-Interpolierende (kubische Splines, natürliche Abschlussbedingung) hat folgende Form



- auch hier kann man die Probleme bei interpolierenden Polynomen höheren Grades deutlich erkennen

Beispiel 198.

- wie wir in Beispiel 196 gesehen haben, definieren $\gamma_k : [0, 1] \rightarrow \mathbb{R}^2$

$$\gamma_1(t) = \begin{pmatrix} \cos(\frac{3}{2}\pi t) \\ \sin(\frac{3}{2}\pi t) \end{pmatrix}, \quad \gamma_2(t) = \begin{pmatrix} \cos(\frac{3}{2}\pi t^2) \\ \sin(\frac{3}{2}\pi t^2) \end{pmatrix}$$

identische Kurven, obwohl die Parametrierung verschieden ist

- benutzen wir für γ_1

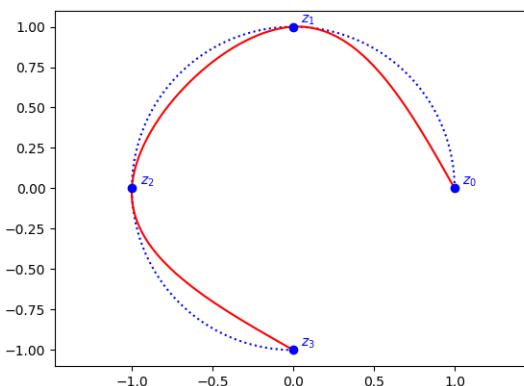
$$z_i = \gamma_1(t_i), \quad t_i = \frac{i}{3}, \quad i = 0, 1, 2, 3$$

bzw. für γ_2

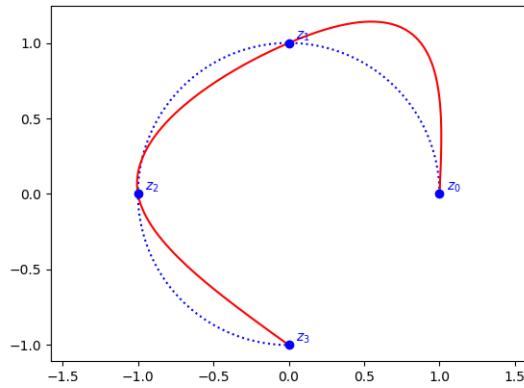
$$\tilde{z}_i = \gamma_2(\tilde{t}_i), \quad \tilde{t}_i = \sqrt{\frac{i}{3}}, \quad i = 0, 1, 2, 3$$

dann gilt $z_i = \tilde{z}_i$, lediglich die zugehörigen Parameterwerte unterscheiden sich

- betrachten wir nun wieder die Spline-Interpolierende (kubische Splines, natürliche Abschlussbedingung), so erhalten wir



γ_1



γ_2

- ▶ die Form der Interpolierenden hängt von der Parametrierung (Wahl der t_i) ab
- ▶ dieser zusätzliche Freiheitsgrad kann in der Praxis benutzt werden, um die Form der interpolierenden Kurven zu beeinflussen

Bemerkung 199.

- ◊ analog zu Kurven lassen sich auch interpolierende Flächen erzeugen
- ◊ Spline-Kurven und -Flächen bilden die Grundlage für CAD-Werkzeuge und damit praktisch für jede Software zur Computer-gestützten Modellierung von Körpern/Bauteilen

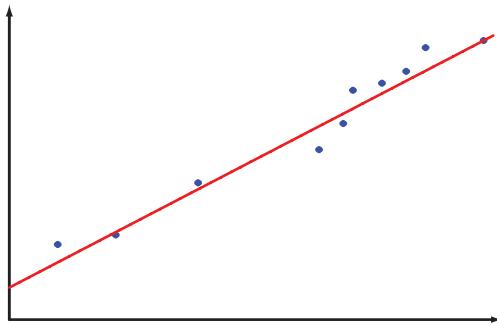
Kapitel 9

Approximation

9.1 Lineare Ausgleichsprobleme

- gegeben sind Punkte $(x_i, y_i) \in \mathbb{R}^2, i = 1, \dots, m$
- die Punkte (x_i, y_i) sollen nicht interpoliert, sondern nur durch eine Funktion approximiert werden

Beispiel 200. Versuche durch 10 gegebene Punkte eine Gerade f zu legen, so dass Punkte möglichst "nahe" an f liegen (**Ausgleichsgerade**):



- was heißt "nahe":
 - betrachte die Abweichung der Funktionswerte an jeder Stützstelle x_i :

$$e_i = f(x_i) - y_i$$

- summiere die e_i^2 auf:

$$\tilde{e} = \sqrt{\sum_{i=1}^m e_i^2} = \|e\|_2, \quad e = \begin{pmatrix} e_1 \\ \vdots \\ e_m \end{pmatrix}$$

- bestimme f so, dass \tilde{e} minimal wird
- lassen wir statt einer Geraden f ein Polynom vom Grad $n - 1$ zu, so erhalten wir folgende Aufgabenstellung:

- zu $(x_i, y_i), i = 1, \dots, m$, soll ein Polynom

$$f(x) = p_1 + p_2 x + \dots + p_n x^{n-1}$$

gesucht werden, so dass

$$\|e\|_2 = \left\| \begin{pmatrix} f(x_1) - y_1 \\ \vdots \\ f(x_m) - y_m \end{pmatrix} \right\|_2$$

minimal wird

- ist $m = n$, so erhalten wir die klassische Polynominterpolation und damit $\|e\|_2 = 0$
- wir betrachten hier $n \leq m$, d.h. die Anzahl der Punkte ist groß gegenüber der Anzahl der Polynomkoeffizienten (was sinnvoll ist, da bei hohem Polynomgrad Stabilitätsprobleme auftreten)
- wir bringen das Problem jetzt in eine neue Form

$$\begin{aligned} \|e\|_2 &= \left\| \begin{pmatrix} f(x_1) - y_1 \\ \vdots \\ f(x_m) - y_m \end{pmatrix} \right\|_2 \\ &= \left\| \begin{pmatrix} p_1 + p_2 x_1 + \dots + p_n x_1^{n-1} - y_1 \\ \vdots \\ p_1 + p_2 x_m + \dots + p_n x_m^{n-1} - y_m \end{pmatrix} \right\|_2 \\ &= \left\| \underbrace{\begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^{n-1} \end{pmatrix}}_{A \in \mathbb{R}^{n \times m}} \underbrace{\begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}}_{p \in \mathbb{R}^n} - \underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}}_{y \in \mathbb{R}^m} \right\|_2 \\ &= \|Ap - y\|_2 \end{aligned}$$

- wir können die Approximationsaufgabe lösen, falls

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

lösbar ist

Satz 201. Sei $A \in \mathbb{R}^{m \times n}$, $m \geq n$, $\text{rang}(A) = n$ (also maximal), dann hat für jedes $b \in \mathbb{R}^m$ das Minimalproblem

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

genau eine Lösung $\hat{x} \in \mathbb{R}^n$. Die Lösung \hat{x} löst auch die **Normalgleichungen**

$$A^T A \hat{x} = A^T b$$

wobei $A^T A \in \mathbb{R}^{n \times n}$ regulär ist.

Beweis.

- der Beweis besteht aus zwei Schritten
 - wir zeigen zunächst, dass $A^T A$ regulär ist und setzen $\hat{x} = (A^T A)^{-1} A^T b$
 - anschließend weisen wir nach, dass \hat{x} der einzige Minimierer von $\|Ax - b\|_2^2$ ist
- $A^T A$ ist regulär:
 - aus $A^T A x = 0$ folgt

$$0 = x^T A^T A x = (Ax)^T A x = \|Ax\|_2^2$$
 und damit $Ax = 0$
 - da nach Voraussetzung A maximalen Rang hat, muss $x = 0$ sein
 - damit haben wir gezeigt, dass aus $A^T A x = 0$ zwingend $x = 0$ folgt, weshalb $A^T A$ regulär ist
- $\hat{x} = (A^T A)^{-1} A^T b$ minimiert $\|Ax - b\|_2^2$:
 - dazu betrachten wir

$$\begin{aligned} J(x) &= \|Ax - b\|_2^2 \\ &= (Ax - b)^T (Ax - b) \\ &= (x^T A^T - b^T)(Ax - b) \\ &= x^T (A^T A x - A^T b) + b^T (b - Ax) \end{aligned}$$

und somit

$$J(\hat{x}) = \hat{x}^T ((A^T A)(A^T A)^{-1} A^T b - A^T b) + b^T (b - A\hat{x}) = b^T (b - A\hat{x})$$

- ist $\hat{y} \in \mathbb{R}^n$, $\hat{y} \neq \hat{x}$ so gilt $\hat{y} = \hat{x} + v$, $v \in \mathbb{R}^n$, $v \neq 0$ und

$$\begin{aligned} J(\hat{y}) &= J(\hat{x} + v) \\ &= (\hat{x} + v)^T (A^T A(\hat{x} + v) - A^T b) + b^T (b - A(\hat{x} + v)) \\ &= (\hat{x} + v)^T (A^T A\hat{x} + A^T A v - A^T b) + b^T (b - A\hat{x} - A v) \\ &\stackrel{A^T A\hat{x}=A^T b}{=} (\hat{x} + v)^T A^T A v + b^T (b - A\hat{x}) - b^T A v \\ &\stackrel{\text{alles in } \mathbb{R}}{=} v^T A^T A(\hat{x} + v) - v^T A^T b + b^T (b - A\hat{x}) \\ &= v^T (A^T A\hat{x} - A^T b) + v^T A^T A v + b^T (b - A\hat{x}) \\ &= \|Av\|_2^2 + J(\hat{x}) \end{aligned}$$

- da $v \neq 0$ und A maximalen Rang hat, ist $Av \neq 0$ und $\|Av\|_2^2 > 0$, so dass für $\hat{y} \neq \hat{x}$

$$J(\hat{y}) = \|A(\hat{y} - \hat{x})\|_2^2 + J(\hat{x}) > J(\hat{x})$$

und \hat{x} somit das einzige Minimum von J ist

□

- der letzte Satz liefert damit einen ersten Algorithmus zur numerischen Behandlung der Polynomapproximation

- stelle Normalgleichungssystem $A^T A x = A^T b$ mit

$$A = \begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^{n-1} \end{pmatrix}, \quad b = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}, \quad x = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$$

auf und löse es ($A^T A$ ist spd, so dass das Cholesky-Verfahren oder iterative Methoden benutzt werden können)

- die Lösung x liefert die Polynomkoeffizienten

Beispiel 202. Wir bestimmen die Ausgleichsgerade durch die Punkte $(0, 1), (2, 2), (3, 2)$

- wir haben 3 Punkte, d.h. $m = 3$
- $f(x) = p_1 + p_2 x$, d.h. $n = 2$ und

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \quad x = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

- für die Normalgleichungen $A^T A x = A^T b$ erhalten wir

$$A^T A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \\ 1 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} 3 & 5 \\ 5 & 13 \end{pmatrix}$$

bzw.

$$A^T b = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & 3 \\ 1 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 5 \\ 10 \\ 2 \end{pmatrix}$$

- $A^T A x = A^T b$ bedeutet also

$$\begin{aligned} \begin{pmatrix} 3 & 5 \\ 5 & 13 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} &= \begin{pmatrix} 5 \\ 10 \end{pmatrix} \\ \Leftrightarrow \begin{pmatrix} 3 & 5 \\ 0 & \frac{14}{3} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} &= \begin{pmatrix} 5 \\ \frac{5}{3} \end{pmatrix} \end{aligned}$$

und somit

$$p_2 = \frac{5}{14}, \quad p_1 = \frac{15}{14},$$

so dass wir als Ausgleichsgrade

$$f(x) = \frac{15}{14} + \frac{5}{14}x$$

erhalten

- das Approximationsproblem kann wie folgt verallgemeinert werden:
 - statt Polynomen kann man auch Linearkombinationen beliebiger Basisfunktionen $\varphi_1, \dots, \varphi_n : \mathbb{R} \rightarrow \mathbb{R}$ benutzen
 - suche eine approximierende Funktion f als Linearkombination dieser Basisfunktionen

$$f(x) = p_1 \varphi_1(x) + \dots + p_n \varphi_n(x)$$

und bestimme die Koeffizienten p_1, \dots, p_n so, dass

$$\left\| \begin{pmatrix} f(x_1) - y_1 \\ \vdots \\ f(x_m) - y_m \end{pmatrix} \right\|_2$$

minimal wird

- mit $\varphi_1(x) = 1, \varphi_2(x) = x, \dots, \varphi_n(x) = x^{n-1}$ erhalten wir daraus das oben untersuchte polynomiale Approximationsproblem
- analog zu oben zeigt man

$$\left\| \begin{pmatrix} f(x_1) - y_1 \\ \vdots \\ f(x_m) - y_m \end{pmatrix} \right\|_2 = \|Ap - y\|_2, \quad A = \begin{pmatrix} \varphi_1(x_1) & \dots & \varphi_n(x_1) \\ \vdots & & \vdots \\ \varphi_1(x_m) & \dots & \varphi_n(x_m) \end{pmatrix},$$

d.h. lediglich A hat sich verändert

9.2 QR Zerlegung für lineare Ausgleichsprobleme

- die Normalgleichungen sind für die Numerik wenig geeignet:

$$A^T A = \begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^{n-1} \end{pmatrix}^T \begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^{n-1} \end{pmatrix} = \left(\sum_k x_k^{i-1} x_k^{j-1} \right)_{i,j=1,\dots,n}$$

- das Aufsummieren der Potenzen ist sehr rundungsfehleranfällig
- in der Praxis zeigt sich, dass eine numerische Lösung von $A^T A x = A^T b$ für größere m, n fast immer unbrauchbar ist
- deshalb suchen wir ein anderes Verfahren um $\|Ax - b\|_2$ zu minimieren
- dazu betrachten wir wieder orthonormale Matrizen Q (vergleiche Givens-Verfahren, Householder-Verfahren)
- mit $\|Qx\|_2 = \|x\|_2$ folgt

$$\|Ax - b\|_2 = \|Q(Ax - b)\|_2$$
 und $\|Ax - b\|_2$ wird minimal falls $\|QAx - Qb\|_2$ minimal wird
- wir versuchen $Q = Q_n \cdot \dots \cdot Q_1$, Q_i orthonormal, so zu wählen, dass $\|QAx - Qb\|_2$ einfach zu minimieren ist

- wir benutzen als Q_1, \dots, Q_n Householder-Matrizen um nacheinander die Spalten $1, \dots, n$ von A zu eliminieren, d.h.

$$QA = Q_n \cdot \dots \cdot Q_1 A = \underbrace{\begin{pmatrix} * & \dots & \dots & * \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & * \\ 0 & \dots & 0 & 0 \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & 0 \end{pmatrix}}_m =: \begin{pmatrix} R \\ 0 \end{pmatrix}$$

wobei $R \in \mathbb{R}^{n \times n}$ eine obere Dreiecksmatrix ist, die regulär ist, falls A vollen Rang n hat

- benutzen wir noch

$$Qb = \begin{pmatrix} c \\ d \end{pmatrix}, \quad c \in \mathbb{R}^n, \quad d \in \mathbb{R}^{m-n},$$

so folgt

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \|Ax - b\|_2 &= \min_{x \in \mathbb{R}^n} \|QAx - Qb\|_2 \\ &= \min_{x \in \mathbb{R}^n} \left\| \begin{pmatrix} R \\ 0 \end{pmatrix} x - \begin{pmatrix} c \\ d \end{pmatrix} \right\|_2 \\ &= \min_{x \in \mathbb{R}^n} \sqrt{\|Rx - c\|_2^2 + \|d\|_2^2} \end{aligned}$$

- für die beiden Terme in der Wurzel gilt:

- $\|d\|_2^2$ ist nicht beeinflussbar, sondern durch b fix gegeben
- $\|Rx - c\|_2^2$ muss durch x so klein wie möglich gemacht werden
- da $R \in \mathbb{R}^{n \times n}$ regulär hat $Rx - c = 0$ eine eindeutige Lösung,
- da R eine obere Dreiecksmatrix ist, kann die Lösung von $Rx = c$ einfach durch rückwärts einsetzen bestimmt werden

Zusammenfassung.

- $A \in \mathbb{R}^{m \times n}$, $m \geq n$, $\text{rang}(A) = n$, $b \in \mathbb{R}^n$ gegeben
- bestimme eindeutiges x mit $\|Ax - b\|_2$ minimal durch
 - Transformation aller n Spalten von A mit Householdermatrizen

$$A \rightarrow QA = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad b \rightarrow Qb = \begin{pmatrix} c \\ d \end{pmatrix}$$

- Lösung von $Rx = c$ durch rückwärts einsetzen

9.3 CGLS

- in vielen Anwendungen ist $\|Ax - b\|_2$ für sehr große, dünn besetzte Matrizen A zu minimieren
- eine Lösung mittels Householder-Transformationen ist sehr rechenintensiv ($\approx \min(m, n)^3$ Operationen für $A \in \mathbb{R}^{m \times n}$)
- betrachten wir also nochmals die Normalgleichungen

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

- $A^T A$ ist in der Regel dicht besetzt und sehr groß, so dass direkte Verfahren zum Lösen des Gleichungssystems einen ähnlich hohen Aufwand wie die Householder-Transformationen mit sich bringen
- $A^T A$ ist aber positiv definit, so dass einige der iterativen Löser anwendbar sind
- die explizite Berechnung von $A^T A$ sollte vermieden werden (Kondition)
- die Anwendung von $A^T A$ auf einen Vektor v ist aber wegen

$$A^T A v = A^T (Av)$$

leicht mit zwei Matrix-Vektor-Produkten zu bewerkstelligen

- nach diesen Vorüberlegungen formulieren wir das klassische CG-Verfahren

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k \tilde{A} p_k & \alpha_k &= \frac{\langle r_k, r_k \rangle}{\langle p_k, \tilde{A} p_k \rangle} \\ p_{k+1} &= r_{k+1} + \beta_k p_k & \beta_k &= \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle} \end{aligned}$$

für die Normalgleichungen

$$\tilde{A}x = \tilde{b}, \quad \tilde{A} = A^T A, \quad \tilde{b} = A^T b$$

so um, dass $A^T A$ nicht explizit berechnet werden muss:

- für das Residuum gilt

$$r_k = \tilde{b} - \tilde{A}x_k = A^T b - A^T A x_k = A^T(b - Ax_k) = A^T s_k, \quad s_k = b - Ax_k$$

mit

$$s_{k+1} = b - Ax_{k+1} = b - A(x_k + \alpha_k p_k) = s_k - \alpha_k A p_k$$

- für den Nenner von α_k folgt

$$\langle p_k, \tilde{A} p_k \rangle = \langle p_k, A^T A p_k \rangle = \langle A p_k, A p_k \rangle$$

- zusammen erhalten wir folgendes Verfahren

Definition 203. Das **CGLS-Verfahren** (Conjugate Gradient Least Squares) ist definiert durch:

- x_0 gegeben, $s_0 = b - Ax_0$, $p_0 = r_0 = A^T s_0$
- wiederhole für $k \geq 0$:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ s_{k+1} &= s_k - \alpha_k A p_k & \alpha_k &= \frac{\langle r_k, r_k \rangle}{\langle A p_k, A p_k \rangle} \\ r_{k+1} &= A^T s_{k+1} \\ p_{k+1} &= r_{k+1} + \beta_k p_k & \beta_k &= \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle} \end{aligned}$$

Satz 204. Die Iterierten x_k aus dem CGLS-Verfahren minimieren $\|Ax - b\|_2$ über der Hyperebene

$$x_0 + \text{span}(A^T r_0, (A^T A) A^T r_0, \dots, (A^T A)^{k-1} A^T r_0)$$

Bemerkung 205.

- ◊ hat $A \in \mathbb{R}^{m \times n}$ vollen Rang, d.h. $A^T A \in \mathbb{R}^{n \times n}$ ist regulär, so liefert das CGLS-Verfahren in exakter Arithmetik nach spätestens n Schritten die eindeutige Lösung x des Ausgleichsproblems
- ◊ hat A keinen vollen Rang, so konvergiert CGLS immer noch gegen einen (der vielen möglichen) Minimierer (siehe nächstes Kapitel) von $\|Ax - b\|_2$
- ◊ auch für CGLS gibt es vorkonditionierte Varianten

9.4 Pseudoinverse

- in den letzten Abschnitten haben wir für $A \in \mathbb{R}^{m \times n}$, $m \geq n$ mit maximalem Rang $\text{rang}(A) = n$ das Minimalproblem

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

betrachtet

- es hat für jedes $b \in \mathbb{R}^m$ genau eine Lösung $\hat{x} \in \mathbb{R}^n$ (Normalgleichungen)
- wie sieht das für allgemeine A aus, also m, n beliebig und $\text{rang}(A)$ nicht unbedingt maximal?

Beispiel 206.

- betrachte

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

- $\|Ax - b\|_2 = 0$ für alle x der Form

$$x = \begin{pmatrix} 2 \\ \alpha \end{pmatrix}, \quad \alpha \in \mathbb{R}$$

- der Minimierer ist also nicht eindeutig

- um im allgemeinen Fall Eindeutigkeit zu erhalten muss eine weitere Bedingung in das Minimalproblem eingefügt werden:

$$X = \{z \mid \|Az - b\|_2 \text{ minimal}\}, \quad x = \operatorname{argmin}_{z \in X} \|z\|_2 \quad (9.1)$$

- unter allen Minimierern von $\|Az - b\|_2$ wird also derjenige ausgesucht, der selbst noch die kleinste Norm besitzt

Satz 207. Sei $A \in \mathbb{R}^{m \times n}$ beliebig, $b \in \mathbb{R}^m$. Dann gibt es genau eine Lösung $x \in \mathbb{R}^n$ von (9.1). Diese hängt linear von b , d.h. es existiert eine eindeutige, von b unabhängige Matrix $A^+ \in \mathbb{R}^{n \times m}$ mit

$$x = A^+b, \quad \forall b \in \mathbb{R}^m.$$

A^+ heißt **Pseudo-Inverse** zu A .

Bemerkung 208. Für $A \in \mathbb{R}^{n \times n}$ regulär ist $A^+ = A^{-1}$

Satz 209. Die Pseudo-Inverse A^+ zu A ist durch die folgenden vier **Penrose-Axiome** festgelegt

$$(A^+A)^T = A^+A, \quad (AA^+)^T = AA^+, \quad A^+AA^+ = A^+, \quad AA^+A = A$$

Beispiel 210.

- betrachte

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

- dann ist

$$Ax - b = \begin{pmatrix} 2x_1 - b_1 \\ -b_2 \end{pmatrix}$$

und $\|Ax - b\|_2 = \sqrt{(2x_1 - b_1)^2 + b_2^2}$ wird minimal für alle x der Form

$$x = \begin{pmatrix} \frac{b_1}{2} \\ \alpha \end{pmatrix}, \quad \alpha \in \mathbb{R}$$

- $\|x\|_2$ wird minimal, falls $x_2 = \alpha = 0$

- zusammen erhalten wir für die Lösung x

$$x = \begin{pmatrix} \frac{b_1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix} b$$

- mit den Penrose-Axiomen kann man zeigen, dass

$$A^+ = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix}$$

wirklich die Pseudo-Inverse zu A ist

- die beiden ersten Axiome sind erfüllt, denn

$$(A^+ A)^T = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = A^+ A, \quad A^+ A A^+ = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix} = A^+$$

- die beiden anderen Penrose-Gleichungen gelten analog

- $A^+ b$ ist also die eindeutige Lösung von (9.1)
- wie berechnet man $A^+ b$?
- mit Hilfe von Householder-Transformationen kann man den Zugang aus Kapitel 9.2 so erweitern, dass man damit auch die Pseudoinverse bestimmen kann
- es handelt sich dabei um ein direktes (nicht iteratives) Verfahren
- dieses Verfahren wird in der Praxis selten eingesetzt, da es effizientere (iterative) Verfahren gibt, die auf den Ergebnissen des nächsten Abschnitts beruhen

9.5 Singulärwertzerlegung

- die Singulärwertzerlegung einer Matrix A steht in engem Zusammenhang mit den hier betrachteten Approximationsaufgaben bzw. der Pseudo-Inversen A^+

Satz 211. Sei $A \in \mathbb{R}^{m \times n}$ beliebig. Dann gibt es orthonormale Matrizen $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, so dass

$$U^T A V = \Sigma$$

mit

$$\Sigma \in \mathbb{R}^{m \times n}, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_p), \quad p = \min(m, n)$$

und

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0.$$

$U^T A V = \Sigma$ bzw. $A = U \Sigma V^T$ heißt **Singulärwertzerlegung** von A mit **Singulärwerten** σ_i .

Bemerkung 212.

- ◊ Σ hat die Form

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \sigma_n \\ 0 & \dots & \dots & 0 \\ \vdots & & & \vdots \\ 0 & \dots & \dots & 0 \end{pmatrix}, \quad m \geq n \quad \text{bzw.} \quad \Sigma = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \dots & 0 & \sigma_m & 0 & \dots & 0 \end{pmatrix}, \quad m \leq n$$

- ◊ ist $A \in \mathbb{R}^{n \times n}$ spd, dann gilt

$$A = Q \Lambda Q^T, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n) \quad Q = (q_1, \dots, q_n)$$

wobei $\lambda_1 \geq \dots \geq \lambda_n$ die Eigenwerte von A und q_i die zugehörigen Eigenvektoren sind, d.h.

$$\Sigma = \Lambda, \quad U = V = Q$$

- im folgenden Satz fassen wir die wichtigsten Eigenschaften der Singulärwertzerlegung zusammen

Satz 213. Es sei $A = U\Sigma V^T$ eine Singulärwertzerlegung der Matrix $A \in \mathbb{R}^{m \times n}$. Dann gilt:

- $V\Sigma^T U^T$ ist eine Singulärwertzerlegung von A^T , d.h. die Singulärwerte von A und A^T sind identisch
- gilt $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0$ dann ist $\text{rang}(A) = r$
- $\sigma_1^2, \dots, \sigma_p^2$ sind Eigenwerte von $A^T A$ bzw. $A A^T$, alle weiteren Eigenwerte der beiden Matrizen sind gleich 0
- die Spalten von U, V sind die Eigenvektoren von $A A^T$ bzw. $A^T A$
- ist $r = \text{rang}(A)$, $A = U\Sigma V^T$ mit

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}$$

dann ist

$$A^+ = V\Sigma^+ U^T, \quad \Sigma^+ = \text{diag}\left(\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0\right) \in \mathbb{R}^{n \times m}$$

Bemerkung 214. Damit können wir die Pseudoinverse wie folgt berechnen:

- erzeuge eine Singulärwertzerlegung $A = U\Sigma V^T$
- bestimme $r = \text{rang}(A)$, d.h. die Anzahl der Singulärwerte, die größer 0 sind
- berechne $x = A^+ b$ durch

$$x = A^+ b = V\Sigma^+ U^T b$$

- wie ermittelt man eine Singulärwertzerlegung?
 - nach oben wissen wir, dass $\sigma_1^2, \dots, \sigma_p^2$ die Eigenwerte und die Spalten von U, V die Eigenvektoren von $A A^T$ bzw. $A^T A$ sind
 - im Prinzip kann man die Singulärwertzerlegung also durch Lösen von Eigenwert- bzw. Eigenvektorproblemen für die symmetrischen Matrizen $A A^T$ bzw. $A^T A$ bestimmen
 - ein mögliches Verfahren dafür wäre die QR-Iteration
 - analog zu CGLS möchte man die explizite Berechnung von $A A^T, A^T A$ vermeiden
 - durch geschickte Ausnutzung der Eigenschaften der Singulärwertzerlegung kann man die beiden wesentlichen Teile des QR-Verfahrens für symmetrische Matrizen (Hessenberg-Transformation auf Tridiagonalgestalt, QR-Iteration mit Tridiagonalmatrizen) so übertragen, dass ein Algorithmus entsteht, der direkt auf der Matrix A arbeitet
- dazu benutzen wir die folgende Eigenschaft

Lemma 215. Ist $A \in \mathbb{R}^{m \times n}$ und sind $S \in \mathbb{R}^{m \times m}, T \in \mathbb{R}^{n \times n}$ orthonormale Matrizen, dann haben A und SAT dieselben Singulärwerte

- wir benutzen also orthonormale Matrizen S, T um A so umzuformen, dass die Singulärwerte leichter zu bestimmen sind
- dazu können wir ohne Einschränkung $m \geq n$ annehmen (ansonsten betrachten wir A^T)
- wir eliminieren zuerst mit einer Householder-Matrix S_1 die erste Spalte von A

$$A \rightarrow S_1 A = \begin{pmatrix} * & * & * & \dots & * \\ 0 & * & * & \dots & * \\ 0 & * & * & \dots & * \\ \vdots & \vdots & & & \vdots \\ 0 & * & * & \dots & * \end{pmatrix}$$

- dann eliminieren wir mit einer Householder-Matrix T_1 die um eins verkürzte erste Zeile von $S_1 A$

$$S_1 A \rightarrow S_1 A T_1 = \begin{pmatrix} * & * & 0 & \dots & 0 \\ 0 & * & * & \dots & * \\ 0 & * & * & \dots & * \\ \vdots & \vdots & & & \vdots \\ 0 & * & * & \dots & * \end{pmatrix}$$

- bei diesem Schritt bleibt die erste Spalte von $S_1 A$ unverändert
- analog kann man alle weiteren Spalten und Zeilen behandeln und erhält schließlich nach n Schritten

$$SAT = \begin{pmatrix} * & * & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & * & * & 0 \\ \vdots & & \ddots & * & * \\ 0 & \dots & \dots & 0 & * \\ 0 & \dots & \dots & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & \dots & \dots & 0 \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \in \mathbb{R}^{n \times n} \quad (9.2)$$

wobei R eine obere Bidiagonal-Matrix ist und

$$S = S_n \cdot \dots \cdot S_1, \quad T = T_1 \cdot \dots \cdot T_{n-1},$$

- nach Lemma 215 haben A und $(R, 0)^T$ und damit A und R dieselben Singulärwerte
- diese Transformation auf Bidiagonalgestalt ist also das Analogon zur Hessenbergtransformation bei der Eigenwertberechnung
- im nächsten Schritt starten wir die folgende Iteration:

- $R_0 = R$
- wiederhole für $k > 0$

$$R_k^T = Q_{k+1} R_{k+1}, \quad R_{k+1}^T = Q_{k+2} R_{k+2} \quad (9.3)$$

- es werden immer zwei Schritte $R_k \rightarrow R_{k+2}$ durchgeführt
- R_k^T ist eine untere Bidiagonal-Matrix und $R_k^T = Q_{k+1} R_{k+1}$ eine QR-Zerlegung (z.B. mit dem Givens- oder Householder-Verfahren)
- analog zur Hessenberg-Transformation für Tridiagonal-Matrizen zeigt man, dass R_{k+1}, R_{k+2} auch wieder obere Bidiagonal-Matrizen sind
- außerdem gilt

$$R_{k+2} = Q_{k+2}^T R_{k+1}^T = Q_{k+2}^T (Q_{k+1}^T R_k^T)^T = Q_{k+2}^T R_k Q_{k+1}$$

- da Q_{k+1}, Q_{k+2} orthonormal sind, haben R_{k+2} und R_k nach Lemma 215 identische Singulärwerte
- die Iteration (9.3) ist das Analogon einer QR-Iteration ohne Shift angewandt auf die Matrix $R^T R$
- R_{2k} konvergiert gegen eine Diagonalmatrix, die die Singulärwerte enthält
- aus den orthonormalen Transformationsmatrizen und den Matrizen T, S aus dem ersten Teil können die Matrizen U und V approximiert werden

Zusammenfassung.

- die Singulärwertzerlegung wird ähnlich wie eine Eigenwert-/Eigenvektorberechnung für symmetrische Matrizen durchgeführt
- der wesentliche Aufwand besteht in der Transformation von A auf Bidiagonalgestalt mit Hilfe von Householdertransformationen
- der iterative Teil des Algorithmus arbeitet ausschließlich auf Bidiagonalmatrizen und ist deswegen sehr effizient ($\mathcal{O}(n)$ Operationen pro Iteration)

9.6 Regularisierung schlecht konditionierter Probleme

9.6.1 Grundlagen

- wie wir in Abschnitt 3.3.2 bereits gesehen haben, kann die numerische Behandlung schlecht konditionierter Probleme sehr schwierig sein
- wir betrachten hier Methoden, um schlecht konditionierte Gleichungssysteme bzw. schlecht konditionierte lineare Ausgleichsprobleme (Pseudoinverse) zu behandeln

Beispiel 216.

- wir betrachten ein lineares Gleichungssystem $Ax = b$ der Dimension $n = 15$ mit der **Hilbertmatrix**

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

wobei b so gewählt wurde, dass die exakte Lösung $x = (1, \dots, 1)^T$ ist

- mit der LU-Zerlegung mit Spalten-Pivot-Suche bzw. dem Householder-Verfahren erhalten wir bei doppeltgenauer Rechnung folgende Ergebnisse:

exakt	LU, Spalten-Pivot	Householder
1	1.0000	1.0000
1	1.0027	1.0000
1	0.9206	0.9999
1	1.9998	1.0019
1	-5.6806	0.9787
1	27.1804	1.1394
1	-61.8486	0.4329
1	93.3362	2.4692
1	-77.1182	-1.3684
1	31.5036	3.0570
1	0.9843	0.8566
1	-0.0967	-0.7446
1	0.0467	2.9170
1	-0.0150	0.0896
1	1.7847	1.1707

- beide Verfahren liefern Resultate, die erheblich von der exakten Lösung abweichen
- die relativen Fehler sind

$$\frac{\|x_{LU} - x\|_2}{\|x\|_2} \approx 36.74, \quad \frac{\|x_{HH} - x\|_2}{\|x\|_2} \approx 1.15$$

- die Abweichungen bei Householder sind geringer, aber trotzdem nicht akzeptabel (über 100%)
- die Konditionszahl von A bezüglich der Norm $\|\cdot\|_2$ ist

$$\varkappa_2(A) \approx 8.5 \cdot 10^{17}$$

- Householder verschlechtert zwar die Kondition des Problems beim Zerlegen nicht, allerdings muss die Kondition auch nicht verbessert werden
- die Maschinengenauigkeit bei doppeltgenauer IEEE-Arithmetik ist

$$\varepsilon_M \approx 1.1 \cdot 10^{-16},$$

was zusammen mit $\varkappa_2(A)$ die schlechten Resultate erklärt

Beispiel 217.

- wir wiederholen die Betrachtungen von Beispiel 216, stören aber die rechte Seite b noch wie folgt

$$\tilde{b} = b + \Delta b, \quad \Delta b = c (1, -1, 1, -1, \dots)^T, \quad c \in \mathbb{R}$$

wobei c so gewählt wird, dass

$$\|\Delta b\|_2 = 10^{-6}$$

- mit der LU-Zerlegung mit Spalten-Pivot-Suche bzw. dem Householder-Verfahren erhalten wir bei doppeltgenauer Rechnung folgende Ergebnisse:

exakt ungestört	LU, Spalten-Pivot	Householder
1	1.0104e1	3.5021e2
1	-8.9708e2	-4.4339e4
1	2.2261e4	1.3469e6
1	-2.4066e5	-1.6494e7
1	1.4028e6	9.1563e7
1	-4.8893e6	-1.4746e8
1	1.0733e7	-9.6850e8
1	-1.5127e7	6.7281e9
1	1.3561e7	-2.0302e10
1	-7.3813e6	3.6709e10
1	2.1458e6	-4.2460e10
1	-1.7897e5	3.1367e10
1	-5.7412e4	-1.4057e10
1	4.7274e3	3.3515e9
1	6.5344e3	-2.9622e8

- im letzten Beispiel versagt selbst unser 'stabilstes' Verfahren (Householder) vollständig
- wie können wir solche Probleme trotzdem vernünftig behandeln?
- dazu müssen wir das genaue Verhalten bei der Fehlerfortpflanzung von b nach x untersuchen
- dabei werden die Singulärwerte von A eine wichtige Rolle spielen

Beispiel 218.

- wir betrachten das lineare Gleichungssystem $Ax = b$ mit

$$A = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}, \quad \sigma_1 \geq \sigma_2 > 0$$

- da A invertierbar ist, ist die exakte Lösung gegeben durch

$$x = A^{-1}b, \quad A^{-1} = \begin{pmatrix} \frac{1}{\sigma_1} & 0 \\ 0 & \frac{1}{\sigma_2} \end{pmatrix}$$

- stört man die rechte Seite $\tilde{b} = b + \Delta b$, so erhält man

$$\tilde{x} = A^{-1}(b + \Delta b) = A^{-1}b + A^{-1}\Delta b,$$

- für den Fehler $\Delta x = \tilde{x} - x$ gilt also

$$\Delta x = \tilde{x} - x = A^{-1} \Delta b$$

bzw. für den relativen Fehler

$$e_r(x) = \frac{\|\Delta x\|_2}{\|x\|_2} = \frac{\|A^{-1} \Delta b\|_2}{\|A^{-1} b\|_2}$$

- mit $b = (1, 0)^T$ und $\Delta b = \delta(0, 1)^T$, $\delta > 0$ folgt

$$x = \begin{pmatrix} \frac{1}{\sigma_1} \\ 0 \end{pmatrix}, \quad \Delta x = \delta \begin{pmatrix} 0 \\ \frac{1}{\sigma_2} \end{pmatrix},$$

und somit

$$e_r(x) = \frac{\|\Delta x\|_2}{\|x\|_2} = \frac{\delta \frac{1}{\sigma_2}}{\frac{1}{\sigma_1}} = \frac{\sigma_1}{\sigma_2} \delta = \frac{\sigma_1}{\sigma_2} \frac{\|\Delta b\|_2}{\|b\|_2} = \frac{\sigma_1}{\sigma_2} e_r(b)$$

- der relative Fehler wird also mit dem Faktor

$$\frac{\sigma_1}{\sigma_2}$$

verstärkt

- ist $\frac{\sigma_1}{\sigma_2}$ sehr groß, dann ist das Problem schlecht konditioniert und Eingabefehler können sich dramatisch auf das Ergebnis auswirken

Beispiel 219.

- betrachten wir das lineare Ausgleichsproblem

$$\|Ax - b\|_2 \rightarrow \min, \quad \|x\|_2 \rightarrow \min$$

zu

$$A = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \dots & 0 & \sigma_r & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{pmatrix} = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}$$

mit $\sigma_1 \geq \dots \geq \sigma_r > 0$

- die exakte Lösung ist gegeben durch

$$x = A^+ b, \quad A^+ = \text{diag}\left(\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0\right) \in \mathbb{R}^{n \times m}$$

und analog zu Beispiel 218 erhält man für den relativen Fehler

$$e_r(x) = \frac{\|\Delta x\|_2}{\|x\|_2} = \frac{\|A^+ \Delta b\|_2}{\|A^+ b\|_2}$$

- mit $b = e_1$ und $\Delta b = \delta e_r$, $\delta > 0$, e_i i -ter Einheitsvektor, folgt

$$e_r(x) = \frac{\sigma_1}{\sigma_r} e_r(b)$$

- der relative Fehler wird also mit dem Quotienten aus dem größten und kleinsten Singulärwert (ungleich 0) verstärkt

- die Aussagen der letzten Beispiele lassen sich verallgemeinern
- dabei genügt es, die Pseudoinverse zu betrachten, da die Inverse regulärer Matrizen ja nur ein Spezialfall davon ist

Satz 220. Es sei $A \in \mathbb{R}^{m \times n}$ mit Singulärwertzerlegung $A = U\Sigma V^T$,

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}.$$

Für $x = A^+b$, $\Delta x = A^+\Delta b$ gilt die Abschätzung

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \frac{\sigma_1}{\sigma_r} \frac{\|\Delta b\|_2}{\|P_r U^T b\|_2},$$

mit $P_r = \text{diag}(\underbrace{1, \dots, 1}_r, 0, \dots, 0) \in \mathbb{R}^{m \times m}$. Es gibt immer ein b und Δb , so dass

$$\frac{\|\Delta x\|_2}{\|x\|_2} = \frac{\sigma_1}{\sigma_r} \frac{\|\Delta b\|_2}{\|b\|_2}.$$

Beweis.

- da U, V orthonormale Matrizen sind erhalten wir

$$\begin{aligned} \|\Delta x\|_2 &= \|A^+\Delta b\|_2 \\ &= \|V\Sigma^+U^T\Delta b\|_2 \\ &= \|\Sigma^+U^T\Delta b\|_2 \\ &\leq \|\Sigma^+\|_2 \|U^T\Delta b\|_2 \\ &\leq \|\Sigma^+\|_2 \|\Delta b\|_2 \end{aligned}$$

- da $\Sigma^+ = \text{diag}(\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0)$ ist gilt für die Matrixnorm

$$\|\Sigma^+\|_2 = \sqrt{\rho((\Sigma^+)^T \Sigma^+)} = \frac{1}{\sigma_r}$$

und somit

$$\|\Delta x\|_2 \leq \frac{1}{\sigma_r} \|\Delta b\|_2 \tag{9.4}$$

- andererseits gilt für x

$$\|x\|_2^2 = \|A^+b\|_2^2 = \|V\Sigma^+U^Tb\|_2^2 = \|\Sigma^+ \underbrace{U^Tb}_{=y}\|_2^2 = \sum_{i=1}^r \left(\frac{y_i}{\sigma_i}\right)^2$$

- σ_1 ist der größte Singulärwert, also folgt

$$\|x\|_2^2 \geq \sum_{i=1}^r \left(\frac{y_i}{\sigma_1}\right)^2 = \frac{1}{\sigma_1^2} \sum_{i=1}^r y_i^2 = \frac{1}{\sigma_1^2} \|P_r y\|_2^2 = \left(\frac{\|P_r U^T b\|_2}{\sigma_1}\right)^2 \quad (9.5)$$

- aus (9.4) und (9.5) erhalten wir schließlich

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \frac{\sigma_1}{\sigma_r} \frac{\|\Delta b\|_2}{\|P_r U^T b\|_2},$$

- sei jetzt $c = Ue_i$, $i \in \{1, \dots, r\}$, e_i i -ter Einheitsvektor, dann ist

$$\|c\|_2 = \|Ue_i\|_2 = \|e_i\|_2 = 1$$

und

$$\|A^+ c\|_2 = \|V\Sigma^+ U^T U e_i\|_2 = \|\Sigma^+ e_i\|_2 = \frac{1}{\sigma_i} = \frac{\|c\|_2}{\sigma_i}$$

- für $b = Ue_1$, $\Delta b = Ue_r$ folgt damit

$$\|x\|_2 = \frac{\|b\|_2}{\sigma_1}, \quad \|\Delta x\|_2 = \frac{\|\Delta b\|_2}{\sigma_r}$$

und

$$\frac{\|\Delta x\|_2}{\|x\|_2} = \frac{\sigma_1}{\sigma_r} \frac{\|\Delta b\|_2}{\|b\|_2}.$$

□

- dieses Ergebnis motiviert die folgende Definition der Konditionszahl für beliebige Matrizen

Definition 221. Es sei $A \in \mathbb{R}^{m \times n}$ mit Singulärwertzerlegung $A = U\Sigma V^T$,

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}, \quad \sigma_1 \geq \dots \geq \sigma_r > 0.$$

Dann ist die Konditionszahl κ_2 von A gegeben durch

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_r}$$

Bemerkung 222. Für reguläre Matrizen A stimmt dieser Ausdruck mit der früheren Definition der **Konditionszahl** κ_2 überein.

- in den nächsten Abschnitten werden wir nun Verfahren untersuchen, mit denen man brauchbare Näherungslösungen für schlecht konditionierte Gleichungssysteme bzw. lineare Ausgleichsprobleme erzeugen kann

9.6.2 Abgeschnittene Singulärwertzerlegung (TSVD)

- wie wir oben gesehen haben, werden die Konditionsprobleme durch die Singulärwerte σ_i von A verursacht, für die

$$\frac{\sigma_1}{\sigma_i}$$

groß ist

- lassen wir diese σ_i einfach weg, dann erhalten wir die abgeschnittene Singulärwertzerlegung (**Truncated Singular Value Decomposition, TSVD**)

Definition 223 (Abgeschnittene Singulärwertzerlegung). Es sei $A \in \mathbb{R}^{m \times n}$ mit Singulärwertzerlegung $A = U\Sigma V^T$,

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}, \quad \sigma_1 \geq \dots \geq \sigma_r > 0.$$

Für $\alpha > 0$ ist $A_\alpha = U\Sigma_\alpha V^T$,

$$\Sigma_\alpha = \text{diag}(\sigma_1, \dots, \sigma_k, 0, \dots, 0) \in \mathbb{R}^{m \times n}, \quad \frac{\sigma_1}{\sigma_k} \leq \frac{1}{\alpha}, \quad \frac{\sigma_1}{\sigma_{k+1}} \geq \frac{1}{\alpha},$$

eine abgeschnittene Singulärwertzerlegung von A

- aus der Definition folgt direkt, dass

$$\varkappa_2(A_\alpha) \leq \frac{1}{\alpha}$$

- für großes α gilt:

- A_α hat gute Kondition und

$$x_\alpha = A_\alpha^+ b = V\Sigma_\alpha^+ U^T b$$

ist problemlos zu berechnen

- A_α, A_α^+ sind unter Umständen keine guten Approximationen von A, A^+ , so dass x_α mit der gesuchten Lösung wenig zu tun hat

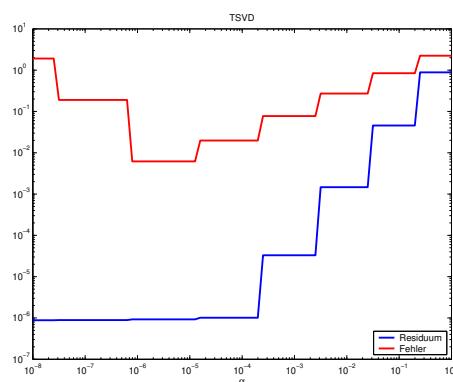
- für kleines α gilt:

- A_α, A_α^+ sind gute Approximationen von A, A^+
- A_α hat schlechte Kondition

- es kommt also wesentlich auf die Wahl des Parameters α an, wie wir auch im folgenden Beispiel sehen

Beispiel 224.

- wir betrachten noch einmal die Aufgabenstellung von Beispiel 217
- TSVD liefert folgende Ergebnisse



- bei $\alpha = 1$ bleibt nach dem Abschneiden nur ein einziger Singulärwert übrig
- mit fallendem α werden immer kleinere Singulärwerte mit berücksichtigt
- ab einem gewissen Punkt steigt der Fehler wieder an, da jetzt sehr kleine Singulärwerte die Störung der rechten Seite extrem verstärken
- das Residuum fällt monoton, ist also als Qualitätsindikator für die erzielte Näherung zunächst unbrauchbar

- das Verhalten des Residuums im letzten Beispiel ist kein Zufall

Satz 225. Sei $x_\alpha = A_\alpha^+ b$. Für $\alpha \rightarrow 0$ konvergiert x_α gegen $x = A^+ b$ und das Residuum $\|b - Ax_\alpha\|_2$ ist monoton nicht wachsend.

Beweis.

- die Konvergenz von x_α gegen x ist offensichtlich, da A_α^+ gegen A^+ geht für $\alpha \rightarrow 0$
- da U, V orthonormale Matrizen sind erhalten wir für das Residuum

$$\begin{aligned}\|b - Ax_\alpha\|_2 &= \|b - AA_\alpha^+ b\|_2 \\ &= \|b - U\Sigma V^T V\Sigma_\alpha^+ U^T b\|_2 \\ &= \|(I - U\Sigma\Sigma_\alpha^+ U^T) b\|_2 \\ &= \|U(I - P_{k_\alpha}) U^T b\|_2 \\ &= \|(I - P_{k_\alpha}) \underbrace{U^T b}_{=y}\|_2 \\ &= \sqrt{\sum_{i=k_\alpha+1}^r y_i^2}\end{aligned}$$

wobei P_{k_α} wie in Satz 220 definiert ist,

- für $\alpha \rightarrow 0$ wird k_α nie kleiner und konvergiert gegen r , so dass $\sum_{i=k_\alpha+1}^r y_i^2$ bei fallendem α nicht wachsen kann

□

Bemerkung 226.

- der aufgrund der schlechten Kondition für sehr kleines α explodierende Fehler kann im Residuum also nicht entdeckt werden
- bei der Berechnung der Pseudoinversen in MATLAB (`pinv`) wird immer eine TSVD durchgeführt, wobei α in der Größenordnung der Maschinengenauigkeit liegt

9.6.3 Tikhonov Regularisierung

- die Berechnung der Pseudoinversen $x = A^+ b$ ist äquivalent zum Minimierungsproblem

$$\|Ax - b\|_2 \rightarrow \min, \quad \|x\|_2 \rightarrow \min$$

- wir betrachten jetzt für $\alpha > 0$ die Funktion

$$J_\alpha(x) = \|Ax - b\|_2^2 + \alpha^2 \|x\|_2^2$$

und minimieren sie über x

- dazu formen wir J_α zunächst um

$$\begin{aligned}
J_\alpha(x) &= \|Ax - b\|_2^2 + \alpha^2\|x\|_2^2 \\
&= \left\| \begin{pmatrix} Ax \\ \alpha x \end{pmatrix} - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2 \\
&= \left\| \underbrace{\begin{pmatrix} A \\ \alpha I \end{pmatrix}}_{A_\alpha} x - \underbrace{\begin{pmatrix} b \\ 0 \end{pmatrix}}_{b_\alpha} \right\|_2^2 \\
&= \|A_\alpha x - b_\alpha\|_2^2
\end{aligned}$$

- wegen $\alpha > 0$ hat A_α vollen Rang, so dass das Minimierungsproblem eindeutig lösbar ist und der Minimierer x_α über die Normalgleichungen

$$A_\alpha^T A_\alpha x_\alpha = A_\alpha^T b_\alpha$$

bestimmt werden kann

- setzen wir A_α, b_α ein, so erhalten wir

$$(A^T, \alpha I) \begin{pmatrix} A \\ \alpha I \end{pmatrix} x_\alpha = (A^T, \alpha I) b_\alpha$$

bzw.

$$(A^T A + \alpha^2 I) x_\alpha = A^T b$$

- damit haben wir also die Normalgleichungen für $\|Ax - b\|_2 \rightarrow \min$ erhalten, ergänzt um einen Zusatzterm $\alpha^2 I$, der dafür sorgt, dass die Systemmatrix $A^T A + \alpha^2 I$ symmetrisch positiv definit ist

Definition 227 (Tikhonov-Regularisierung). Für $A \in \mathbb{R}^{m \times n}$, $\alpha > 0$ ist

$$x_\alpha = (A^T A + \alpha^2 I)^{-1} A^T b$$

die Tikhonov-Regularisierung von $x = A^+ b$ zum Parameter α .

Satz 228. Sei x_α die Tikhonov-Regularisierung von $x = A^+ b$. Für $\alpha \rightarrow 0$ konvergiert x_α gegen x und das Residuum $\|b - Ax_\alpha\|_2$ ist monoton nicht wachsend.

Beweis.

- für x_α gilt

$$x_\alpha = (A^T A + \alpha^2 I)^{-1} A^T b$$

- ist $A = U\Sigma V^T$ eine Singulärwertzerlegung von A , so folgt

$$A^T A + \alpha^2 I = V \Sigma^T U^T U \Sigma V^T + \alpha^2 I = V (\Sigma^T \Sigma + \alpha^2 I) V^T = V S_\alpha V^T$$

mit

$$S_\alpha = \Sigma^T \Sigma + \alpha^2 I = \text{diag}(\sigma_1^2 + \alpha^2, \dots, \sigma_r^2 + \alpha^2, \alpha^2, \dots, \alpha^2) \in \mathbb{R}^{n \times n}$$

- damit ist

$$x_\alpha = (A^T A + \alpha^2 I)^{-1} A^T b = V S_\alpha^{-1} V^T V \Sigma^T U^T b = V S_\alpha^{-1} \Sigma^T U^T b$$

- wegen

$$S_\alpha^{-1} \Sigma^T = \text{diag}\left(\frac{\sigma_1}{\sigma_1^2 + \alpha^2}, \dots, \frac{\sigma_r}{\sigma_r^2 + \alpha^2}, 0, \dots, 0\right) \in \mathbb{R}^{n \times m}$$

folgt

$$S_\alpha^{-1} \Sigma^T \xrightarrow{\alpha \rightarrow 0} \text{diag}\left(\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0\right) = \Sigma^+$$

und somit

$$x_\alpha = V(S_\alpha^{-1} \Sigma^T) U^T b \xrightarrow{\alpha \rightarrow 0} V \Sigma^+ U^T b = A^+ b$$

- für das Residuum erhalten wir

$$\begin{aligned} \|b - Ax_\alpha\|_2 &= \|b - U \Sigma V^T V S_\alpha^{-1} \Sigma^T U^T b\|_2 \\ &= \|b - U \Sigma S_\alpha^{-1} \Sigma^T U^T b\|_2 \\ &= \|U(I - \Sigma S_\alpha^{-1} \Sigma^T) U^T b\|_2 \\ &= \|(I - \Sigma S_\alpha^{-1} \Sigma^T) \underbrace{U^T b}_{=y}\|_2 \end{aligned}$$

- wegen

$$I - \Sigma S_\alpha^{-1} \Sigma^T = \text{diag}\left(1 - \frac{\sigma_1^2}{\sigma_1^2 + \alpha^2}, \dots, 1 - \frac{\sigma_r^2}{\sigma_r^2 + \alpha^2}, 1, \dots, 1\right) \in \mathbb{R}^{m \times m}$$

folgt

$$\|b - Ax_\alpha\|_2^2 = \sum_{i=1}^r \left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2}\right) y_i^2 + \sum_{i=r+1}^n y_i^2$$

- da $1 - \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2}$, $i = 1, \dots, r$, monoton fallend für $\alpha \rightarrow 0$ ist, folgt die Behauptung

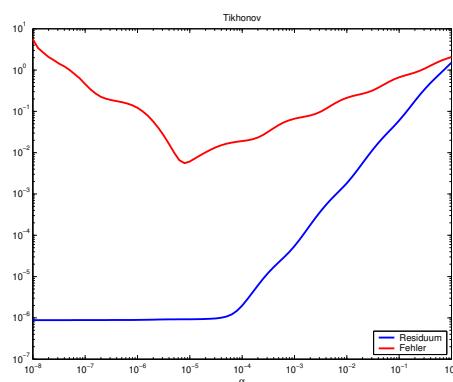
□

- damit haben wir ein ähnliches Verhalten wie bei TSVD

- zu großes α liefert schlechte Approximationsqualität
- zu kleines α stabilisiert zu wenig

Beispiel 229.

- wir wenden auf die Aufgabenstellung von Beispiel 224 die Tikhonov-Regularisierung an



- ein vernünftige Parameterwahl ist entscheidend

9.6.4 Parameterwahl

- es gibt zahlreiche Strategien, um den Regularisierungsparameter α zu wählen
- die bekannteste davon arbeitet nach folgender Idee:
 - sei b die ungestörte rechte Seite
 - b ist nicht bekannt, sondern nur die gestörte Version \tilde{b}
 - \tilde{b} sei mit einer absoluten Genauigkeit δ gegeben, d.h.

$$\|\tilde{b} - b\|_2 = \|\Delta b\|_2 \leq \delta,$$

\tilde{x}_α sei eine Regularisierung von $A^+ \tilde{b}$

- wähle das größte α , so dass

$$\|\tilde{b} - A\tilde{x}_\alpha\|_2$$

in der Größenordnung von δ liegt

- man soll also das Residuum nicht unnötig klein machen, da sonst die Gefahr besteht, dass man wieder in den Bereich schlechter Kondition gerät
- das gewählte α hängt von δ und \tilde{b} ab

Definition 230 (Diskrepanz-Prinzip nach Morozov). Seien A, \tilde{b}, δ gegeben, $\|\tilde{b} - b\|_2 \leq \delta$ und \tilde{x}_α eine Regularisierung von $A^+ \tilde{b}$. Dann wählt man den Regularisierungsparameter gemäß

$$\alpha(\delta, \tilde{b}) = \sup_{\alpha > 0} (\|\tilde{b} - A\tilde{x}_\alpha\|_2 \leq \tau\delta),$$

mit $\tau \geq 1$ fest.

- wir wenden jetzt das Diskrepanz-Prinzip auf die Tikhonov-Regularisierung an

Beispiel 231.

- wir betrachten wieder unser Ausgangsproblem aus Beispiel 217
- dort war $\delta = 10^{-6}$
- wir benutzen die Tikhonov-Regularisierung und wählen α nach dem Diskrepanz-Prinzip mit $\tau = 1.1$
- dazu versuchen wir die nichtlineare Gleichung

$$\|\tilde{r}_\alpha\|_2 = \|\tilde{b} - A\tilde{x}_\alpha\|_2 = \tau\delta$$

nach α aufzulösen

- nach Satz 9.6.3 ist $\|\tilde{r}_\alpha\|_2$ monoton in α , so dass wir dazu z.B. einfach das Bisektionsverfahren benutzen können
- als Abbruchkriterium benutzen wir

$$|\|\tilde{r}_\alpha\|_2 - \tau\delta| \leq 0.1 \cdot \delta$$

- damit erhalten wir $\alpha \approx 6.1 \cdot 10^{-5}$ bei einem Residuum von etwa $1.19 \cdot 10^{-6}$ und einem relativen Fehler $e_r(x) \approx 0.0045$
- die zugehörige Näherung x_α ist in der folgenden Tabelle angegeben

exakt ungestört	LU, Spalten-Pivot	Householder	Tikhonov
1	1.0104e1	3.5021e2	1.0002
1	-8.9708e2	-4.4339e4	0.9973
1	2.2261e4	1.3469e6	1.0072
1	-2.4066e5	-1.6494e7	0.9997
1	1.4028e6	9.1563e7	0.9949
1	-4.8893e6	-1.4746e8	0.9947
1	1.0733e7	-9.6850e8	0.9973
1	-1.5127e7	6.7281e9	1.0007
1	1.3561e7	-2.0302e10	1.0037
1	-7.3813e6	3.6709e10	1.0054
1	2.1458e6	-4.2460e10	1.0055
1	-1.7897e5	3.1367e10	1.0041
1	-5.7412e4	-1.4057e10	1.0011
1	4.7274e3	3.3515e9	0.9967
1	6.5344e3	-2.9622e8	0.9911

Kapitel 10

Numerische Integration

10.1 Einleitung

- wir betrachten folgende Aufgabenstellung: gegeben f stetig, berechne

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

für endliche a, b

- aus der Analysis ist folgendes bekannt:

- es ist $\int_a^b f(x)dx = F(b) - F(a)$, wobei F die Stammfunktion von f ist
- ist f stetig, so existiert F immer
- F ist oft nicht in geschlossener Form darstellbar, z.B. für $f(x) = e^{-x^2}$
- deswegen braucht man Methoden um $\int_a^b f(x)dx$ numerisch zu nähern
- Hauptanwendungsgebiete sind Methoden zur Approximation von Differentialgleichungen (Algorithmen für gewöhnliche Differentialgleichungen, Finite Elemente)
- die Grundidee für numerische Integrationsverfahren ist:
 - nähre f auf $[a, b]$ durch eine Funktion g , die einfach integrierbar ist (d.h. die Stammfunktion von g ist einfach zu bestimmen)
 - benutze dann $\int_a^b f(x)dx \approx \int_a^b g(x)dx$

10.2 Newton-Cotes Quadraturformeln

- wir wollen für $f : [a, b] \rightarrow \mathbb{R}$ stetig $\int_a^b f(x)dx$ approximieren
- Idee:

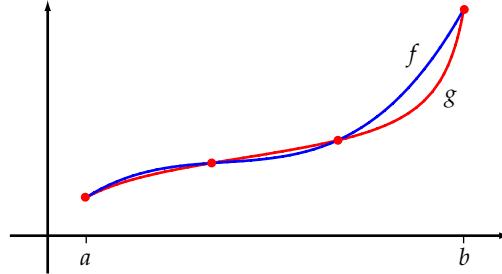
- “verteile” $n + 1$ Punkte $x_k, k = 0, \dots, n$ gleichmäßig in $[a, b]$, d.h.

$$x_k = a + kh, \quad h = \frac{b - a}{n}$$

- berechne

$$y_k = f(x_k), \quad k = 0, \dots, n$$

- interpoliere ein Polynom vom Grad n durch die Punkte (x_k, y_k)



- benutze das Interpolationspolynom als Approximation g von f auf $[a, b]$

- berechne $\int_a^b g(x) dx$ als Approximation von $\int_a^b f(x) dx$

- wegen

$$g(x) = g_0 + g_1 x + \dots + g_n x^n, \quad g_i \in \mathbb{R}$$

ist die Stammfunktion G einfach berechenbar

$$G(x) = g_0 x + g_1 \frac{x^2}{2} + \dots + g_n \frac{x^{n+1}}{n+1}$$

- um die Vorgehensweise etwas systematischer darzustellen benutzen wir die Lagrange-Darstellung des Interpolationspolynoms

$$g(x) = \sum_{k=0}^n f(x_k) L_k(x)$$

und erhalten

$$I_n(f) := \int_a^b g(x) dx = \sum_{k=0}^n f(x_k) \int_a^b L_k(x) dx = \sum_{k=0}^n \beta_k f(x_k)$$

mit $\beta_k = \int_a^b L_k(x) dx$

- die Integralapproximation $I_n(f)$ besteht also aus einer Linearkombination der Funktionswerte des Integranden f an den **Stützstellen** x_k mit geeigneten **Gewichten** β_k
- nach Abschnitt 8.2 gilt

$$L_k(x) = \frac{x - x_0}{x_k - x_0} \cdot \dots \cdot \frac{x - x_{k-1}}{x_k - x_{k-1}} \cdot \frac{x - x_{k+1}}{x_k - x_{k+1}} \cdot \dots \cdot \frac{x - x_n}{x_k - x_n}$$

- außerdem ist $a = x_0, b = x_n$ so dass

$$\beta_k = \int_a^b L_k(x) dx$$

nur von den $x_i, i = 0, \dots, n$, abhängt

- die Darstellung der β_k kann weiter vereinfacht werden:

- wir benutzen in $\beta_k = \int_a^b L_k(x) dx$ die Substitution

$$x \mapsto \tilde{x} := \frac{x-a}{h}, \quad h = \frac{b-a}{n}$$

d.h. $[a, b] \rightarrow [0, n]$

- damit folgt

$$\beta_k = h \int_0^n \underbrace{L_k(a + h\tilde{x})}_{=: \tilde{L}_k(\tilde{x})} d\tilde{x}$$

- wegen $\tilde{L}_k(\tilde{x}) = L_k(a + h\tilde{x})$ ist \tilde{L}_k ein Polynom vom Grad n in \tilde{x}
- für $\tilde{x}_i = i$, $i = 0, \dots, n$, folgt

$$\tilde{L}_k(\tilde{x}_i) = \tilde{L}_k(i) = L_k(a + ih) = L_k(x_i) = \delta_{ki},$$

d.h. \tilde{L}_k ist das k -te Lagrange Polynom zu den Stützwerten $\tilde{x}_i = i$ und damit

$$\begin{aligned} \tilde{L}_k(\tilde{x}) &= \frac{\tilde{x} - 0}{k - 0} \cdot \dots \cdot \frac{\tilde{x} - (k-1)}{k - (k-1)} \cdot \frac{\tilde{x} - (k+1)}{k - (k+1)} \cdot \dots \cdot \frac{\tilde{x} - n}{k - n} \\ &= \prod_{\substack{i=0 \\ i \neq k}}^n \frac{\tilde{x} - i}{k - i} \end{aligned}$$

- damit erhalten wir

$$\beta_k = h \int_0^n \tilde{L}_k(\tilde{x}) d\tilde{x} = (b-a) \underbrace{\frac{1}{n} \int_0^n \prod_{\substack{i=0 \\ i \neq k}}^n \frac{\tilde{x} - i}{k - i} d\tilde{x}}_{=: \alpha_k}$$

- die α_k hängen ausschließlich von n ab, nicht von a, b, f oder den x_k
- bei gegebenem n werden immer dieselben α_k benutzt, die vorab berechnet und tabelliert werden
- insgesamt erhalten wir folgendes Resultat:

Satz 232. Die Integralapproximation $I_n(f)$ von $\int_a^b f(x) dx$ mit Hilfe des Interpolationspolynoms vom Grad n durch die Punkte

$$(x_k, f(x_k)), \quad x_k = a + kh, \quad h = \frac{b-a}{n}, \quad k = 0, \dots, n$$

ist gegeben durch

$$I_n(f) = (b-a) \sum_{k=0}^n \alpha_k f(x_k)$$

mit **Gewichten**

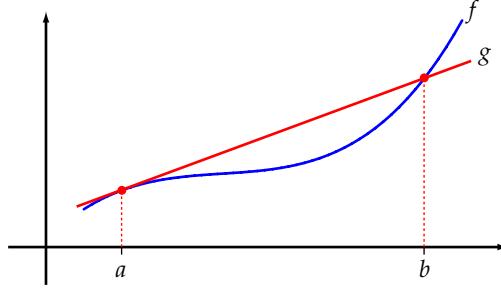
$$\alpha_k = \frac{1}{n} \int_0^n \prod_{\substack{i=0 \\ i \neq k}}^n \frac{x - i}{k - i} dx,$$

die nur von n abhängen.

Beispiel 233.

► $n = 1$:

- in diesem Fall erhalten wir zwei Stützstellen $x_0 = a$, $x_1 = b$ und Polynomgrad 1, d.h. g ist eine Gerade durch die Funktionswerte an den Intervallendpunkten



- $\int_a^b f(x) dx$ wird genähert durch $\int_a^b g(x) dx$, also durch die Trapezfläche unterhalb der Geraden, weshalb diese Integralnäherung auch **Trapez-Regel** heißt
- es ist

$$\int_a^b g(x) dx = (b-a) \frac{1}{2} (f(a) + f(b)) = (b-a) \left(\frac{1}{2} f(x_0) + \frac{1}{2} f(x_1) \right)$$

und somit $\alpha_0 = \alpha_1 = \frac{1}{2}$

- berechnen wir die α_k nach dem letztem Satz, so erhalten wir das gleiche Ergebnis

$$\alpha_0 = \frac{1}{1} \cdot \int_0^1 \frac{x-1}{0-1} dx = \int_0^1 1-x dx = \left[x - \frac{x^2}{2} \right]_0^1 = \frac{1}{2}$$

und $\alpha_1 = \frac{1}{2}$ analog

► $n = 2$:

- für Polynomgrad 2 benutzen wir eine weitere Stützstelle in der Intervallmitte und erhalten die **Simpson-Regel**

$$I_2(f) = (b-a) \left(\frac{1}{6} f(a) + \frac{4}{6} f\left(\frac{a+b}{2}\right) + \frac{1}{6} f(b) \right),$$

d.h.

i	0	1	2
x_i	a	$\frac{a+b}{2}$	b
α_i	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$

► $n = 3$:

- für Polynomgrad 3 benutzen wir 4 Stützstellen und erhalten die $\frac{3}{8}$ -Regel, die auch **Newton-Integration** bzw. **Fass-Regel** genannt wird:

i	0	1	2	3
x_i	a	$\frac{2a+b}{3}$	$\frac{a+2b}{3}$	b
α_i	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

Bemerkung 234. Für die interpolatorische Integralnäherung $I_n(f)$ gilt immer $\sum_{k=0}^n \alpha_k = 1$, so dass $f(x) = 1$ exakt integriert wird

Beispiel 235.

- wir approximieren

$$\int_1^2 \frac{1}{x} dx = \ln(2) \approx 0.69314718056$$

mit den drei Verfahren von oben

- mit $b - a = 1$, $f(x) = \frac{1}{x}$ erhalten wir:

- Trapez-Regel:

$$I_1(f) = (b - a) \left(\frac{1}{2} f(a) + \frac{1}{2} f(b) \right) = \frac{1}{2} \left(\frac{1}{1} + \frac{1}{2} \right) = \frac{3}{4}$$

- Simpson-Regel:

$$I_2(f) = (b - a) \frac{1}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) = \frac{1}{6} \left(\frac{1}{1} + 4 \cdot \frac{1}{\frac{3}{2}} + \frac{1}{2} \right) = \frac{1}{6} \left(1 + \frac{8}{3} + \frac{1}{2} \right) = \frac{25}{36}$$

- $\frac{3}{8}$ -Regel:

$$I_3(f) = \frac{1}{8} \left(f(1) + 3f\left(\frac{4}{3}\right) + 3f\left(\frac{5}{3}\right) + f(2) \right) = \frac{111}{160}$$

- ein Vergleich mit dem exakten Wert liefert

	$\int_1^2 f(x) dx \approx$	relativer Fehler \approx
exakt	0.69314718056	—
Trapez	0.75	0.082
Simpson	0.694	0.0019
$\frac{3}{8}$ -Regel	0.69375	0.00087

- wir betrachten jetzt den Fehler von I_n genauer:

$$E_n(f) := \int_a^b f(x) dx - I_n(f)$$

- ist f ein Polynom vom Grad $\leq n$, so stimmt das Interpolationspolynom mit f überein und $E_n(f) = 0$
- sei $f \in C^2[a, b]$, $n = 1$ (d.h. I_n ist die Trapez-Regel) und ohne Beschränkung der Allgemeinheit $a = 0$ (also $h = \frac{b-a}{n} = b$), dann ist

$$I_1(f) = \frac{h}{2} (f(0) + f(h))$$

bzw.

$$E_1(f) = \int_0^h f(x) dx - \frac{h}{2} (f(0) + f(h))$$

- partielle Integration liefert

$$\int_0^h f(x) dx = [xf(x)]_0^h - \int_0^h xf'(x) dx = hf(h) - \int_0^h xf'(x) dx$$

und somit

$$\begin{aligned} E_1(f) &= hf(h) - \int_0^h xf'(x) dx - \frac{h}{2}(f(0) + f(h)) \\ &= \frac{h}{2}(f(h) - f(0)) - \int_0^h xf'(x) dx \\ &= \frac{h}{2} \int_0^h f'(x) dx - \int_0^h xf'(x) dx \\ &= \int_0^h \left(\frac{h}{2} - x\right) f'(x) dx \end{aligned}$$

- eine weitere partielle Integration liefert

$$E_1(f) = \left[\frac{x}{2}(h-x)f'(x)\right]_0^h - \int_0^h \frac{x}{2}(h-x)f''(x) dx = \int_0^h \frac{x(x-h)}{2}f''(x) dx$$

- mit dem Mittelwertsatz der Integralrechnung folgt

$$E_1(f) = f''(\xi) \int_0^h \frac{x(x-h)}{2} dx = -\frac{1}{12}h^3 f''(\xi), \quad \xi \in [0, h],$$

weshalb der Fehler für die Trapez-Regel für $h \rightarrow 0$ wie h^3 fällt

- analog zeigt man für beliebiges n

Satz 236. Sei $f \in C^{2+n}[a, b]$, $h = \frac{b-a}{n}$, $I_n(f)$ wie oben. Dann gibt es ein $\xi \in [a, b]$ mit

$$E_n(f) = \begin{cases} \int_0^n \frac{t \prod_{k=0}^n (t-k)}{(n+2)!} dt & h^{n+3} f^{(n+2)}(\xi) \quad \text{für } n \text{ gerade} \\ \int_0^n \frac{\prod_{k=0}^n (t-k)}{(n+1)!} dt & h^{n+2} f^{(n+1)}(\xi) \quad \text{für } n \text{ ungerade} \end{cases}$$

Bemerkung 237. $I_n(f)$ zusammen mit der Fehlerdarstellung für $E_n(f)$ heißen (geschlossene) **Newton-Cotes-Formeln**

Beispiel 238.

- rechnet man die Fehlerdarstellung für die oben besprochenen Verfahren aus, so erhält man

Verfahren	n	h	$E_n(f)$
Trapez	1	$b-a$	$-\frac{1}{12}h^3 f''(\xi)$
Simpson	2	$\frac{b-a}{2}$	$-\frac{1}{90}h^5 f^{(IV)}(\xi)$
$\frac{3}{8}$ -Regel	3	$\frac{b-a}{3}$	$-\frac{3}{80}h^5 f^{(IV)}(\xi)$

- führt man damit eine Fehlerabschätzung für $\int_1^2 \frac{1}{x} dx$ durch so folgt

► Trapez-Regel:

- nach der Fehlerformel gilt $|E_1(f)| = \frac{h^3}{12} |f''(\xi)|$ mit $\xi \in [1, 2]$ und deshalb

$$|E_1(f)| \leq \frac{h^3}{12} \max_{\xi \in [1, 2]} |f''(\xi)|$$

- wegen $f(x) = \frac{1}{x}$ ist $f''(x) = \frac{2}{x^3}$ und $\max_{\xi \in [1, 2]} |f''(\xi)| = 2$
- mit $h = b - a = 1$ erhalten wir insgesamt

$$|E_1(f)| \leq \frac{1}{12} \cdot 2 = \frac{1}{6}$$

► Simpson-Regel:

- analog zur Trapez-Regel benutzen wir

$$|E_2(f)| \leq \frac{h^5}{90} \max_{\xi} |f^{(IV)}(\xi)|, \quad h = \frac{b-a}{2} = \frac{1}{2}, \quad f^{(IV)}(x) = \frac{24}{x^5}$$

und erhalten

$$|E_2(f)| \leq \frac{1}{90} \left(\frac{1}{2}\right)^5 24 = \frac{1}{120}$$

► $\frac{3}{8}$ -Regel:

- da $h = \frac{b-a}{3} = \frac{1}{3}$ erhalten wir eine kleinere obere Schranke als bei der Simpson-Regel

$$|E_3(f)| \leq \frac{3}{80} \left(\frac{1}{3}\right)^5 24 = \frac{1}{270},$$

- der Fehler in den Newton-Cotes-Formeln (siehe oben) wird durch $h = \frac{b-a}{n}$ und n bestimmt
- für kleine Fehler braucht man große n , d.h. einen hohen Polynomgrad
- dabei entstehen durch die Instabilitäten des benutzten Interpolationspolynoms g bei hohem Grad Probleme, d.h. g approximiert den Integranden f schlecht und I_n ist eine schlechte Integralnäherung
- typisches Anzeichen sind negative Gewichte α_k in den Integrationsformeln, die ab $n = 8$ auftreten
- dieses Problem wird analog zu den Splines gelöst
 - halte Polynomgrad niedrig
 - setze mehrere Polynome zusammen
- konkret heißt das:

- zerlege $[a, b]$ in m Teilintervalle $[a_i, b_i]$ mit

$$a = a_0 < b_0 = a_1 < b_1 \dots < b_{m-1} =: a_m = b$$

- zerlege das Integral in eine Summe von Integralen auf den Teilintervallen

$$\int_a^b f(x) dx = \sum_{i=0}^{m-1} \int_{a_i}^{b_i} f(x) dx$$

- wende für jedes Teilintervall $[a_i, b_i]$ eine Newton-Cotes Formel zur Approximation von $\int_{a_i}^{b_i} f(x) dx$ an
- damit erhalten wir die **summierten Newton-Cotes-Formeln** $I_{m,n}(f)$

$$\int_a^b f(x) dx \approx I_{m,n}(f) = \sum_{i=0}^{m-1} (b_i - a_i) \sum_{k=0}^n \alpha_k f(a_i + kh_i)$$

$$\text{mit } h_i = \frac{b_i - a_i}{n}$$

- damit wird f durch ein stückweises Polynom g ersetzt, das dann integriert wird
- im Gegensatz zu Splines braucht g an den Nahtstellen nicht glatt zu sein, da Stetigkeit des Integranden völlig ausreicht
- im äquidistanten Fall

$$l := b_i - a_i = \frac{b - a}{m}, \quad h := h_i = \frac{b_i - a_i}{n} = \frac{l}{n}, \quad a_i = a + i \cdot l, \quad i = 0, \dots, m$$

erhält man eine besonders einfache Darstellungen für $I_{m,n}$

- für die äquidistante **summierte Trapez-Regel** gilt somit $n = 1, h = l = \frac{b-a}{m}$ und

$$\begin{aligned} T_m(f) &= I_{m1}(f) = \sum_{i=0}^{m-1} l \frac{1}{2} (f(a_i) + f(b_i)) \\ &= \frac{l}{2} \sum_{i=0}^{m-1} (f(a_i) + f(a_{i+1})) \\ &= \frac{l}{2} (f(a_0) + f(a_1) \\ &\quad + f(a_1) + f(a_2) \\ &\quad \vdots \\ &\quad + f(a_{m-1}) + f(a_m)) \end{aligned}$$

also

$$T_m(f) = \frac{l}{2} (f(a_0) + 2f(a_1) + \dots + 2f(a_{m-1}) + f(a_m))$$

$$\text{mit } l = \frac{b-a}{m}, \quad a_i = a + i l$$

Beispiel 239.

- wir wenden die äquidistante summierte Trapez-Regel $T_4(f)$ auf $\int_{-4}^4 f(x) dx$ mit $f(x) = \frac{1}{1+x^2}$ an

► der exakte Wert ist $2 \arctan(4) \approx 2.652$

- es ist $m = 4$, $l = \frac{b-a}{m} = \frac{4-(-4)}{4} = 2$, $a_0 = -4$, $a_1 = -2$, $a_2 = 0$, $a_3 = 2$, $a_4 = 4$ und

$$\begin{aligned} T_4(f) &= \frac{l}{2} (f(a_0) + 2f(a_1) + 2f(a_2) + 2f(a_3) + f(a_4)) \\ &= 1 \cdot \left(\frac{1}{1+16} + 2 \frac{1}{1+4} + 2 \frac{1}{1+0} + 2 \frac{1}{1+4} + \frac{1}{1+16} \right) \\ &= \frac{248}{85} \\ &\approx 2.918 \end{aligned}$$

mit relativem Fehler $e_r \approx 10\%$

- zur Herleitung der äquidistanten summierten Simpson-Regel gehen wir analog vor
- durch einfaches Einsetzen erhalten wir

$$I_{m,2}(f) = l \sum_{i=0}^{m-1} \frac{1}{6} \left(f(a_i) + 4f\left(\frac{a_i + a_{i+1}}{2}\right) + f(a_{i+1}) \right)$$

mit $a_i = a + i l$, $l = \frac{b-a}{m}$

- wir setzen nun

$$\tilde{m} = 2m, \quad \tilde{l} = \frac{b-a}{\tilde{m}} = \frac{l}{2}, \quad \tilde{a}_i = a + i \tilde{l}$$

so dass

$$\tilde{a}_{2i} = a_i, \quad \tilde{a}_{2i+1} = \frac{a_i + a_{i+1}}{2}$$

- die **Streifenzahl** \tilde{m} ist immer gerade, $\tilde{m} + 1$ ist die Anzahl der Funktionsauswertungen
- für die äquidistante **summierte Simpson-Regel** $S_{\tilde{m}}$ erhalten wir

$$\begin{aligned} S_{\tilde{m}}(f) &= I_{m,2}(f) \\ &= \frac{\tilde{l}}{6} \left(f(\tilde{a}_0) + 4f(\tilde{a}_1) + f(\tilde{a}_2) \right. \\ &\quad \left. + f(\tilde{a}_2) + 4f(\tilde{a}_3) + f(\tilde{a}_4) \right. \\ &\quad \vdots \\ &\quad \left. + f(\tilde{a}_{\tilde{m}-2}) + 4f(\tilde{a}_{\tilde{m}-1}) + f(\tilde{a}_{\tilde{m}}) \right) \end{aligned}$$

und damit

$$S_{\tilde{m}}(f) = \frac{\tilde{l}}{3} \left(f(\tilde{a}_0) + 4f(\tilde{a}_1) + 2f(\tilde{a}_2) + 4f(\tilde{a}_3) + \dots + 2f(\tilde{a}_{\tilde{m}-2}) + 4f(\tilde{a}_{\tilde{m}-1}) + f(\tilde{a}_{\tilde{m}}) \right)$$

mit gerader **Streifenzahl** $\tilde{m} = 2m$ und $\tilde{l} = \frac{b-a}{\tilde{m}}$, $\tilde{a}_i = a + i \tilde{l}$

Beispiel 240.

- wir approximieren das Integral aus Beispiel 239

$$\int_{-4}^4 \frac{1}{1+x^2} dx$$

mit Hilfe von $S_4(f)$

- dann ist $\tilde{m} = 4$, $\tilde{l} = \frac{b-a}{\tilde{m}} = 2$, $\tilde{a}_0 = -4$, $\tilde{a}_1 = -2$, $\tilde{a}_2 = 0$, $\tilde{a}_3 = 2$, $\tilde{a}_4 = 4$ und

$$S_4(f) = \frac{2}{3} \left(\frac{1}{17} + 4 \cdot \frac{1}{5} + 2 \cdot 1 + 4 \cdot \frac{1}{5} + \frac{1}{17} \right) = \frac{632}{255} \approx 2.479$$

mit relativem Fehler $e_r \approx 6.5\%$

- bei gleicher Anzahl von Funktionsauswertungen wie bei der summierten Trapez-Regel in Beispiel 239 erhalten wir eine höhere Genauigkeit

10.3 Gauß-Integration

- wir betrachten jetzt wieder den nicht-summierten Fall, d.h. ein einziges Intervall
- die Herleitung der Newton-Cotes-Verfahren besteht aus folgenden Schritten:
 - gebe äquidistante Stützstellen $x_i = a + ih$, $h = \frac{b-a}{n}$ vor
 - bestimme über die Lagrange-Polynome die Gewichte β_i
 - approximiere $\int_a^b f(x) dx \approx \sum_{i=0}^n \beta_i f(x_i) =: I_n(f)$
- die Fehlerbetrachtung (siehe Satz 236) liefert

$$\int_a^b f(x) dx - I_n(f) \sim \begin{cases} h^{n+3} f^{(n+2)}(\xi) & n \text{ gerade} \\ h^{n+2} f^{(n+1)}(\xi) & n \text{ ungerade} \end{cases}$$

- der Fehler verhält sich also wie h^{n+3} (bzw. h^{n+2})
- für Polynome mit Grad $\leq n+1$ (n) ist der Fehler wegen des Faktors $f^{(n+2)}(\xi)$ ($f^{(n+1)}(\xi)$) gleich 0
- kann man diese beide Aussagen bei gleichem Aufwand noch verbessern?
- Idee:
 - bei Newton-Cotes werden die x_i willkürlich festgelegt
 - wir versuchen jetzt die x_i möglichst günstig in $[a, b]$ zu verteilen
- die folgenden Überlegungen gehen zurück auf Gauß
- wir benutzen sogenannte **Orthogonalpolynome**

- wir definieren ein Skalarprodukt

$$(f, g) = \int_a^b f(x)g(x) dx$$

für Funktionen auf $[a, b]$ und betrachten Polynome $P_k(x)$ vom Grad k mit führendem Koeffizienten 1

$$P_k(x) = p_{k0} + p_{k1}x + \dots + p_{kk-1}x^{k-1} + x^k$$

- Polynome P_i , $i = 0, \dots, m$, dieses Typs heißen orthogonal falls

$$(P_i, P_j) = \int_a^b P_i(x)P_j(x) dx = \gamma_i \delta_{ij}, \quad \gamma_i \neq 0$$

ist

Satz 241. Die Orthogonalpolynome P_i , $i = 0, \dots, m$ sind durch die oben definierten Eigenschaften eindeutig bestimmt. Sie können durch die Rekursion

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x - \frac{(x, 1)}{(1, 1)} \\ P_k(x) &= (x - a_k)P_{k-1}(x) - b_k P_{k-2}(x) \end{aligned} \quad \begin{aligned} a_k &= \frac{(xP_{k-1}, P_{k-1})}{(P_{k-1}, P_{k-1})} \\ b_k &= \frac{(P_{k-1}, P_{k-1})}{(P_{k-2}, P_{k-2})} \end{aligned}$$

berechnet werden.

Beispiel 242.

► es sei $[a, b] = [-1, 1]$, $(f, g) = \int_{-1}^1 f(x)g(x) dx$

► $P_0(x) = 1$

$$P_1(x) = x - \frac{\int_{-1}^1 x dx}{\int_{-1}^1 dx} = x$$

► $P_2(x) = (x - a_2)P_1(x) - b_2 P_0(x) = x \cdot x - \frac{1}{3} = x^2 - \frac{1}{3}$, denn

$$a_2 = \frac{(xP_1, P_1)}{(P_1, P_1)} = \frac{\int_{-1}^1 x^2 \cdot x dx}{\int_{-1}^1 x^2 dx} = 0, \quad b_2 = \frac{(P_1, P_1)}{(P_0, P_0)} = \frac{\int_{-1}^1 x^2 dx}{\int_{-1}^1 dx} = \frac{1}{3}$$

► $P_3(x) = (x - a_3)P_2(x) - b_3 P_1(x) = x(x^2 - \frac{1}{3}) - \frac{4}{15}x$, denn

$$a_3 = \frac{(xP_2, P_2)}{(P_2, P_2)} = 0, \quad b_3 = \frac{(P_2, P_2)}{(P_1, P_1)} = \frac{4}{15}$$

Satz 243. Das Orthogonalpolynom P_k vom Grad k hat auf (a, b) genau k verschiedene reelle Nullstellen x_{ki} , $i = 0, \dots, k - 1$

- der nächste Satz zeigt
 - dass die Nullstellen die optimalen Stützstellen sind
 - wie die Gewichte dazu aussehen
 - wie der Fehler aussieht

Satz 244. Sei P_{n+1} das $(n + 1)$ -te Orthogonalpolynom auf (a, b) mit Nullstellen x_0, \dots, x_n . Dann gilt für die **Gauß-Quadratur**

$$G_n(f) = \sum_{i=0}^n \beta_i f(x_i)$$

mit **Gewichten**

$$\beta_i = \left(P_{n+1}(x_i), \frac{1}{(x - x_i)P'_{n+1}(x_i)} \right), \quad i = 0, \dots, n, \quad (f, g) = \int_a^b f(x)g(x) dx$$

die Fehlerabschätzung

$$\int_a^b f(x) dx - G_n(f) = c h^{2n+3} f^{(2n+2)}(\xi)$$

mit $\xi \in [a, b]$, $h = \frac{b-a}{n}$

Bemerkung 245.

- ◊ mit $n + 1$ Funktionsauswertungen erreicht man h^{2n+3} anstatt h^{n+3} bei den Newton-Cotes Formeln
- ◊ wegen $f^{(2n+2)}(\xi)$ werden Polynome vom Grad $\leq 2n + 1$ exakt integriert
- ◊ der Polynomgrad kann bei gleichem Aufwand nicht mehr verbessert werden

- Praxis:

- die x_i liegen in (a, b) , d.h. nie auf dem Rand
- für verschiedene n sind die x_i komplett verschieden, d.h. beim Übergang von n auf $n + 1$ kommt nicht nur eine neue Stützstelle hinzu, sondern alle Stützstellen verändern sich
- die x_i hängen von a, b ab:
 - x_i werden für $[-1, 1]$ tabelliert
 - für beliebiges $[a, b]$ erhält man:

$$\tilde{x}_i = \frac{b-a}{2}x_i + \frac{b+a}{2}$$

d.h. die relative Verteilung innerhalb des Intervalls bleibt erhalten

- analog hängen auch die Gewichte β_i von (a, b) ab und werden für $[-1, 1]$ tabelliert:

- für $[-1, 1]$ gilt $\sum_{i=0}^n \beta_i = 2$
- für beliebiges $[a, b]$ erhält man

$$\tilde{\beta}_i = \frac{b-a}{2} \beta_i$$

Beispiel 246.

- auf $[-1, 1]$ gilt nach oben

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= x^2 - \frac{1}{3} \\ P_3(x) &= x(x^2 - \frac{3}{5}) \end{aligned}$$

- damit ergeben sich für die ersten drei G_n als Stützstellen (d.h. Nullstellen von P_{n+1})

n	$n+1$	x_0	x_1	x_2
0	1	0		
1	2	$-\sqrt{\frac{1}{3}}$	$\sqrt{\frac{1}{3}}$	
2	3	$-\sqrt{\frac{3}{5}}$	0	$\sqrt{\frac{3}{5}}$

- für die Gewichte β_i gilt bei G_n

$$\beta_i = \int_{-1}^1 P_{n+1}(x) \frac{1}{(x-x_i)P'_{n+1}(x_i)} dx$$

und damit

- $n = 0$

$$\beta_0 = \int_{-1}^1 P_1(x) \frac{1}{(x-x_0)P'_1(x_0)} dx = \int_{-1}^1 x \frac{1}{(x-0) \cdot 1} dx = 2$$

- $n = 1$

$$\begin{aligned} \beta_0 &= \int_{-1}^1 P_2(x) \frac{1}{(x-x_0)P'_2(x_0)} dx = \int_{-1}^1 (x^2 - \frac{1}{3}) \frac{1}{(x-(-\sqrt{\frac{1}{3}}))2(-\sqrt{\frac{1}{3}})} dx = 1 \\ \beta_1 &= 1 \end{aligned}$$

- $n = 2$

$$\beta_0 = \beta_2 = \frac{5}{9}, \quad \beta_1 = \frac{8}{9}$$

Zusammenfassung. Die ersten drei Gauß-Quadraturformeln sind durch folgende Stützstellen und Gewichte gegeben

$$\begin{aligned}
 n = 0 : \quad & x_i = 0 \\
 & \beta_i = 2 \\
 n = 1 : \quad & x_i = -\sqrt{\frac{1}{3}} \quad \sqrt{\frac{1}{3}} \\
 & \beta_i = 1 \quad 1 \\
 n = 2 : \quad & x_i = -\sqrt{\frac{3}{5}} \quad 0 \quad \sqrt{\frac{3}{5}} \\
 & \beta_i = \frac{5}{9} \quad \frac{8}{9} \quad \frac{5}{9}
 \end{aligned} \tag{10.1}$$

Beispiel 247.

- wir wenden die Gauß-Quadraturformeln auf $\int_1^2 \frac{1}{x} dx = \ln(2)$ an
- dazu müssen die x_i und β_i zunächst von $[-1, 1]$ auf $[1, 2]$ umgerechnet werden

$$\tilde{x}_i = \frac{b-a}{2}x_i + \frac{b+a}{2} = \frac{1}{2}x_i + \frac{3}{2}, \quad \tilde{\beta}_i = \frac{b-a}{2}\beta_i = \frac{1}{2}\beta_i$$

- für $n = 1$ ist

$$\begin{aligned}
 \tilde{x}_0 &= \frac{3 - \sqrt{\frac{1}{3}}}{2} & \tilde{x}_1 &= \frac{3 + \sqrt{\frac{1}{3}}}{2} \\
 \tilde{\beta}_0 &= \frac{1}{2} & \tilde{\beta}_1 &= \frac{1}{2}
 \end{aligned}$$

und damit

$$G_1(f) = \tilde{\beta}_0 f(\tilde{x}_0) + \tilde{\beta}_1 f(\tilde{x}_1) = \frac{1}{2} \left(\frac{2}{3 - \sqrt{\frac{1}{3}}} + \frac{2}{3 + \sqrt{\frac{1}{3}}} \right) = \frac{3 + \sqrt{\frac{1}{3}} + 3 - \sqrt{\frac{1}{3}}}{9 - \frac{1}{3}} = \frac{9}{13} \approx 0.69231$$

mit relativem Fehler $e_r \approx 0.0012$

- für $n = 2$ erhalten wir analog

$$G_2(f) = \frac{131}{189} \approx 0.693122$$

mit relativem Fehler $e_r \approx 0.000037$

- ein Vergleich mit den Newton-Cotes Verfahren auf Basis der relativen Fehler liefert

n	Anzahl Funktionsauswertungen	Newton-Cotes	Gauß
1	2	(Trapez) ≈ 0.082	≈ 0.0012
2	3	(Simpson) ≈ 0.019	≈ 0.000037

Bemerkung 248.

- ◊ man kann zeigen, dass die Nullstellen x_i von P_{n+1} Eigenwerte einer einfach zu bestimmenden symmetrischen Tridiagonalmatrix sind, so dass das Nullstellenproblem mit dem Jacobi- oder QR-Verfahren sehr einfach zu lösen ist
- ◊ die Gauß-Formeln können auch als summierte Varianten eingesetzt werden, wobei Nachteile beim Aufwand entstehen, da keine Stützstellen auf den Intervallrändern liegen
- ◊ für große n bekommt man auch hier Stabilitätsprobleme, da ja nach wie vor ein Interpolationspolynom benutzt wird
- ◊ eine Verallgemeinerung mit Gewichtsfunktion $w(x) > 0$ im Skalarprodukt ist möglich, d.h.

$$(f, g) = \int_a^b w(x) f(x) g(x) dx$$

10.4 Extrapolation

- **summierte Newton Cotes Formeln** sind einfach anzuwenden:
 - der Polynomgrad kann niedrig gehalten werden, was Instabilitäten vermeidet
 - die Implementierung ist einfach
- ein typischer Vertreter ist die summierte Trapez-Regel, die im äquidistanten Fall gegeben ist durch

$$\begin{aligned} T_m(f) &= \frac{l}{2} (f(a_0) + 2f(a_1) + \dots + 2f(a_{m-1}) + f(a_m)) \\ &= \sum_{i=0}^{m-1} l \left(\frac{1}{2} f(a_i) + \frac{1}{2} f(a_{i+1}) \right) \end{aligned}$$

mit $l = \frac{b-a}{m}$, $a_i = a + i l$

- für den Fehler erhalten wir durch Einsetzen der Fehlerformel der *einfachen* Trapez-Regel

$$\begin{aligned} E_m(f) &:= \int_a^b f(x) dx - T_m(f) \\ &= \sum_{i=0}^{m-1} \left(\int_{a_i}^{a_{i+1}} f(x) dx - I_1(f)|_{[a_i, a_{i+1}]} \right) \\ &= \sum_{i=0}^{m-1} -\frac{1}{12} l^3 f''(\xi_i) \end{aligned}$$

so dass

$$|E_m(f)| \leq \frac{ml^3}{12} \max_{\xi \in [a,b]} |f''(\xi)| = l^2 \frac{b-a}{12} \max_{\xi \in [a,b]} |f''(\xi)|, \quad l = \frac{b-a}{m},$$

d.h. $|E_m(f)| \sim l^2$ für $l \rightarrow 0$

- um kleine Fehler zu erhalten kann man

- l klein machen (viele Unterteilungen von $[a, b]$ benutzen), was den Aufwand hoch treibt
- die Ordnung erhöhen (d.h. einen höheren Polynomgrad benutzen) was ebenfalls mehr Aufwand bedeutet und durch die Instabilitäten bei hohem Polynomgrad nach oben begrenzt ist
- kann man mit einfachen Mitteln hochgenaue, stabile Verfahren herleiten?
- dazu betrachten wir die äquidistante, summierte Trapez-Regel nochmal genauer

Satz 249 (Euler-McLaurin). Sei $f \in C^{2k+1}[a, b]$, $l = \frac{b-a}{m}$, dann ist

$$T_m(f) = \int_a^b f(t)dt + \tau_1 l^2 + \tau_2 l^4 + \dots + \tau_k (l^2)^k + R_{k+1}(l)(l^2)^{k+1}$$

für $k \geq 1$ wobei

$$|R_{k+1}(l)| \leq c_{k+1} |b-a|$$

und die τ_i und c_{k+1} nicht von m abhängen (nur von a, b, f)

Bemerkung 250.

- ◊ das ist eine **asymptotische Entwicklung** des Fehlers in Potenzen von l^2
- ◊ wesentlich dabei ist die Unabhängigkeit der Koeffizienten von m

- wir benutzen nun zur Approximation von $\int_a^b f(t)dt$ die summierte Trapez-Regel mit zwei unterschiedlichen Intervalllängen $l_0 = \frac{b-a}{m_0}$ bzw. $l_1 = \frac{b-a}{m_1}$, d.h.

$$\begin{aligned} T_{m_0}(f) &= \int_a^b f(t)dt + \tau_1 l_0^2 + R_2(l_0)l_0^4 \\ T_{m_1}(f) &= \int_a^b f(t)dt + \tau_1 l_1^2 + R_2(l_1)l_1^4 \end{aligned} \tag{10.2}$$

- setze

$$T = \alpha_0 T_{m_0} + \alpha_1 T_{m_1} \tag{10.3}$$

und bestimme α_0, α_1 so, dass

$$T = \int_a^b f(t)dt + \beta_0 l_0^4 + \beta_1 l_1^4$$

d.h. T enthält keine quadratischen Terme in l_0 bzw. l_1 mehr

- damit ist T eine Integralapproximation vierter Ordnung (und nicht zweiter Ordnung, wie die Trapez-Regel)

- zur Bestimmung von α_0, α_1 setzen wir (10.2) in (10.3) ein

$$\begin{aligned} T &= \alpha_0 T_{m_0} + \alpha_1 T_{m_1} \\ &= (\alpha_0 + \alpha_1) \int_a^b f(t) dt + \tau_1(\alpha_0 l_0^2 + \alpha_1 l_1^2) + \alpha_0 R_0(l_0) l_0^4 + \alpha_1 R_0(l_1) l_1^4 \end{aligned}$$

- damit T weiterhin das Integral annähert und die quadratischen Terme verschwinden, muss

$$\alpha_0 + \alpha_1 = 1, \quad \alpha_0 l_0^2 + \alpha_1 l_1^2 = 0,$$

d.h. α_0, α_1 müssen das folgende lineare Gleichungssystem lösen

$$\begin{pmatrix} 1 & 1 \\ l_0^2 & l_1^2 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- mit

$$\alpha_0 = \frac{l_1^2}{l_1^2 - l_0^2}, \quad \alpha_1 = -\frac{l_0^2}{l_1^2 - l_0^2}$$

erhalten wir für T schließlich

$$T = \frac{l_1^2 T_{m_0} - l_0^2 T_{m_1}}{l_1^2 - l_0^2} = \int_a^b f(t) dt + R l^4, \quad l = \max(l_0, l_1)$$

- das Verfahren kann analog auf q verschiedene Schrittweiten erweitert werden
- es sei $m_0, \dots, m_{q-1}, l_i = \frac{b-a}{m_i}$, und $I := \int_a^b f(t) dt$
- aus der Euler-McLaurin Entwicklung folgt

$$\begin{aligned} T_{m_0} &= I + \tau_1 l_0^2 + \tau_2(l_0^2)^2 + \dots + \tau_{q-1}(l_0^2)^{q-1} + R_q(l_0)(l_0^2)^q \\ T_{m_1} &= I + \tau_1 l_1^2 + \tau_2(l_1^2)^2 + \dots + \tau_{q-1}(l_1^2)^{q-1} + R_q(l_1)(l_1^2)^q \\ &\vdots \\ T_{m_{q-1}} &= I + \tau_1 l_{q-1}^2 + \tau_2(l_{q-1}^2)^2 + \dots + \tau_{q-1}(l_{q-1}^2)^{q-1} + R_q(l_{q-1})(l_{q-1}^2)^q \end{aligned}$$

- wir setzen

$$\begin{aligned} T &= \sum_{i=0}^{q-1} \alpha_i T_{m_i} \\ &= \left(\sum_{i=0}^{q-1} \alpha_i \right) I + \tau_1 \left(\sum_{i=0}^{q-1} \alpha_i l_i^2 \right) + \dots + \tau_{q-1} \left(\sum_{i=0}^{q-1} \alpha_i (l_i^2)^{q-1} \right) + R l^{2q} \end{aligned}$$

mit $l = \max_{i=0, \dots, q-1} l_i$

- analog zu oben folgt

$$\begin{array}{ll} \sum \alpha_i = 1 & \\ \sum \alpha_i l_i^2 = 0 & \\ \sum \alpha_i (l_i^2)^2 = 0 & \\ \vdots & \\ \sum \alpha_i (l_i^2)^{q-1} = 0 & \end{array} \Leftrightarrow \underbrace{\begin{pmatrix} 1 & \dots & 1 \\ l_0^2 & \dots & l_{q-1}^2 \\ (l_0^2)^2 & \dots & (l_{q-1}^2)^2 \\ \vdots & & \vdots \\ (l_0^2)^{q-1} & \dots & (l_{q-1}^2)^{q-1} \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{q-1} \end{pmatrix}}_{=:a} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{=:e} \quad (10.4)$$

- bestimme α_i aus $Aa = e$:

- A^T ist die Vandermonde-Matrix zu $x_i = l_i^2$
- A ist regulär $\Leftrightarrow A^T$ regulär $\Leftrightarrow l_i$ sind paarweise verschieden ($l_i > 0$)
- $A\alpha = e$ ist also immer eindeutig lösbar, falls für die T_{m_i} paarweise verschiedene Trapezbreiten benutzt werden

Zusammenfassung.

➢ rechne für q Schrittweiten $l_i = \frac{b-a}{m_i}$, $l_0 > l_1 > \dots > l_{q-1}$, die summierte Trapez-Regel $T_{m_i}(f)$

➢ löse $A\alpha = e$

➢ berechne

$$T = \sum_{i=0}^{q-1} \alpha_i T_{m_i} = \alpha^T t, \quad t := \begin{pmatrix} T_{m_0} \\ \vdots \\ T_{m_{q-1}} \end{pmatrix} \quad (10.5)$$

➢ dann gilt

$$T = \int_a^b f(t) dt + R(l^2)^q,$$

$$l = \max_{i=0,\dots,q-1} l_i, \text{ d.h. } T \text{ hat Ordnung } 2q \text{ für } l \rightarrow 0$$

Beispiel 251.

► wir wenden das Verfahren auf

$$\int_0^1 \frac{1}{1+t^2} dt = \arctan(1) = \frac{\pi}{4} \approx 0.785398$$

an

► $q = 1$

- in diesem Fall wird mit einer einzigen Trapezbreite gearbeitet
- wir benutzen $l_0 = 1$ also $m_0 = 1$ und erhalten die gewöhnliche summierte Trapezregel

$$T = T_{m_0} = \frac{1}{2}(f(0) + f(1)) = \frac{1}{2}(1 + \frac{1}{2}) = \frac{3}{4}$$

► $q = 2$

- wir benutzen die beiden Schrittweiten $l_0 = 1, m_0 = 1$ und $l_1 = \frac{1}{2}, m_1 = 2$
- aus dem Fall $q = 1$ wissen wir, dass $T_{m_0} = \frac{3}{4}$ ist
- für T_{m_1} erhalten wir

$$T_{m_1} = \frac{l_1}{2} \left(f(0) + 2f\left(\frac{1}{2}\right) + f(1) \right) = \frac{1}{4} \left(1 + 2 \frac{1}{1+\frac{1}{4}} + \frac{1}{2} \right) = \frac{31}{40} = 0.775$$

- das System $A\alpha = e$ mit

$$A = \begin{pmatrix} 1 & 1 \\ l_0^2 & l_1^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & \frac{1}{4} \end{pmatrix}, \quad e = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

hat die Lösung

$$\alpha = \begin{pmatrix} -\frac{1}{3} \\ \frac{4}{3} \end{pmatrix}$$

und als Approximation vierter Ordnung ergibt sich

$$T = -\frac{1}{3}T_{m_0} + \frac{4}{3}T_{m_1} = \frac{47}{60} = 0.78\bar{3}$$

- $q = 3$

- als Schrittweiten benutzen wir $l_0 = 1, l_1 = \frac{1}{2}, l_2 = \frac{1}{4}$
- $T_{m_0} = \frac{3}{4}$ und $T_{m_1} = \frac{31}{40}$ können wir aus den vorherigen Schritten übernehmen, für T_{m_2} erhält man

$$T_{m_2} = \frac{5323}{6800} \approx 0.7828$$

- das System $A\alpha = e$ mit

$$A = \begin{pmatrix} 1 & 1 & 1 \\ l_0^2 & l_1^2 & l_2^2 \\ l_0^4 & l_1^4 & l_2^4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & \frac{1}{4} & \frac{1}{16} \\ 1 & \frac{1}{16} & \frac{1}{256} \end{pmatrix}, \quad e = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

hat die Lösung

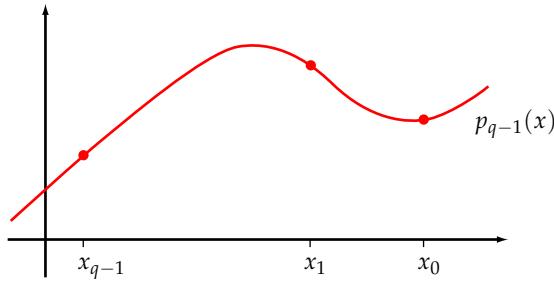
$$\alpha = \frac{1}{45} \begin{pmatrix} 1 \\ -20 \\ 64 \end{pmatrix}$$

und als Approximation sechster Ordnung ergibt sich $T \approx 0.78553$

- die Berechnung von T ist durch das Auflösen von $A\alpha = e$ sehr mühsam
- zur Implementierung wäre eine elegantere Form wünschenswert
- um diese herzuleiten, benutzen wir die folgende geometrische Interpretation von T
- die Trapezbreiten l_i sind gegeben, $l_0 > l_1 > \dots > l_{q-1} > 0$ ($l_i = 0$ geht nicht)
- zu jedem $x_i = l_i^2$ berechnen wir eine Integralnäherung $y_i = T_{m_i}$
- wir interpolieren durch die q Punkte $(x_i, y_i), i = 0, \dots, q-1$ ein Polynom

$$p_{q-1}(x) = p_0 + p_1 x + \dots + p_{q-1} x^{q-1}$$

vom Grad $q-1$



- die Koeffizienten p_i sind gegeben durch

$$\begin{pmatrix} 1 & x_0 & \dots & x_0^{q-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{q-1} & \dots & x_{q-1}^{q-1} \end{pmatrix} \begin{pmatrix} p_0 \\ \vdots \\ p_{q-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_{q-1} \end{pmatrix}$$

also

$$\begin{pmatrix} 1 & l_0^2 & \dots & (l_0^2)^{q-1} \\ \vdots & \vdots & & \vdots \\ 1 & l_{q-1}^2 & \dots & (l_{q-1}^2)^{q-1} \end{pmatrix} \underbrace{\begin{pmatrix} p_0 \\ \vdots \\ p_{q-1} \end{pmatrix}}_{=:p} = \begin{pmatrix} T_{m_0} \\ \vdots \\ T_{m_{q-1}} \end{pmatrix}$$

- benutzen wir (10.4) und (10.5), so können wir dieses System schreiben als

$$A^T p = t$$

- andererseits gilt nach (10.5)

$$T = \alpha^T t, \quad A\alpha = e, \quad e = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- damit erhalten wir

$$T = \alpha^T t \stackrel{A^T p = t}{=} \alpha^T A^T p = (A\alpha)^T p \stackrel{A\alpha = e}{=} e^T p = p_0$$

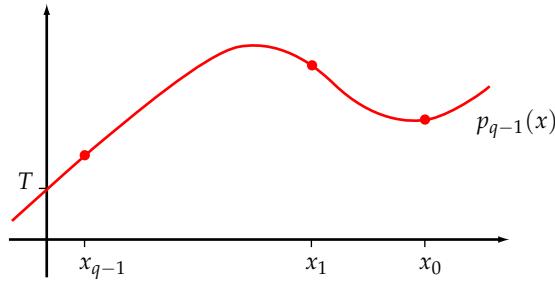
so dass

$$T = p_0 = p_{q-1}(0)$$

wobei $p_{q-1}(x)$ das Interpolationspolynom durch die Punkte (l_i^2, T_{m_i}) ist

Zusammenfassung.

- betrachte (l_i^2, T_{m_i}) , $i = 0, \dots, q-1$, $l_i > 0$
- bestimme das Interpolationspolynom $p_{q-1}(x)$ durch diese Punkte
- berechne $T = p_{q-1}(0)$



- da 0 wegen $l_i > 0$ nicht zwischen zwei Stützstellen $x_i = l_i^2$, liegen kann, nennt man diese Auswertung **Extrapolation**
- bei q Schrittweiten l_0, \dots, l_{q-1} hat T die Fehlerordnung $2q$

- wieso liefert dieses Vorgehen vernünftige Ergebnisse?
 - für den Fehler der summierten Trapez-Regel gilt

$$|T_m - \int_a^b f(t)dt| = c l^2, \quad l = \frac{b-a}{m}$$
 - für $l \rightarrow 0$ (d.h. $m \rightarrow \infty$) konvergiert T_m gegen den exakten Integralwert
 - idealerweise würde man also die Trapez-Regel T_m für $l = 0$ bzw. $m = \infty$ auswerten, was aber nicht möglich ist
 - die Extrapolation bestimmt T_m für verschiedene l , versucht den Trend für $l \rightarrow 0$ abzuschätzen und damit eine gute Näherung für $l = 0$ (d.h. für das exakte Integral) zu erreichen
- analog zur Newton-Interpolation soll das Extrapolationsschema leicht erweiterbar aufgebaut werden
 - starte mit einer Schrittweite und berechne $T = T_{m_0}$
 - reicht die Genauigkeit nicht aus, so fügt man eine weitere Schrittweite hinzu und bestimmt ein neues T durch $p_{q-1}(0)$
 - das wiederholt man so lange, bis die gewünschte Genauigkeit erreicht ist
- pro Schrittweite erhalten wir einen neuen Interpolationspunkt, d.h. wir erweitern das Polynom wie bei der Newton-Interpolation
- nach jeder Erweiterung muss das neue Polynom an $\hat{x} = 0$ ausgewertet werden um das neue T zu erhalten
- das Auswerten des Interpolationspolynoms an einer gegebenen Stelle \hat{x} kann sehr einfach durch das **Aitken-Neville-Schema** erfolgen, das mit den dividierten Differenzen verwandt ist:
 - gegeben sind die Punkte $(x_i, y_i), i = 0, 1, 2, \dots$
 - setze $P_{i0} = y_i$ und
$$P_{ij} = P_{ij-1} + \frac{\hat{x} - x_i}{x_i - x_{i-j}} (P_{ij-1} - P_{i-1j-1}), \quad i \geq j \geq 1$$

d.h.

$$\begin{array}{lll}
 y_0 & = & P_{00} \\
 & & \searrow \\
 y_1 & = & P_{10} \rightarrow P_{11} \\
 & & \searrow \quad \searrow \\
 y_2 & = & P_{20} \rightarrow P_{21} \rightarrow P_{22} \\
 & \vdots & \vdots \quad \vdots \quad \vdots \quad \ddots
 \end{array}$$

dann ist $P_{kk} = p_k(\hat{x})$, wobei p_k das Interpolationspolynom durch $(x_0, y_0), \dots, (x_k, y_k)$ ist

- in unserem Fall gilt $\hat{x} = 0, x_i = l_i^2, y_i = T_{m_i}$ so dass $P_{i0} = T_{m_i}$,

$$P_{ij} = P_{ij-1} + \frac{P_{ij-1} - P_{i-1j-1}}{\left(\frac{l_{i-j}}{l_i}\right)^2 - 1} \quad i \geq j \geq 1$$

und $T = P_{q-1q-1}$

- wir bauen also zeilenweise folgendes Schema auf

$$\begin{array}{cccccc}
 \text{Ordnung} & 2 & 4 & 6 & \dots \\
 \hline
 T_{m_0} & = & P_{00} & & & \\
 T_{m_1} & = & P_{10} & P_{11} & & \\
 T_{m_2} & = & P_{20} & P_{21} & P_{22} &
 \end{array}$$

und erhalten die Werte für das jeweilige T aus der Diagonalen

Satz 252. Sei T die zu q Schrittweiten $l_0 > l_1 > \dots > l_{q-1}$ extrapolierte äquidistante summierte Trapez-Regel. Dann ist $T = P_{q-1q-1}$ mit

$$P_{i0} = T_{m_i} \quad i = 0, \dots, q-1$$

$$P_{ij} = P_{ij-1} + \frac{P_{ij-1} - P_{i-1j-1}}{\left(\frac{l_{i-j}}{l_i}\right)^2 - 1} \quad 1 \leq j \leq i \leq q-1.$$

T approximiert $\int_a^b f(t) dt$ für $f \in C^{2q-1}[a, b]$ mit Ordnung $2q$.

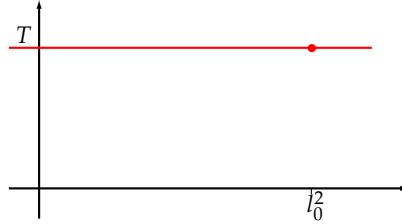
Beispiel 253.

- wir betrachten wieder $\int_0^1 \frac{1}{1+x^2} dx$ und wenden zunächst wie in Beispiel 251 die summierte Trapez-Regel mit drei verschiedenen Trapezbreiten l_i an:

i	0	1	2
l_i	1	$\frac{1}{2}$	$\frac{1}{4}$
T_{m_i}	$\frac{3}{4}$	$\frac{31}{40}$	$\frac{5323}{6800}$

- $q = 1$

- hier ist $T = T_{m_0} = \frac{3}{4}$, also eine Approximation zweiter Ordnung



- $q = 2$

- wir erweitern das Extrapolationsschema um eine Zeile und erhalten

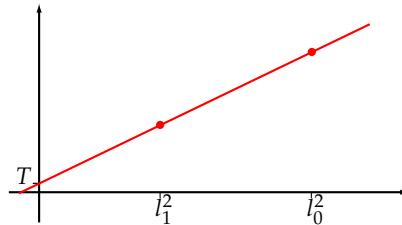
$$\frac{3}{4} = P_{00}$$

$$\frac{31}{40} = P_{10} \quad P_{11} = P_{10} + \frac{P_{10} - P_{00}}{\left(\frac{l_0}{l_1}\right)^2 - 1} = \frac{47}{60}$$

denn

$$P_{10} + \frac{P_{10} - P_{00}}{\left(\frac{l_0}{l_1}\right)^2 - 1} = \frac{31}{40} + \frac{\frac{31}{40} - \frac{3}{4}}{2^2 - 1} = \frac{31}{40} + \frac{1}{120} = \frac{94}{120} = \frac{47}{60}$$

- damit ist $T = P_{11} = \frac{47}{60}$ eine Approximation vierter Ordnung

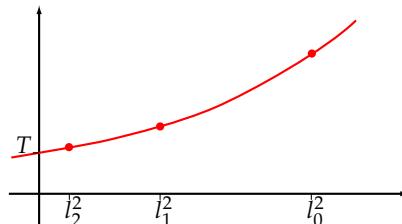


- $q = 3$

- durch Anfügen einer weiteren Zeile erhalten wir

$$\begin{array}{ll} \frac{3}{4} & \\ \frac{31}{40} & \frac{47}{60} \\ \frac{5323}{6800} & P_{21} = P_{20} + \frac{P_{20} - P_{10}}{\left(\frac{l_1}{l_2}\right)^2 - 1} = \frac{8011}{10200} \quad P_{22} = P_{21} + \frac{P_{21} - P_{11}}{\left(\frac{l_0}{l_2}\right)^2 - 1} = \frac{6677}{8500} \end{array}$$

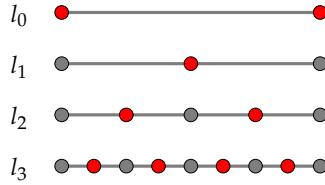
und $T = P_{22} = \frac{6677}{8500}$ ist eine Approximation sechster Ordnung



- insgesamt erhalten wir

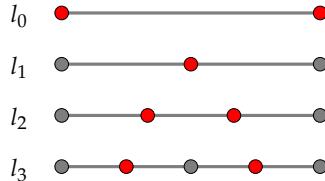
q	0	1	2
Ordnung	2	4	6
T	$\frac{3}{4}$	$\frac{47}{60}$	$\frac{6677}{8500}$
\approx	0.75	0.783	0.78553

- im Beispiel wurden die Schrittweiten fortlaufend halbiert, weshalb man viele Funktionswerte wiederverwenden kann



- die Anzahl der Funktionsauswertungen steigt trotzdem stark an
- diese Folge der l_i heißt **Romberg-Folge**
- die l_i müssen nicht notwendig fortlaufend halbiert werden, sondern nur paarweise verschieden sein
- eine alternative Wahl der Trapezbreiten ist durch die **Bulirsch-Folge** gegeben

$$l_0 = b - a, \quad l_1 = \frac{l_0}{2}, \quad l_2 = \frac{l_0}{3}, \quad l_i = \frac{l_{i-2}}{2} \quad i > 2$$



- für die Anzahl der Funktionsauswertungen bei optimaler Implementierung erhält man

Ordnung	2	4	6	8	10	12	14
Romberg	2	3	5	9	17	33	65
Bulirsch	2	3	5	7	9	13	17

- bei der Bulirsch-Folge sind für eine gegebene Fehlerordnung sehr viel weniger Auswertungen nötig, aber:

- die einzelnen l_i sind größer als bei der Romberg-Folge, so dass die einzelnen $T_{m_i} = P_{i0}$ weniger genau sind
- T kann für alle q als Integrationsformel mit Gewichten α_i geschrieben werden

$$T = \sum \alpha_i f(a_i)$$

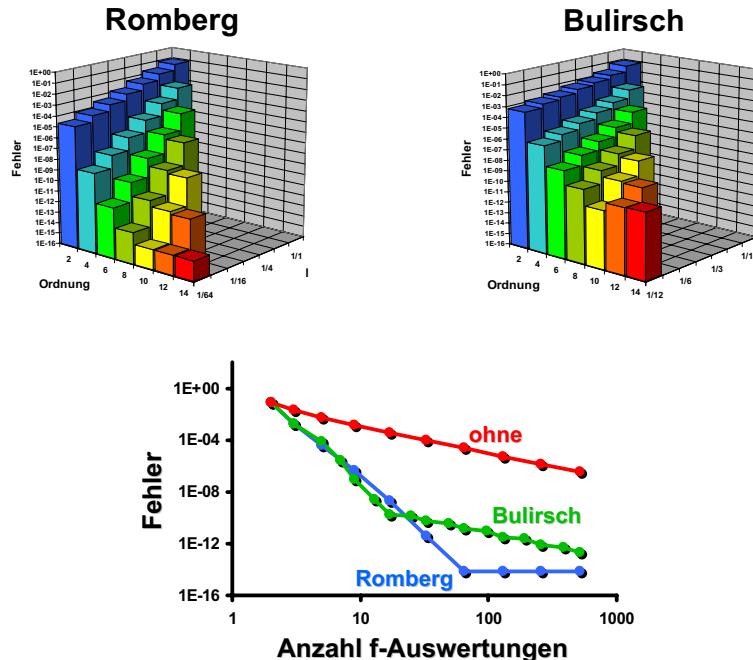
- für die Romberg-Folge sind für alle q und alle i die $\alpha_i > 0$
- bei der Bulirsch-Folge kommen ab Ordnung 6 negative Gewichte vor, was wieder zu Stabilitätsproblemen führt

- Fazit:

- bei hoher Ordnung ist die Romberg-Folge wegen der höheren Stabilität vorzuziehen
- bei geringer Ordnung verursacht die Bulirsch-Folge den geringeren Aufwand

Beispiel 254.

► wir betrachten $\int_1^2 \frac{1}{x} dx$



Zusammenfassung.

- Extrapoliere um T_m für $l \rightarrow 0$ ($m \rightarrow \infty$) abzuschätzen
- funktioniert wegen asymptotischer Entwicklung des Fehlers mit von l (m) unabhängigen Koeffizienten (Euler-McLaurin)
- die Auswertung erfolgt mit dem Aitken-Neville-Schema
- Idee ist nicht beschränkt auf Integration, sondern kann bei jedem numerischen Verfahren mit asymptotischer Fehlerentwicklung eingesetzt werden

10.5 Adaptive Verfahren

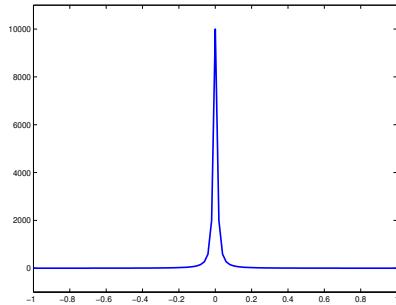
- die Genauigkeit einer Integralapproximation kann erhöht werden durch
 - kleinere Schrittweite h bzw. l bei den Newton-Cotes-Formeln
 - höhere Fehlerordnung (Extrapolation)
- in beiden Fällen wächst der Aufwand, d.h. die Anzahl der Funktionsauswertungen
- wie wir in Beispiel 254 gesehen haben, liefern die Romberg- bzw. Bulirsch-Extrapolation für bestimmte Integranden sehr effiziente, hochgenaue Approximationen
- unter bestimmten Umständen treten aber erhebliche Schwierigkeiten auf

Beispiel 255.

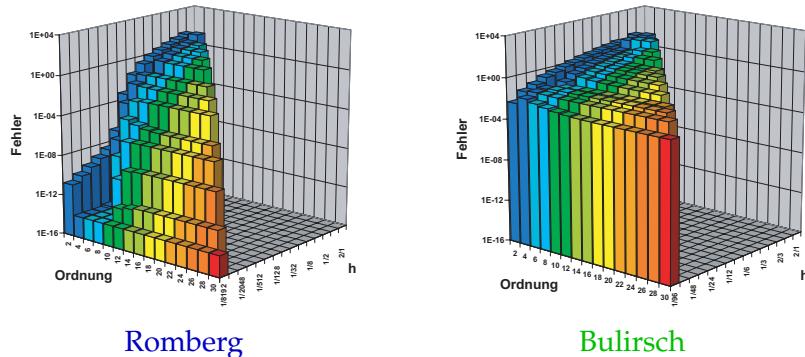
- wir bestimmen eine Approximation von

$$\int_{-1}^1 \frac{1}{10^{-4} + x^2} dx = 200 \arctan(100) \approx 312.159$$

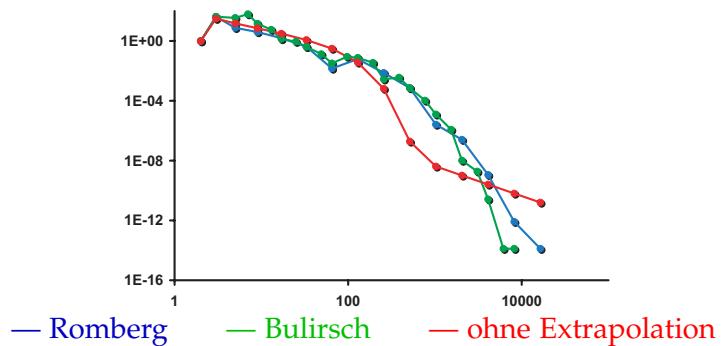
- der Integrand (**Nadelimpuls**) hat folgende Form



- bei Extrapolation mit dem Romberg- bzw. Bulirsch-Ansatz erhalten wir



bzw.



- auf dem größten Teil des Intervalls $[-1, 1]$ ist keine Approximation hoher Ordnung nötig
- Extrapolationsergebnisse sind erst dann gut, wenn die zugrunde liegenden Trapezregeln auf einem hinreichend feinen Gitter arbeiten, so dass die Spitze bei 0 vernünftig aufgelöst wird
- die einfache summierte Trapez-Regel ist teilweise effizienter, die Steigerung der Ordnung durch Extrapolation bringt wenig

- das Problem im letzten Beispiel ist die starke Lokalisierung des Integranden um die 0
- alle bisher betrachteten Verfahren erzeugen auf dem ganzen Integrationsintervall $[a, b]$ eine gleichmäßige Approximation des Integranden mit einer gewissen Fehlerordnung
- sehr lokales Verhalten des Integranden wird nicht berücksichtigt
- summierte Newton-Cotes-Formeln können leicht entsprechend modifiziert werden:
 - die Grundidee summierten Newton-Cotes-Formeln ist:
 - zerlege $[a, b]$ in m Teilintervalle $[a_i, b_i]$ mit

$$a = a_0 < b_0 = a_1 < b_1 \dots < b_{m-1} =: a_m = b$$
 - zerlege das Integral in eine Summe von Integralen auf den Teilintervallen

$$\int_a^b f(x) dx = \sum_{i=0}^{m-1} \int_{a_i}^{b_i} f(x) dx$$
 - wende für jedes Teilintervall $[a_i, b_i]$ eine Newton-Cotes-Formel zur Approximation von $\int_{a_i}^{b_i} f(x) dx$ an
 - bisher haben wir nur gleich lange Teilintervalle $[a_i, b_i]$ benutzt
 - wähle jetzt die a_i, b_i so, dass in kritischen Bereichen (0 beim Nadelimpuls) sehr viele, kurze bzw. in den anderen Bereichen wenige große Teilintervalle liegen
- die a_i, b_i sollen vom Algorithmus selbst in Abhängigkeit vom Integranden **adaptiv** gewählt werden
- dabei geht man wie folgt vor:
 - wir starten mit einer Anfangszerlegung von $[a, b]$ in Teilintervalle $[a_i, b_i]$
 - auf jedem Teilintervall wenden wir die *einfache* Newton-Cotes-Formel an und erhalten eine Integralapproximation von $\int_{a_i}^{b_i} f(x) dx$
 - wir schätzen den Fehler dieser Integralapproximation ab
 - deutet die Fehlerschätzung auf eine schlechte Approximation hin, dann unterteilen wir $[a_i, b_i]$ in der Mitte in 2 gleichgroße Teilintervalle
 - nachdem alle Teilintervalle der Anfangszerlegung so untersucht und ggf. geteilt wurden, erhalten wir eine neue Zerlegung von $[a, b]$
 - die neue Zerlegung untersuchen wir analog zur Anfangszerlegung auf Teilintervalle mit großen Approximationsfehlern
 - das wiederholen wir so lange, bis alle Teilintervalle unseren Anforderungen genügen
 - als Approximation des Integrals über $[a, b]$ benutzen wir die summierte Newton-Cotes-Formel auf der letzten erzeugten Zerlegung
- die Grundlagen dieser adaptiven Steuerung sollen anhand des summierten Simpson-Verfahrens erläutert werden
- dabei genügt es, ein einzelnes Teilintervalls $[a_i, b_i]$ zu betrachten
- wir wenden zunächst die einfache Simpson-Regel auf $[a_i, b_i]$ an, d.h. mit $h = b_i - a_i$ erhalten wir

$$S_i = \frac{h}{6} \left(f(a_i) + 4f\left(\frac{a_i + b_i}{2}\right) + f(b_i) \right) \approx \int_{a_i}^{b_i} f(x) dx$$
- den Fehler E dieser Approximation schätzen wir wie folgt ab:

- nach Satz 236 gilt

$$E = \int_{a_i}^{b_i} f(x) dx - S = -\frac{1}{90} h^5 f^{(IV)}(\xi), \quad \xi \in [a_i, b_i], \quad h = b_i - a_i$$

- $f^{(IV)}$ ist in der Regel nicht bekannt
- zur Fehlerabschätzung benutzen wir deshalb folgenden Zugang:
 - teile das Intervall in der Mitte

$$[a_l, b_l] = [a_i, m_i], \quad [a_r, b_r] = [m_i, b_i], \quad m_i = \frac{a_i + b_i}{2}$$

- wende auf beide Hälften die einfache Simpson-Regel an, d.h. $h_l = h_r = \frac{h}{2}$ und

$$\begin{aligned} E_l &= \int_{a_i}^{m_i} f(x) dx - S_l = -\frac{1}{90} \left(\frac{h}{2}\right)^5 f^{(IV)}(\xi_l), & \xi_l \in [a_i, m_i], \\ E_r &= \int_{m_i}^{b_i} f(x) dx - S_r = -\frac{1}{90} \left(\frac{h}{2}\right)^5 f^{(IV)}(\xi_r), & \xi_r \in [m_i, b_i], \end{aligned}$$

- damit erhalten wir eine neue (in der Regel genauere) Näherung

$$\tilde{S} = S_l + S_r$$

von

$$\int_{a_i}^{b_i} f(x) dx$$

mit Fehlerabschätzung

$$\tilde{E} = \int_{a_i}^{b_i} f(x) dx - \tilde{S} = E_l + E_r = -\left(\frac{1}{2}\right)^4 \frac{1}{90} h^5 \frac{f^{(IV)}(\xi_l) + f^{(IV)}(\xi_r)}{2}$$

- unter der Annahme

$$f^{(IV)}(\xi_l) \approx f^{(IV)}(\xi_r) \approx f^{(IV)}(\xi)$$

folgt

$$\tilde{E} \approx \frac{1}{16} E$$

und damit

$$\tilde{S} - S = E - \tilde{E} \approx \frac{15}{16} E \approx 15 \tilde{E} \quad (10.6)$$

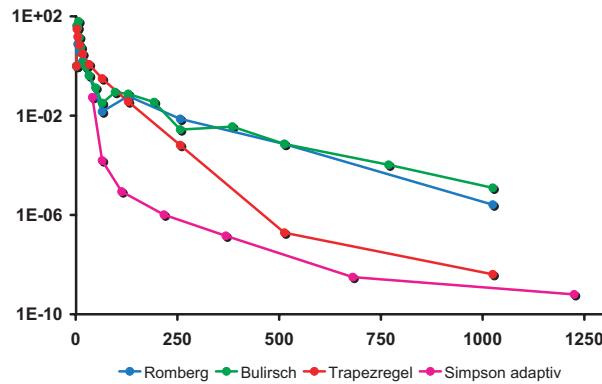
- wir berechnen also S, \tilde{S} pro Intervall und erhalten aus (10.6) **Fehlerschätzer** F, \tilde{F} für S, \tilde{S}

$$F = \frac{16}{15} |\tilde{S} - S| \approx E, \quad \tilde{F} = \frac{1}{15} |\tilde{S} - S| \approx \tilde{E}$$

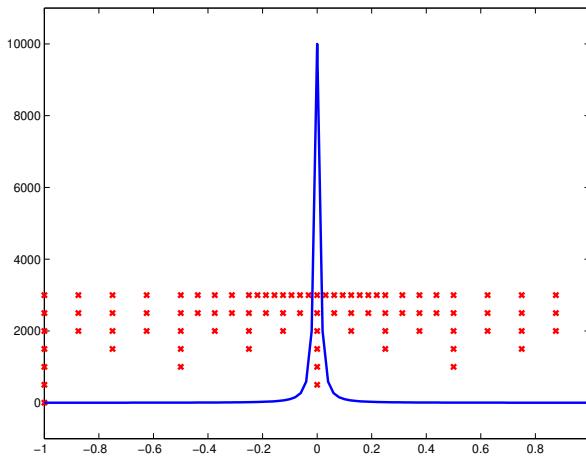
- \tilde{S} ist eine summierte Simpson-Regel
- wegen der höheren Genauigkeit benutzt man \tilde{S}, \tilde{F} und nicht S, F
- die Auswertungen von S_l, S_r können bei Verfeinerung wiederverwendet werden

Beispiel 256.

- wir betrachten nochmals die Aufgabenstellung von Beispiel 255 und wenden die äquidistante summierte Trapezregel, die beiden Extrapolationsmethoden sowie das adaptive Simpson-Verfahren an
- vergleicht man den relativen Fehler mit der Anzahl der benötigten Funktionsauswertungen des Integranden, so erhält man



- insbesondere bei niedriger bzw. mittlerer Genauigkeit (10^{-2} bis 10^{-6}) ist das adaptive Verfahren deutlich effizienter
- eine typische Folge adaptiver Unterteilungen beim Nadelimpuls sieht wie folgt aus:



Kapitel 11

Optimierung ohne Nebenbedingungen

11.1 Grundlagen

- wir suchen Minima einer gegebenen **Zielfunktion**

$$f : \mathbb{R}^n \rightarrow \mathbb{R}, \quad x \mapsto f(x),$$

wobei man x als die Parameter des Optimierungsproblems bezeichnet

- alternativ maximieren wir f indem $-f$ minimiert wird
- wir haben keine weiteren Einschränkungen an die Parameter x (wie z.B. $x_i \geq 0$)
- deshalb bezeichnet man diese Aufgabenstellung als Optimierung ohne Nebenbedingungen bzw. **nicht restriktive Optimierung**
- wir unterscheiden verschiedene Typen von Minima

Definition 257. Sei $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\tilde{x} \in \mathbb{R}^n$.

- \tilde{x} ist ein **globales Minimum** von f , falls

$$f(\tilde{x}) \leq f(x) \quad \forall x \in \mathbb{R}^n$$

- \tilde{x} ist ein **lokales Minimum** von f , falls ein $\varepsilon > 0$ existiert mit

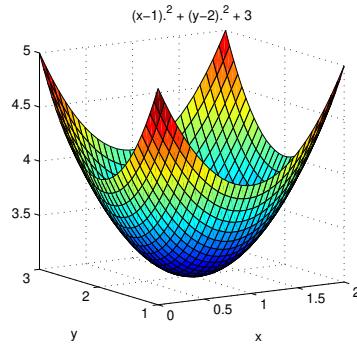
$$f(\tilde{x}) \leq f(x) \quad \forall x \in \mathbb{R}^n, \quad \|x - \tilde{x}\| < \varepsilon$$

Beispiel 258.

- ▶ die Funktion

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2)^2 + 3$$

hat ein globales Minimum bei $x = (1, 2)^T$



- die Funktion

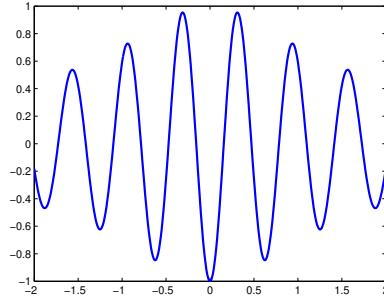
$$f(x) = -\cos(x)$$

hat unendlich viele globale Minima, nämlich $x_k = 2k\pi, k \in \mathbb{Z}$

- die Funktion

$$f(x) = -\frac{\cos(10x)}{\sqrt{1+x^2}}$$

hat ein globales Minimum bei $x = 0$ und unendlich viele lokale Minima



- ist $f : \mathbb{R} \rightarrow \mathbb{R}$ hinreichend oft differenzierbar, so kann man über die Ableitungen von f hinreichende und notwendige Bedingungen für lokale Minima angeben:
 - ist \tilde{x} ein lokales Minimum (Maximum), dann ist $f'(\tilde{x}) = 0$ und $f''(\tilde{x}) \geq 0$ (≤ 0) (notwendige Bedingung)
 - ist $f'(\tilde{x}) = 0$ und $f''(\tilde{x}) > 0$ (< 0), dann ist \tilde{x} ein lokales Minimum (Maximum) (hinreichende Bedingung)
 - ist $f'(\tilde{x}) = 0$ und $f''(\tilde{x}) = 0$, dann müssen Ableitungen höherer Ordnung untersucht werden, da auch ein Sattelpunkt vorliegen kann
- für Funktionen $f : \mathbb{R}^n \rightarrow \mathbb{R}$ erhalten wir analoge Ergebnisse
- auch sie können mit Hilfe der Taylor-Entwicklung von f bewiesen werden

Lemma 259. Ist $f : \mathbb{R}^n \rightarrow \mathbb{R}$ in \tilde{x} dreimal stetig differenzierbar, so gilt

$$f(x) = f(\tilde{x}) + f'(\tilde{x})^T(x - \tilde{x}) + \frac{1}{2}(x - \tilde{x})^T f''(\tilde{x})(x - \tilde{x}) + O(\|x - \tilde{x}\|^3)$$

mit **Gradient**

$$f'(x) = \begin{pmatrix} \partial_{x_1} f(x) \\ \vdots \\ \partial_{x_n} f(x) \end{pmatrix}$$

und **Hesse-Matrix**

$$f''(x) = \begin{pmatrix} \partial_{x_1} \partial_{x_1} f(x) & \dots & \partial_{x_1} \partial_{x_n} f(x) \\ \vdots & & \vdots \\ \partial_{x_n} \partial_{x_1} f(x) & \dots & \partial_{x_n} \partial_{x_n} f(x) \end{pmatrix}$$

- wir erhalten folgende Aussagen:

Satz 260. $f : \mathbb{R}^n \rightarrow \mathbb{R}$ sei hinreichend oft differenzierbar, $\tilde{x} \in \mathbb{R}^n$.

- ist \tilde{x} ein lokales Minimum (Maximum), dann gilt $f'(\tilde{x}) = 0$ und $f''(\tilde{x})$ ist positiv (negativ) semidefinit (notwendige Bedingung)
- ist $f'(\tilde{x}) = 0$ und $f''(\tilde{x}) \neq 0$, dann gilt:
 - ist $f''(\tilde{x})$ positiv (negativ) definit, dann ist \tilde{x} ein lokales Minimum (Maximum) (hinreichende Bedingung)
 - ist $f''(\tilde{x})$ positiv (negativ) semidefinit, dann ist \tilde{x} ein lokales Minimum (Maximum) oder ein Sattelpunkt
 - ist $f''(\tilde{x})$ indefinit, dann ist \tilde{x} ein Sattelpunkt
- ist $f'(\tilde{x}) = 0$ und $f''(\tilde{x}) = 0$, dann müssen Ableitungen höherer Ordnung untersucht werden
- ist $f'(\tilde{x}) = 0$ und $f''(x)$ positiv (negativ) semidefinit für alle x in einer Umgebung von \tilde{x} , dann ist \tilde{x} ein lokales Minimum (Maximum)

Bemerkung 261. Gilt $f'(x) = 0$, so wird x auch als **stationärer Punkt** von f bezeichnet.

Beispiel 262.

- wir betrachten

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2)^2 + 3,$$

dann folgt

$$f'(x) = \begin{pmatrix} 2(x_1 - 1) \\ 2(x_2 - 2) \end{pmatrix}, \quad f''(x) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

- damit ist $f'(\tilde{x}) = 0$ genau dann wenn $\tilde{x} = (1, 2)^T$

- da $f''(\tilde{x})$ positiv definit ist, ist \tilde{x} ein lokales Minimum

Beispiel 263.

- wir betrachten

$$f(x_1, x_2) = (x_1 - 1)^2 - (x_2 - 2)^2 + 3,$$

dann folgt

$$f'(x) = \begin{pmatrix} 2(x_1 - 1) \\ -2(x_2 - 2) \end{pmatrix}, \quad f''(x) = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

- damit ist $f'(\tilde{x}) = 0$ genau dann wenn $\tilde{x} = (1, 2)^T$

- da $f''(\tilde{x})$ indefinit ist, ist \tilde{x} ein Sattelpunkt

Beispiel 264.

- wir betrachten

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2)^4 + 3,$$

dann folgt

$$f'(x) = \begin{pmatrix} 2(x_1 - 1) \\ 4(x_2 - 2)^3 \end{pmatrix}, \quad f''(x) = \begin{pmatrix} 2 & 0 \\ 0 & 12(x_2 - 2)^2 \end{pmatrix}$$

- damit ist $f'(\tilde{x}) = 0$ genau dann wenn $\tilde{x} = (1, 2)^T$

- da $f''(\tilde{x})$ positiv semidefinit in einer Umgebung von \tilde{x} ist, ist \tilde{x} ein lokales Minimum

- wir gehen ab jetzt davon aus, dass die untersuchten Funktionen f hinreichend oft differenzierbar sind
- da die Bestimmung von globalen Minima für allgemeines f extrem kompliziert ist, werden wir hier zunächst nur Verfahren untersuchen, um lokale Minima von f zu finden
- prinzipiell läuft alles auf eine Nullstellensuche für f' hinaus, wobei durch zusätzliche Maßnahmen Maxima bzw. Sattelpunkte ausgeschlossen werden sollen
- da f' im allgemeinen nichtlinear ist, sind die Verfahren wieder iterativ

11.2 Abstiegsverfahren

11.2.1 Grundlagen

- wir wollen iterative Verfahren bauen, die ausgehend von $x_0 \in \mathbb{R}^n$ eine Folge $x_k \in \mathbb{R}^n$ erzeugen mit

$$f(x_{k+1}) < f(x_k) \quad \forall k$$

- optimal wäre, wenn x_k gegen ein lokales Minimum konvergiert, d.h.

$$x_k \xrightarrow{k \rightarrow \infty} \tilde{x}$$

und damit auch

$$f'(x_k) \xrightarrow{k \rightarrow \infty} f'(\tilde{x}) = 0$$

- die Verfahren die wir hier betrachten, sollen folgende Grundstruktur haben:

- wähle zu x_k eine Suchrichtung p_k und bestimme x_{k+1} durch

$$x_{k+1} = x_k + \alpha p_k, \quad \alpha > 0$$

- berechne α so, dass

$$f(x_k) - f(x_{k+1})$$

möglichst groß ist

- stoppe die Iteration mit einem geeigneten Abbruchkriterium

- damit das Verfahren funktionieren kann, muss zur Suchrichtung p_k ein $\alpha > 0$ existieren, so dass

$$f(x_{k+1}) = f(x_k + \alpha p_k) < f(x_k)$$

ist

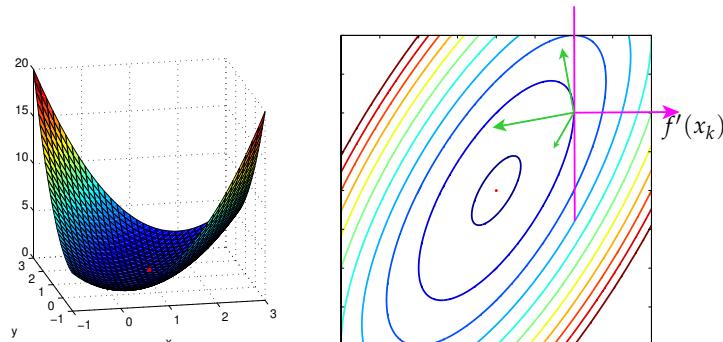
- für welche p_k ist das erfüllt?
- dazu betrachten wir die Taylorentwicklung von f um x_k

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= f'(x_k)^T (x_{k+1} - x_k) + O(\|x_{k+1} - x_k\|^2) \\ &= \alpha f'(x_k)^T p_k + O(\alpha^2) \end{aligned}$$

- für $\alpha > 0$ genügend klein dominiert der erste Term auf der rechten Seite
- damit kann die rechte Seite wie gewünscht kleiner 0 gemacht werden, falls

$$f'(x_k)^T p_k < 0,$$

d.h. der Winkel θ_k zwischen p_k und dem negativen Gradienten $-f'(x_k)$ muss in $[0, \frac{\pi}{2})$ liegen



Definition 265.

- $p \in \mathbb{R}^n$ ist **Abstiegsrichtung** von f in $x \in \mathbb{R}^n$, falls

$$f'(x)^T p < 0$$

- ein Abstiegsverfahren heißt **strikt**, falls für den Winkel θ_k zwischen p_k und dem negativen Gradienten $-f'(x_k)$

$$\theta_k \in [0, \frac{\pi}{2} - \delta) \quad \forall k$$

gilt, mit $\delta > 0$ unabhängig von k

Bemerkung 266.

- ◊ der Gradient $f'(x)$ zeigt im Punkt x in Richtung des (lokal) steilsten Anstiegs und steht immer Senkrecht auf den Höhenlinien
- ◊ $-f'(x)$ zeigt im Punkt x in Richtung des (lokal) steilsten Abstiegs
- ◊ p_k ist Abstiegsrichtung, falls es eine positive Projektion auf $-f'(x_k)$ hat ($\theta_k < \frac{\pi}{2}$)
- ◊ das Verfahren ist strikt, falls der Winkel auch für $k \rightarrow \infty$ nicht gegen $\frac{\pi}{2}$ geht

- $-f'(x)$ scheint damit zunächst die ideale Abstiegsrichtung zu sein
- aus $-f'(x)$ lassen sich einfach weitere Abstiegsrichtungen konstruieren, die später noch eine wichtige Rolle spielen werden

Satz 267. Ist $f'(x) \neq 0$, $B \in \mathbb{R}^{n \times n}$ spd, dann sind

$$p = -Bf'(x), \quad \text{bzw.} \quad p = -B^{-1}f'(x)$$

Abstiegsrichtungen in x

Beweis.

- ◊ wegen B spd und $f'(x) \neq 0$ gilt

$$f'(x)^T p = -f'(x)^T B f'(x) < 0$$

- ◊ mit B ist auch B^{-1} spd und der zweite Teil folgt analog

□

- damit haben wir genügend Möglichkeiten, Abstiegsrichtungen auszuwählen
- um einen implementierbaren Algorithmus zu erhalten, fehlt noch ein Verfahren zur Bestimmung der α (siehe Abschnitt 11.2.2) sowie ein Abbruchkriterium
- als Abbruchkriterium ideal, aber nicht praktikabel, ist die Kontrolle des Fehlers

$$\|x_k - \tilde{x}\|$$

- deswegen benutzt man folgende Kriterien bzw. Kombinationen davon

$$\|x_{k+1} - x_k\| < \varepsilon_1, \quad f(x_k) - f(x_{k+1}) < \varepsilon_2, \quad \|f(x_k)'\| < \varepsilon_3$$

- die Qualität dieser Kriterien hängt sehr stark von der Kondition des Optimierungsproblem ab, d.h. bei schlecht konditionierten Problemen können die betrachteten Größen sehr klein sein, obwohl man noch sehr weit vom wahren Minimum \tilde{x} entfernt ist

11.2.2 Liniensuche

- nachdem wir in x_k eine Abstiegsrichtung p_k gewählt haben, müssen wir entscheiden, wie weit wir absteigen, d.h. wie wir $\alpha_k > 0$ bestimmen für

$$x_{k+1} = x_k + \alpha_k p_k$$

- da wir dabei in \mathbb{R}^n entlang der Linie

$$x_k + \alpha p_k, \quad \alpha \in (0, \infty)$$

suchen, nennt man dies **Liniensuche**

- dazu definieren wir

$$\varphi(\alpha) = f(x_k + \alpha p_k)$$

d.h.

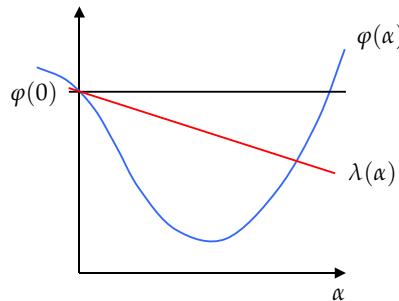
$$\varphi : \mathbb{R} \rightarrow \mathbb{R}$$

und

$$\varphi(0) = f(x_k), \quad \varphi(\alpha_k) = f(x_{k+1}), \quad \varphi'(0) = f'(x_k)^T p_k < 0$$

da p_k Abstiegsrichtung ist

- φ ist also eine einfache Funktion in \mathbb{R} , die in 0 streng monoton fallend ist
- daher gibt es auf jeden Fall ein $\alpha > 0$ mit $\varphi(\alpha) < \varphi(0) = f(x_k)$
- das Optimum an Abstieg erreichen wir, wenn wir φ auf $(0, \infty)$ minimieren, d.h. als α_k die globale Minimalstelle von φ auf $(0, \infty)$ benutzen (**exakte Liniensuche**)
- exakte Liniensuche wird selten benutzt, da sie zu aufwändig ist
- stattdessen begnügt man sich mit vereinfachten Varianten ("Soft Linesearch")



- wir minimieren $\varphi(\alpha)$ nicht, sondern verlangen nur, dass wir einen hinreichenden Abstieg erzielen
- dazu betrachten wir die Gerade

$$\lambda(\alpha) = \varphi(0) + \rho \varphi'(0) \alpha, \quad \rho \in (0, \frac{1}{2})$$

und verlangen für α_k

$$\varphi(\alpha_k) \leq \lambda(\alpha_k) \tag{11.1}$$

- (11.1) wird oft als **Armijo-Bedingung** bezeichnet
- wegen $\rho < 1$ ist die Existenz eines solchen α_k gewährleistet
- in der Praxis wird α_k z.B. durch **Backtracking** bestimmt:

- starte mit einem gegeben Wert α (oft 1)
- ist $\varphi(\alpha) \leq \lambda(\alpha)$, dann setze $\alpha_k = \alpha$
- ist $\varphi(\alpha) \geq \lambda(\alpha)$, dann ersetze α durch

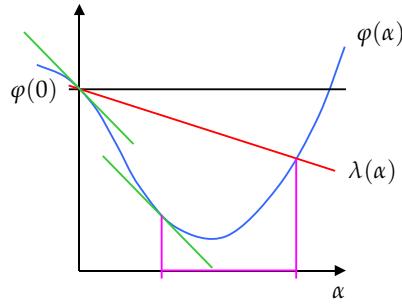
$$\tau \alpha, \quad \tau \in (0, 1)$$

und wiederhole die letzten beiden Schritte

- um unter Umständen unnötig kurze Abstiege zu verhindern (und damit den Aufwand zu reduzieren) wird neben (11.1) oft noch die Einschränkung

$$\varphi'(\alpha_k) \geq \beta \varphi'(0), \quad \rho < \beta < 1 \quad (11.2)$$

an α_k gestellt



- α_k wird also erst aus dem Bereich gewählt, in dem φ nennenswert "flacher" ist als in $\alpha = 0$
- in der Praxis bestimmt man zunächst ein Intervall $[a, b]$, das den interessanten Bereich einschließt und bestimmt dann in einem zweiten Schritt möglichst effizient α_k
- strikte Abstiegsverfahren mit einer der beiden Varianten der Liniensuche liefern stationäre Punkte von f

Satz 268. Sei $f \in C^2(\mathbb{R}^n)$, $x_0 \in \mathbb{R}^n$ und $\{x \mid f(x) \leq f(x_0)\}$ kompakt. Ein striktes Abstiegsverfahren mit Liniensuche nach (11.1) oder nach (11.1),(11.2) liefert eine Folge x_k mit

$$f(x_{k+1}) < f(x_k) \quad \forall k.$$

Entweder ist $f'(x_k) = 0$ für ein $k \in \mathbb{N}$ oder x_k besitzt mindestens einen Häufungspunkt \tilde{x} , wobei $f'(\tilde{x}) = 0$ für alle Häufungspunkte

11.2.3 Methode des steilsten Abstiegs (Steepest-Descent)

- als Beispiel für ein Abstiegsverfahren betrachten wir das **Steepest-Descent-Verfahren**
- dabei benutzen wir als Abstiegsrichtung

$$p_k = -f'(x_k)$$

kombiniert mit der Liniensuche aus dem letzten Abschnitt

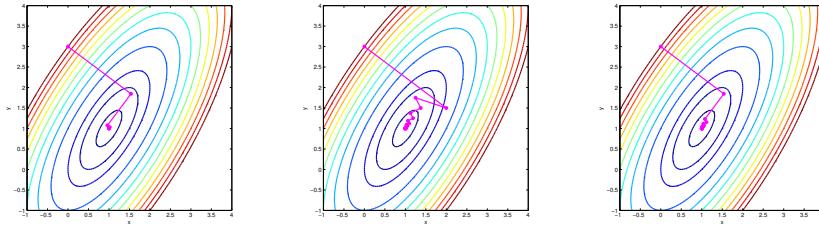
Beispiel 269.

- wir betrachten die Funktion

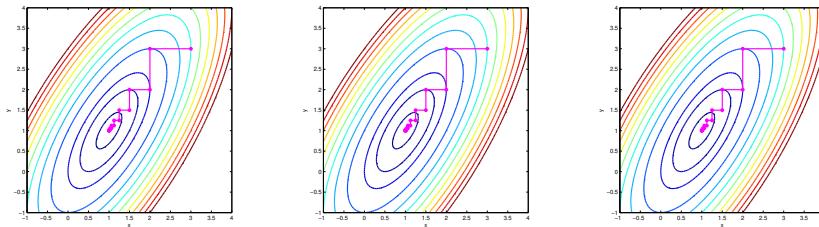
$$f(x_1, x_2) = (x_1 - x_2)^2 + (x_1 - 1)^2,$$

die ein (globales) Minimum bei $\tilde{x} = (1, 1)^T$ besitzt

- für $x_0 = (0, 3)^T$ erhalten wir folgenden Iterationsverlauf für exakte Liniensuche bzw. Liniensuche nach (11.1) und nach (11.1),(11.2)



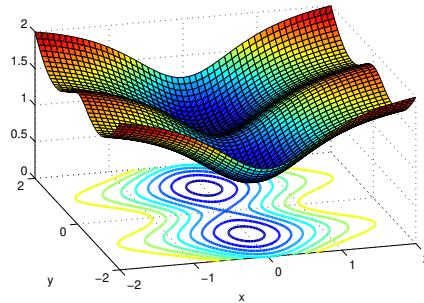
- analog ergibt sich für $x_0 = (3, 3)^T$



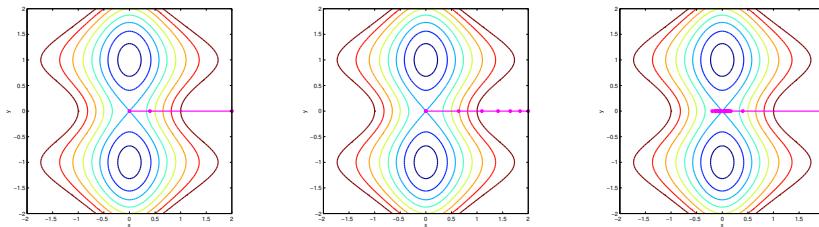
Beispiel 270.

- wir betrachten die Funktion

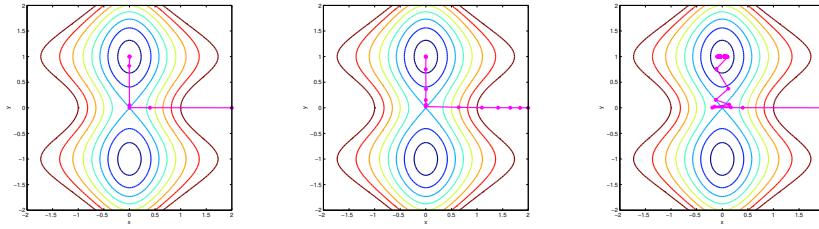
$$f(x_1, x_2) = 1 - \frac{1}{1+x_1^2} + \frac{1}{1+x_2^2} + \frac{x_2^2}{4}$$



- sie besitzt zwei globale Minima bei $\tilde{x} = (0, \pm 1)^T$ sowie einen Sattelpunkt bei $x = (0, 0)^T$
- für $x_0 = (2, 0)^T$ erhalten wir folgenden Iterationsverlauf für exakte Liniensuche bzw. Liniensuche nach (11.1) und nach (11.1),(11.2)



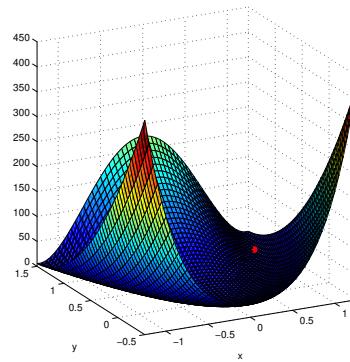
- analog ergibt sich für $x_0 = (2, 10^{-4})^T$



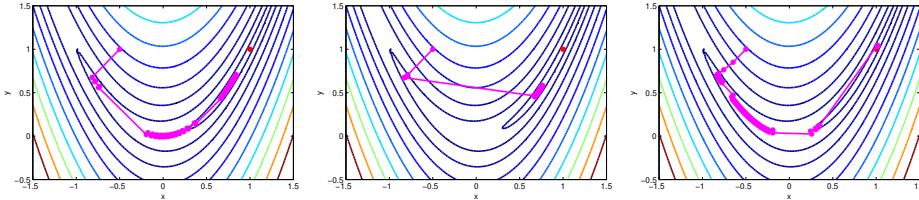
Beispiel 271.

- wir betrachten die **Rosenbrock-Funktion**

$$f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$$



- sie besitzt ein globales Minimum bei $\tilde{x} = (1, 1)^T$
- für $x_0 = (-0.5, 1)^T$ erhalten wir folgenden Iterationsverlauf für exakte Liniensuche bzw. Liniensuche nach (11.1) und nach (11.1),(11.2)



Bemerkung 272.

- ◊ für $A \in \mathbb{R}^{n \times n}$ spd, $b \in \mathbb{R}^n$ hat

$$f : \mathbb{R}^n \rightarrow \mathbb{R}, \quad f(x) = \frac{1}{2}x^T Ax - x^T b$$

ein globales Minimum bei $\tilde{x} \in \mathbb{R}^n$ mit $A\tilde{x} = b$

- ◊ als Suchrichtung für das Steepest-Descent-Verfahren erhalten wir

$$p_k = -f'(x_k) = b - Ax_k$$

- ◊ das Problem der Liniensuche kann exakt gelöst werden mit

$$\alpha_k = \frac{p_k^T p_k}{p_k^T A p_k}$$

- ◊ wir erhalten damit das selbe Verfahren wie Abschnitt 5.3.1

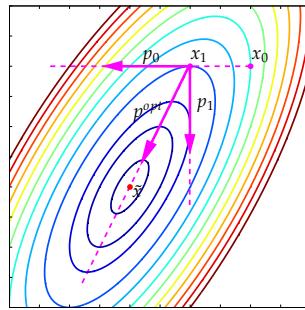
- insgesamt erhalten wir für das Steepest-Descent-Verfahren:
 - einfach zu implementieren
 - benötigt nur Auswertungen von f bzw. f'
 - relativer guter Abstieg in den ersten Iterationen
 - relativ schwach, wenn man nahe am lokalen Minimum ist
 - leichte Unterschiede bei verschiedenen Verfahren zur Liniensuche
 - keine klaren Vorteile bei exakter Liniensuche
 - wird das Verfahren mit exakter Liniensuche auf quadratische Zielfunktionen

$$f(x) = \frac{1}{2} x^T A x - x^T b + c$$

mit A spd angewandt, so kann man lineare Konvergenz beweisen

11.3 CG-Verfahren

- wie wir gesehen haben, ist das Steepest-Descent-Verfahren oft sehr langsam
- wie kann man das verbessern?
- dazu betrachten wir zwei aufeinander folgende Iterierte



- wir starten bei x_0 senkrecht zur Höhenlinie in Richtung $p_0 = -f'(x_0)$
- bei exakter Liniensuche erhalten wir x_1 genau dort, wo unsere Linie tangential zu einer Höhenlinie verläuft
- in x_1 starten wir erneut senkrecht zur Höhenlinie in Richtung $p_1 = -f'(x_1)$
- damit ist $p_1 \perp p_0$
- p_1 zeigt in der Regel *nicht* von x_1 auf das Minimum \tilde{x}
- $p^{opt} = \tilde{x} - x_1$ wäre die optimale Suchrichtung
- Idee:

- bestimme neue Abstiegsrichtung als Linearkombination von p_0 und p_1

$$\hat{p}_1 = p_1 + \gamma p_0$$

- bestimme γ so, dass \hat{p}_1 näher an $p^{opt} = \tilde{x} - x_1$ ist
- wie soll man γ wählen?
- dazu betrachten wir zunächst quadratische Funktionen f , leiten ein Kriterium ab und wenden es später auch für andere f an
- es sei also $A \in \mathbb{R}^{n \times n}$ spd, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$ und $f : \mathbb{R}^n \rightarrow \mathbb{R}$,

$$f(x) = \frac{1}{2}x^T A x - x^T b + c \quad (11.3)$$

- damit ist $f'(x) = Ax - b$ und f hat ein globales Minimum bei $\tilde{x} = A^{-1}b$
- für die Abstiegsrichtung beim Steepest-Descent-Verfahren erhalten wir

$$p_k = -f'(x_k) = b - Ax_k$$

- aus $p_1 \perp p_0$ folgt

$$0 = p_0^T p_1 = p_0^T (b - Ax_1) = p_0^T (A\tilde{x} - Ax_1) = p_0^T A (\tilde{x} - x_1) = p_0^T A p^{opt}$$

- wir bestimmen jetzt γ so, dass $\hat{p}_1 = p_1 + \gamma p_0$ diese Eigenschaft ebenfalls erfüllt

$$0 = p_0^T A \hat{p}_1 = p_0^T A (p_1 + \gamma p_0),$$

also

$$\gamma = -\frac{p_0^T A p_1}{p_0^T A p_0} \quad (11.4)$$

- diesen Zusammenhang wollen wir nicht nur für den ersten Schritt herstellen, sondern auch für alle folgenden Schritte, d.h.

$$\hat{p}_i^T A \hat{p}_j = 0 \quad \forall i \neq j$$

Definition 273. Sei $A \in \mathbb{R}^{n \times n}$ spd. Die Vektoren $p_i \in \mathbb{R}^n$ heißen **A-konjugiert** falls

$$p_i^T A p_j = 0 \quad \forall i \neq j.$$

- wir bauen also ein neues Verfahren mit modifizierten Steepest-Descent-Suchrichtungen auf
- wenn das Verfahren auf quadratische Funktionen vom Typ (11.3) angewandt wird, dann sind die Suchrichtungen A-konjugiert
- wir erhalten daraus das (nichtlineare) **CG-Verfahren**:

- x_0 gegeben, $p_0 = p_0^{SD} = -f'(x_0)$
- wiederhole für $k = 0, 1, 2, \dots$

$$f'(x_k)^T p_k \geq 0 \quad \Rightarrow \quad p_k = p_k^{SD}$$

$$\alpha_k = \text{linesearch}(x_k, p_k)$$

$$x_{k+1} = x_k + \alpha_k p_k$$

$$p_{k+1}^{SD} = -f'(x_{k+1})$$

γ_{k+1} berechnen

$$p_{k+1} = p_{k+1}^{SD} + \gamma_{k+1} p_k$$

- zur Berechnung der γ_k können verschiedene Formeln angegeben werden, die für quadratisches f A -konjugierte Richtungen liefern:

$$\gamma_{k+1} = \frac{f'(x_{k+1})^T f'(x_{k+1})}{f'(x_k)^T f'(x_k)} \quad \text{Fletcher-Reeves}$$

$$\gamma_{k+1} = \frac{f'(x_{k+1})^T (f'(x_{k+1}) - f'(x_k))}{f'(x_k)^T f'(x_k)} \quad \text{Polak-Ribi re}$$

Satz 274. Sei f wie in (11.3) quadratisch. Bei exakter Liniensuche und $f'(x_k) \neq 0 \forall k$ gilt für die Fletcher-Reeves- und Polak-Ribi re-Version des nichtlinearen CG-Verfahrens:

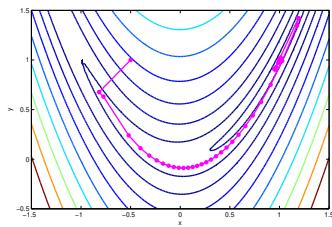
- die p_k sind A -konjugiert
- alle p_k sind Abstiegsrichtungen
- der Algorithmus stoppt nach sp testens n Schritten

Bemerkung 275.

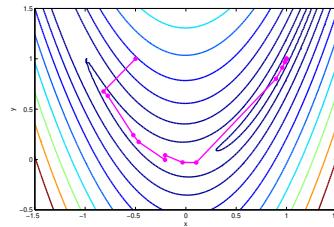
- bei quadratischem f (wie in (11.3)) und exakter Liniensuche erhalten wir mit beiden Varianten das klassische CG-Verfahren aus Abschnitt 5.3.2
- für Fletcher-Reeves kann man (unter einigen Einschr nkungen) auch bei soft linesearch ein  hnliches Ergebnis wie im letzten Satz beweisen
- bei Polak-Ribi re ist das nicht m glich (Gegenbeispiele)
- Praxis: Polak-Ribi re ist oft g nstiger als Fletcher-Reeves

Beispiel 276.

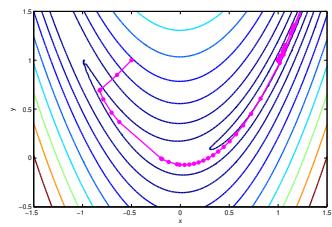
- ▶ wir betrachten nochmal die Rosenbrock-Funktion aus Beispiel 271
- ▶ f r $x_0 = (-0.5, 1)^T$ bzw. $x_0 = (-1.2, 1)^T$ wenden wir das Fletcher-Reeves- bzw. Polak-Ribi re-CG mit exakter bzw. 'soft' Liniensuche an



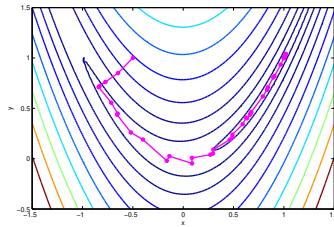
Fletcher-Reeves, exakt



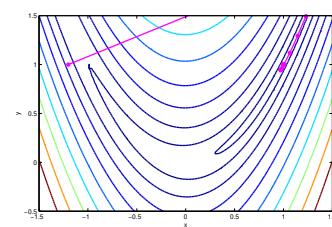
Polak-Ribière, exakt



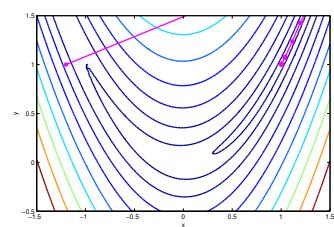
Fletcher-Reeves, soft



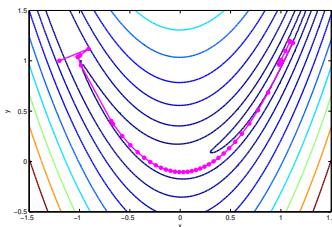
Polak-Ribière, soft



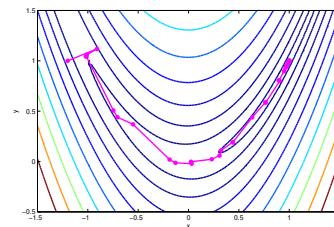
Fletcher-Reeves, exakt



Polak-Ribière, exakt



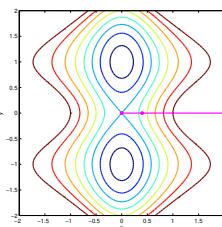
Fletcher-Reeves, soft



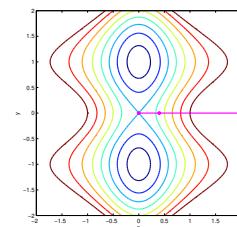
Polak-Ribière, soft

Beispiel 277.

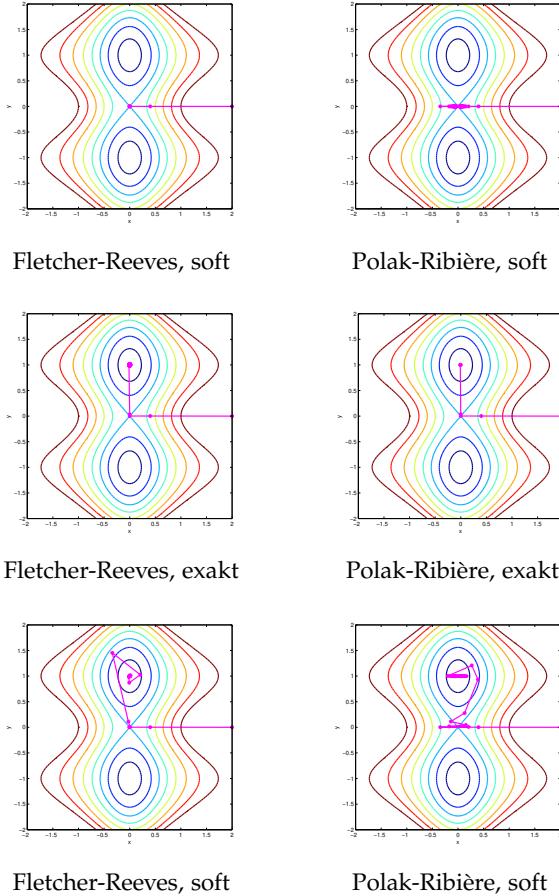
- ▶ wir betrachten nochmal Beispiel 270
- ▶ für $x_0 = (2, 0)^T$ bzw. $x_0 = (2, 10^{-4})^T$ wenden wir das Fletcher-Reeves- bzw. Polak-Ribière-CG mit exakter bzw. 'soft' Liniensuche an



Fletcher-Reeves, exakt



Polak-Ribière, exakt



- insgesamt erhalten wir für das CG-Verfahren:
 - einfach zu implementieren
 - benötigt nur Auswertungen von f bzw. f'
 - relativer guter Abstieg, auch nahe am lokalen Minimum
 - sensibel gegenüber verschiedenen Verfahren zur Liniensuche
 - wird das Verfahren mit exakter Liniensuche auf quadratische Zielfunktionen

$$f(x) = \frac{1}{2}x^T Ax - x^T b + c$$

mit A spd angewandt, so kann man quadratische Konvergenz beweisen

11.4 Newton-Verfahren

11.4.1 Grundlagen

- lokale Minima von f sind stationäre Punkte, d.h. $f'(x) = 0$
- erste Herleitung:
 - wir bestimmen zunächst stationäre Punkte und überprüfen dann, ob es sich um Minima handelt
 - damit erhalten wir ein Nullstellenproblem für

$$g : \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad g(x) = f'(x)$$

- zur Lösung wenden wir das Newton-Verfahren auf g an:

- wir nähern g in einer Umgebung von x_k durch eine lineare Funktion (Taylor-Entwicklung) an

$$g(x) \approx l_k(x) = g(x_k) + g'(x_k)(x - x_k)$$

- als neue Näherung x_{k+1} benutzen wir eine Nullstelle von l_k , d.h.

$$0 = l_k(x_{k+1}) = g(x_k) + g'(x_k)(x_{k+1} - x_k)$$

bzw.

$$g'(x_k)(x_{k+1} - x_k) = -g(x_k)$$

- ist die Jacobi-Matrix $g'(x_k) \in \mathbb{R}^{n \times n}$ invertierbar, so folgt

$$x_{k+1} = x_k - g'(x_k)^{-1}g(x_k)$$

bzw.

$$x_{k+1} = x_k - f''(x_k)^{-1}f'(x_k)$$

- zweite Herleitung:

- wir nähern f in einer Umgebung von x_k durch eine quadratische Funktion (Taylor-Entwicklung) an

$$f(x) \approx q_k(x) = f(x_k) + f'(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T f''(x_k)(x - x_k)$$

- als neue Näherung x_{k+1} benutzen wir einen Minimierer von q_k

- ist $f''(x_k)$ spd, so gibt es genau einen Minimierer, nämlich die eindeutige Lösung von $q'_k(x) = 0$, d.h.

$$0 = q'_k(x_{k+1}) = f'(x_k) + f''(x_k)(x_{k+1} - x_k),$$

also wieder

$$x_{k+1} = x_k - f''(x_k)^{-1}f'(x_k)$$

- die Konvergenzresultate für das Newton-Verfahren für skalare Funktionen lassen sich analog erweitern

Satz 278. Sei \tilde{x} ein lokales Minimum von f mit $f''(\tilde{x})$ spd und x_0 hinreichend nahe an \tilde{x} . Dann ist das Newton-Verfahren durchführbar und es konvergiert quadratisch gegen \tilde{x} .

- Vorteile:

- quadratische Konvergenz
- einfache Implementierung

- Nachteile:

- lokale Konvergenz (Steepest-Descent: global)
- wenn $f''(\tilde{x})$ singulär ist kann die Iteration (zunächst) nicht fortgesetzt werden
- benötigt in jedem Schritt $f''(x_k)$, d.h. Auswertung von n^2 partiellen Ableitungen zweiter Ordnung

- das Newton-Verfahren wurde aus lokalen Modellen $l_k(x)$ bzw. $q_k(x)$ hergeleitet, die beide aus einer Taylor-Entwicklung gewonnen wurden
- diese Modelle sind nur lokal um den Entwicklungspunkt x_k gute Approximationen (d.h. für $\|x - x_k\|$ klein), was zum Teil die lokale Konvergenz erklärt
- wie kann man das Newton-Verfahren so modifizieren, dass

- singuläres $f''(x_k)$ kein Problem mehr ist
- sichergestellt ist, dass die lokalen Modelle sinnvolle Approximationen sind, also $\|x - x_k\|$ klein genug bleibt und damit eventuell globale Konvergenz erreicht wird, ohne dass die quadratische Konvergenz verloren geht
- $f''(x_k)$ nicht ständig berechnet werden muss
- der erste Punkt führt zu **gedämpften Newton-Verfahren**:
 - wir versuchen, die gute Konvergenzrate des Newton-Verfahrens mit der globalen Konvergenz des Steepest-Descent-Verfahrens zu kombinieren
 - es ist

$$x_{k+1} = x_k - f''(x_k)^{-1} f'(x_k)$$
 - ist $f''(x_k)$ spd, so ist nach Satz 267

$$p_k = -f''(x_k)^{-1} f'(x_k)$$

eine Abstiegsrichtung und das Newton-Verfahren ein Abstiegsverfahren

$$x_{k+1} = x_k + \alpha_k p_k, \quad \alpha_k = 1$$

- statt $\alpha_k = 1$ könnte man α_k durch Liniensuche bestimmen und damit versuchen, globale Konvergenz zu erreichen (was unter einigen Einschränkungen auch funktioniert)
- allerdings muss dazu $f''(x_k)$ nach wie vor spd sein
- ist $f''(x_k)$ nicht spd, aber invertierbar, so kann die Situation wie folgt gerettet werden:
 - für $f'(x_k) \neq 0$ ist $p_k \neq 0$
 - ist p_k keine Abstiegsrichtung, dann ist $-p_k$ eine Abstiegsrichtung
- für singuläres $f''(x_k)$ kommt man damit aber auch nicht weiter
- andererseits wissen wir aber folgendes:
 - $f''(x_k)$ ist die Hesse-Matrix von f und damit symmetrisch
 - alle Eigenwerte von $f''(x_k)$ sind reell
 - ist $f''(x_k)$ nicht spd, dann ist

$$\lambda_{\min}(f''(x_k)) \leq 0$$

und für

$$\mu_k > -\lambda_{\min}(f''(x_k)) \geq 0$$

ist $f''(x_k) + \mu_k I$ spd

Definition 279. Eine Iteration des **gedämpften Newton-Verfahrens** ist definiert durch:

- wähle $\mu_k \in \mathbb{R}$ groß genug, s.d. $f''(x_k) + \mu_k I$ spd ist
- $p_k = -(f''(x_k) + \mu_k I)^{-1} f'(x_k)$
- $x_{k+1} = x_k + \alpha_k p_k$

wobei $\alpha_k = 1$ benutzt wird oder $\alpha_k > 0$ durch Liniensuche bestimmt wird.

Bemerkung 280.

- ◊ ein Schritt des ungedämpften Newton-Verfahrens war für $f''(x_k)$ spd äquivalent zur Minimierung des lokalen quadratischen Modells

$$q_k(x) = f(x_k) + f'(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T f''(x_k)(x - x_k)$$

- ◊ ein Schritt des gedämpften Newton-Verfahrens ist äquivalent zur Minimierung des modifizierten Modells

$$\tilde{q}_k(x) = q_k(x) + \frac{1}{2}\mu_k(x - x_k)^T(x - x_k),$$

das bei geeignet gewähltem μ_k immer ein eindeutiges Minimum besitzt

- ◊ der neue Term

$$\frac{1}{2}\mu_k(x - x_k)^T(x - x_k) = \frac{1}{2}\mu_k\|x - x_k\|_2^2$$

ist ein **Strafterm**, der zu große Werte von $\|x_{k+1} - x_k\|_2$ verhindert

- ◊ für große μ_k ($\mu_k \rightarrow \infty$) folgt

$$p_k = -(f''(x_k) + \mu_k I)^{-1}f'(x_k) = -\frac{1}{\mu_k}(\frac{1}{\mu_k}f''(x_k) + I)^{-1}f'(x_k) \approx -\frac{1}{\mu_k}f'(x_k),$$

d.h. wir erhalten die Abstiegsrichtung des Steepest-Descent-Verfahrens (mit einem sehr kurzen Schritt $\alpha_k = \frac{1}{\mu_k}$)

- wie wählt man μ_k ?

- λ_i seien die Eigenwerte von $f''(x_k)$
- wähle μ_k so, dass

$$\lambda_{\min} + \mu_k > 0$$

und

$$\frac{\lambda_{\max} + \mu_k}{\lambda_{\min} + \mu_k}$$

nicht zu groß ist, damit die Kondition des linearen Gleichungssystems

$$(f''(x_k) + \mu_k I)p_k = -f'(x_k)$$

nicht zu schlecht ist

- in der Praxis setzt man verschiedene Methoden ein, um μ_k zu bestimmen
- Cholesky:
 - starte mit $\mu > 0$ und führe eine Cholesky-Zerlegung von $f''(x_k) + \mu I$ durch
 - teste dabei die Größe der Diagonalelemente vor dem Ziehen der Wurzel
 - ist ein Diagonalelement negativ, dann ist die Matrix nicht spd
 - setze $\mu := 2\mu$ und starte von vorne
- Abschätzung der Eigenwerte von $f''(x_k)$:
 - berechne (möglichst billig, z.B. mit dem **Satz von Gerschgorin**) eine untere Schranke c für die Eigenwerte von $f''(x_k)$
 - wähle $\mu_k > c$
- auf die Wahl der μ_k wird später nochmal genauer eingegangen

11.4.2 Trust-Region-Verfahren

- das ungedämpfte Newton-Verfahren konnte über das lokal quadratische Modell

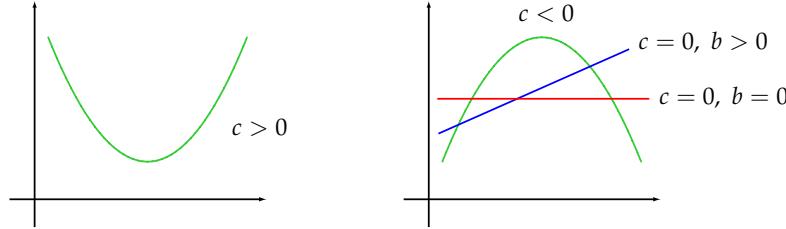
$$q_k(x) = f(x_k) + f'(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T f''(x_k)(x - x_k)$$

hergeleitet werden

- ist $f''(x_k)$ spd, dann ist x_{k+1} das eindeutige (globale) Minimum von q_k über ganz \mathbb{R}^n
- ist $f''(x_k)$ nicht spd, so kann q_k unendlich viele bzw. gar keine Minimierer besitzen:
 - im Fall $n = 1$ gilt

$$q_k(x) = a + bx + \frac{1}{2}cx^2, \quad a, b, c \in \mathbb{R}$$

mit $c = f''(x_k)$



- für $c \leq 0$ (d.h. $f''(x_k)$ nicht spd) erhalten wir kein eindeutiges Minimum:
 - $c = 0, b = 0$ liefert unendlich viele Minima
 - $c < 0$ bzw. $c = 0, b > 0$ liefert gar kein Minimum, da

$$\lim_{x \rightarrow -\infty} q_k(x) = -\infty$$

- andererseits macht es wenig Sinn, globale Minima von q_k auf ganz \mathbb{R}^n zu suchen:
 - $q_k(x)$ ist eine lokale Approximation von $f(x)$
 - die Approximation taugt nur in einer (eventuell sehr kleinen) Umgebung von x_k was
 - Minima von $q_k(x)$ die weit von x_k weg liegen haben in der Regel mit dem tatsächlichen Verhalten von f wenig zu tun
- deshalb minimieren wir jetzt $q_k(x)$ nicht global über ganz \mathbb{R}^n , sondern nur lokal in

$$\Omega_k = \{x \mid \|x - x_k\| \leq r_k\}, \quad r_k > 0$$

- Ω_k heißt **Trust-Region (Vertrauensbereich)**, r_k **Vertrauensradius**
- da q_k stetig auf Ω_k ist und Ω_k kompakt ist, gibt es mindestens eine Lösung des Problems

$$\min_{x \in \Omega_k} q_k(x)$$

- wie berechnet man solche Probleme möglichst einfach?

Satz 281. Ist $f''(x_k) + \mu I$ spd und x_μ die eindeutige Lösung von

$$(f''(x_k) + \mu I)(x_\mu - x_k) = -f'(x_k)$$

dann ist x_μ auch Lösung von

$$\min_{\|x - x_k\|_2 \leq r_\mu} q_k(x), \quad r_\mu = \|x_\mu - x_k\|_2.$$

Außerdem gilt $r_\mu \xrightarrow{\mu \rightarrow \infty} 0$.

Bemerkung 282.

- ◊ x_μ ist das Ergebnis eines gedämpften Newton-Schritts mit Parameter μ
- ◊ x_μ löst das restriktierte Minimierungsproblem für q_k mit Vertrauensradius r_μ der von μ abhängt
- ◊ durch Vergrößern von μ kann der Vertrauensradius r_μ beliebig klein gemacht werden
- ◊ die Größe des Vertrauensbereichs kann also über den Parameter μ des gedämpften Newton-Verfahrens kontrolliert werden

Beweis.

- ◊ wir betrachten

$$q_k(x) = f(x_k) + f'(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T f''(x_k)(x - x_k)$$

bzw.

$$\begin{aligned} \tilde{q}_k(x) &= q_k(x) + \frac{1}{2}\mu(x - x_k)^T(x - x_k) \\ &= f(x_k) + f'(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T(f''(x_k) + \mu I)(x - x_k) \end{aligned}$$

- ◊ wegen

$$\tilde{q}'_k(x) = f'(x_k) + (f''(x_k) + \mu I)(x - x_k)$$

ist $\tilde{q}'_k(x_\mu) = 0$, d.h. x_μ ist stationärer Punkt von $\tilde{q}_k(x)$

- ◊ da $f''(x_k) + \mu I$ spd ist, ist x_μ damit das globale Minimum von $\tilde{q}_k(x)$

- ◊ ist nun \hat{x} Lösung von

$$\min_{\|x - x_k\|_2 \leq r_\mu} q_k(x),$$

so erhalten wir

$$\begin{aligned} \tilde{q}_k(\hat{x}) &= q_k(\hat{x}) + \frac{1}{2}\mu(\hat{x} - x_k)^T(\hat{x} - x_k) \\ &= q_k(\hat{x}) + \frac{1}{2}\mu\|\hat{x} - x_k\|_2^2 \\ &\leq q_k(\hat{x}) + \frac{1}{2}\mu r_\mu^2 \\ &\leq q_k(x_\mu) + \frac{1}{2}\mu\|x_\mu - x_k\|_2^2 \\ &= \tilde{q}_k(x_\mu) \end{aligned}$$

- ◊ da x_μ das eindeutige globale Minimum von \tilde{q}_k ist, muss damit $\hat{x} = x_\mu$ folgen

- ◊ zum Beweis der asymptotischen Eigenschaften von r_μ betrachten wir

$$(f''(x_k) + \mu I)(x_\mu - x_k) = -f'(x_k)$$

bzw.

$$\left(\frac{1}{\mu}f''(x_k) + I\right)(x_\mu - x_k) = -\frac{1}{\mu}f'(x_k)$$

- ◊ daraus folgt

$$r_\mu = \|x_\mu - x_k\|_2 = \frac{1}{\mu} \left\| \left(\frac{1}{\mu}f''(x_k) + I\right)^{-1} f'(x_k) \right\|_2 \xrightarrow{\mu \rightarrow \infty} 0$$

□

- damit sieht unser Trust-Region-Verfahren wie folgt aus:
 - betreibe gedämpftes Newton-Verfahren mit Parameter μ_k
 - wähle μ_k so groß, dass $f''(x_k) + \mu_k I$ spd ist und x_{k+1} im Vertrauensbereich liegt, d.h. in einem Bereich wo q_k ein brauchbares Modell für f ist
 - wie stellt man fest, ob q_k ein brauchbares Modell für f ist?
 - wir betrachten dazu den **Gain-Faktor**
- $$\rho_k = \frac{f(x_k) - f(x_{k+1})}{q_k(x_k) - q_k(x_{k+1})},$$
- der den tatsächlichen Abstieg $f(x_k) - f(x_{k+1})$ mit dem durch das quadratische Modell q_k vorhergesagten Abstieg $q_k(x_k) - q_k(x_{k+1})$ in Relation setzt
- $\rho_k \leq 0$:
 - wegen $f''(x_k) + \mu_k I$ spd gilt für $f'(x_k) \neq 0$ immer $q_k(x_k) - q_k(x_{k+1}) > 0$
 - damit ist also $f(x_k) - f(x_{k+1}) \leq 0$, d.h. wir haben keinen Abstieg
 - an der Stelle x_{k+1} ist q_k ist also ein ganz schlechtes Modell für f
 - wir werfen x_{k+1} weg und verkleinern den Vertrauensbereich indem μ_k vergrößert wird und berechnen damit ein neues x_{k+1}
 - $0 < \rho_k \ll 1$:
 - der tatsächliche Abstieg ist sehr gering
 - an der Stelle x_{k+1} ist q_k kein gutes Modell für f
 - wir behalten x_{k+1} , da ja ein Abstieg erzielt wurde
 - für den nächsten Schritt verkleinern wir den Vertrauensbereich, indem wir $\mu_{k+1} > \mu_k$ verwenden
 - $\rho_k \approx 1$:
 - an der Stelle x_{k+1} ist q_k ein gutes Modell für f , da vorhergesagter und tatsächlicher Abstieg fast identisch sind
 - im nächsten Schritt vergrößern wir den Vertrauensbereich, indem wir $\mu_{k+1} < \mu_k$ verwenden
 - $\rho_k \gg 1$:
 - an der Stelle x_{k+1} ist q_k ein schlechtes Modell für f , aber der tatsächliche Abstieg ist viel besser als der vorhergesagte
 - deshalb verfahren wir wie im Fall $\rho_k \approx 1$
 - in der Praxis benutzt man z.B.

$$\mu_{\text{neu}} = \begin{cases} 2\mu & \rho < \frac{1}{4} \\ \frac{\mu}{3} & \rho > \frac{3}{4} \\ \mu & \text{sonst} \end{cases}$$

oder

$$\mu_{\text{neu}} = \begin{cases} 2\mu & \rho \leq 0 \\ \mu \max\left(\frac{1}{3}, 1 - (2\rho - 1)^3\right) & \rho > 0 \end{cases}$$

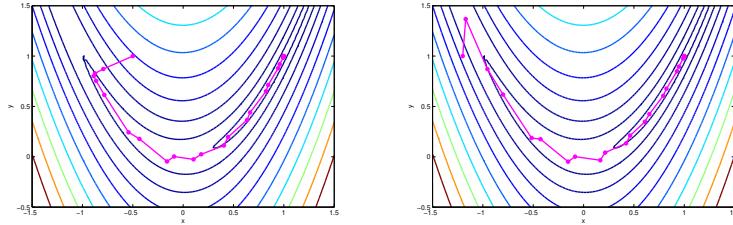
- durch die Faktoren 2, $\frac{1}{3}$ wird eine Stagnation durch alternierendes Vergrößern bzw. Verkleinern vermieden
- die zweite Variante ändert μ "kontinuierlich" und hat in der Praxis oft Vorteile
- insgesamt erhalten wir damit ein gedämpftes Newton-Verfahren vom **Levenberg-Marquardt-Typ**:

- starte mit $x := x_0$, $\mu := \mu_0$, x_0, μ_0 gegeben
- wiederhole bis Genauigkeit erreicht ist:
 - wiederhole bis $f''(x) + \mu I$ spd ist:
 - $\mu := 2\mu$
 - löse $(f''(x) + \mu I)p = -f'(x)$
 - $x_{\text{neu}} := x + p$ (oder $x_{\text{neu}} := x + \alpha p$ bei Liniensuche)
 - berechne

$$\rho := \frac{f(x) - f(x_{\text{neu}})}{q(x) - q(x_{\text{neu}})}$$
 - mit $q(y) = f(x) + f'(x)^T(y - x) + \frac{1}{2}(y - x)^T f''(x)(y - x)$
 - ist $\rho > \delta > 0$:
 - $x := x_{\text{neu}}$
 - $\mu := \mu \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$
 - ist $\rho \leq \delta$:
 - x_{neu} wegwerfen (aktuellen Schritt wiederholen)
 - $\mu := 2\mu$

Beispiel 283.

- wir betrachten nochmal die Rosenbrock-Funktion aus Beispiel 271
- für $x_0 = (-0.5, 1)^T$ bzw. $x_0 = (-1.2, 1)^T$ wenden wir das gedämpfte Newton-Verfahren vom Levenberg-Marquardt-Typ an (ohne Liniensuche, $\mu_0 = 1$, $\delta = 10^{-3}$)



- Vorteile:
 - globale Konvergenz (unter einigen Einschränkungen an f)
 - superlineare Konvergenz
 - einfach zu implementieren
- Nachteil:
 - Berechnung von $f''(x)$ in jedem Schritt

11.4.3 Quasi-Newton-Verfahren

- im letzten Abschnitt haben wir $f''(x)$ aus dem Original-Newton-Verfahren durch $f''(x) + \mu I$ ersetzt
- machen noch andere Modifikationen von f'' Sinn?
- wir betrachten das gedämpfte Newton-Verfahren vom Levenberg-Marquardt-Typ und ersetzen $f''(x)$ durch die 0-Matrix:

- das bedeutet, dass das quadratische Modell

$$q(y) = f(x) + f'(x)^T(y - x) + \frac{1}{2}(y - x)^T f''(x)(y - x)$$

abgeändert wird zu

$$q(y) = f(x) + f'(x)^T(y - x),$$

also nur noch ein lineares Modell für f ist

- ist $\mu > 0$ dann ist μI spd und

$$p = -\frac{1}{\mu}f'(x)$$

- benutzt man $x_{\text{neu}} = x + p$ so erhalten wir

$$x_{\text{neu}} = x - \frac{1}{\mu}f'(x)$$

also gerade das Steepest-Descent-Verfahren mit Parameter $\alpha = \frac{1}{\mu}$

- die Veränderung von μ anhand des Gain-Faktors ρ wirkt dann wie eine "softe" Linien-suche
- offensichtlich braucht man $f''(x)$ nicht sehr genau zu kennen, um ein konvergentes Verfahren zu erhalten
- wichtig ist, dass die Näherung von $f''(x)$ spd ist
- ist die Näherung gut, dann wird man damit die Konvergenzgeschwindigkeit erhöhen
- Idee:

- betrachte Standard-Newton (mit Liniensuche)

$$f''(x_k)p_k = -f'(x_k), \quad x_{k+1} = x_k + \alpha_k p_k$$

- ersetze $f''(x_k)$ durch eine Näherung $B_k \in \mathbb{R}^{n \times n}$, die am besten spd ist
- berechne damit die neue Näherung x_{k+1}

$$B_k p_k = -f'(x_k), \quad x_{k+1} = x_k + \alpha_k p_k$$

- führe anschließend ein sinnvolles Update

$$B_k \rightarrow B_{k+1}$$

durch

- solche Verfahren nennt man **Quasi-Newton-Verfahren**
- wie berechnet man ein Update für B_k ?
- dazu betrachten wir das Newton-Verfahren nochmal als Verfahren zur Nullstellensuche für $g(x) = f'(x)$
- für $\alpha_k = 1$ erhalten wir im eindimensionalen Fall

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}$$

- vermeiden von $f''(x_k)$ bedeutet, dass wir das Verfahren so abändern müssen, dass $g'(x_k)$ nicht mehr berechnet werden muss

- das trifft z.B. auf das **Sekanten-Verfahren** zu

$$x_{k+1} = x_k - \frac{g(x_k)}{\frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}}},$$

das superlinear konvergent ist (siehe Satz 145)

- hier wird die Ableitung durch einen Differenzenquotienten angenähert, d.h.

$$g'(x_k) \approx \frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}}$$

bzw. (nach einem Index-Shift um 1)

$$g'(x_{k+1})(x_{k+1} - x_k) \approx g(x_{k+1}) - g(x_k)$$

und damit

$$f''(x_{k+1})(x_{k+1} - x_k) \approx f'(x_{k+1}) - f'(x_k)$$

- das Update B_{k+1} soll diese Bedingung exakt erfüllen

Definition 284. Es seien B_k Approximationen von $f''(x_k)$, $k \in \mathbb{N}$. B_k erfüllt die **Quasi-Newton-Bedingung**, falls

$$B_{k+1}h_k = y_k, \quad h_k = x_{k+1} - x_k, \quad y_k = f'(x_{k+1}) - f'(x_k)$$

- spalten wir das Update auf in

$$B_{k+1} = B_k + W, \quad W \in \mathbb{R}^{n \times n}$$

so liefert die Quasi-Newton-Bedingung

$$Wh_k = y_k - B_k h_k,$$

- h_k und die rechte Seite ist gegeben, d.h. wir erhalten n lineare Gleichungen für die n^2 Koeffizienten von W
- wir brauchen also zusätzliche Einschränkungen, um W eindeutig fest zu legen
- erste Idee:

 - benutze für W eine einfache Matrix mit Rang 1, d.h.

$$W = ab^T, \quad a, b \in \mathbb{R}^n,$$

d.h. wir haben $2n$ freie Parameter

- neben der Quasi-Newton-Bedingung verlangen wir, dass

$$Wv = 0 \quad \forall v \perp h_k,$$

d.h. B_{k+1} unterscheidet sich von B_k nur "in Richtung h_k "

- damit erhalten wir **Broyden-Rang-1-Updates**

$$B_{k+1} = B_k + \frac{(y_k - B_k h_k)h_k^T}{h_k^T h_k}$$

- man startet meistens mit $B_0 = I$

- in der Regel sind die B_{k+1} nicht spd
- Problem kann beseitigt werden, indem für W geeignete Matrizen mit Rang 2 benutzt werden
- die erfolgreichste Update-Variante ist die **BFGS-Update-Formel** (Broyden, Fletcher, Goldfarb, Shanno)

$$B_{k+1} = B_k + \frac{y_k y_k^T}{h_k^T y_k} - \frac{u_k u_k^T}{h_k^T u_k}$$

mit

$$h_k = x_{k+1} - x_k, \quad y_k = f'(x_{k+1}) - f'(x_k), \quad u_k = B_k h_k$$

- für $h_k^T y_k > 0$ ist B_{k+1} spd, falls B_k spd ist ($h_k^T u_k = h_k^T B_k h_k > 0$, da B_k spd)
- damit erhalten wir insgesamt als BFGS-Quasi-Newton-Verfahren:
 - starte mit $x_0, B_0 = I$
 - wiederhole:
 - löse $B_k p_k = -f'(x_k)$
 - bestimme $x_{k+1} = x_k + \alpha_k p_k$ mit Liniensuche
 - berechne Update $B_{k+1} = B_k + \frac{y_k y_k^T}{h_k^T y_k} - \frac{u_k u_k^T}{h_k^T u_k}$
- in jedem Schritt muss das lineare Gleichungssystem $B_k p_k = -f'(x_k)$ gelöst werden, was einen Aufwand $O(n^3)$ verursacht
- man kann die Update-Formeln auch direkt für $D_k = B_k^{-1}$ herleiten so dass p_k dann direkt als

$$p_k = -D_k f'(x_k),$$

berechnet werden kann, wofür nur noch $O(n^2)$ Operationen nötig sind

- für **BFGS** erhält man

$$D_{k+1} = D_k + \varkappa_1 h_k h_k^T - \varkappa_2 (h_k v_k^T + v_k h_k^T)$$

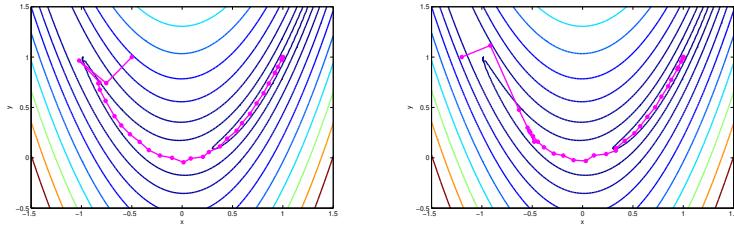
mit

$$\begin{aligned} h_k &= x_{k+1} - x_k \\ y_k &= f'(x_{k+1}) - f'(x_k) \\ v_k &= D_k y_k \\ \varkappa_2 &= \frac{1}{h_k^T y_k} \\ \varkappa_1 &= \varkappa_2 (1 + \varkappa_2 y_k^T v_k) \end{aligned}$$

- insgesamt erhalten wir damit Verfahren mit folgenden Eigenschaften:
 - globale Konvergenz (unter einigen Einschränkungen an f)
 - superlineare Konvergenz
 - einfach zu implementieren
 - Berechnung von $f''(x)$ nicht erforderlich

Beispiel 285.

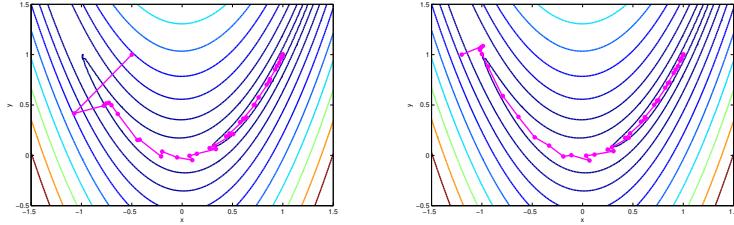
- wir betrachten die Rosenbrock-Funktion aus Beispiel 271
- für $x_0 = (-0.5, 1)^T$ bzw. $x_0 = (-1.2, 1)^T$ wenden wir das Quasi-Newton-Verfahren mit der BFGS-Update-Formel für die D_k an



- BFGS-Updates lassen sich auch in das gedämpfte Newton-Verfahren vom **Levenberg-Marquardt-Typ** einbauen
- da die B_k immer spd sind, ist $B_k + \mu I$ für $\mu \geq 0$ immer spd und die zweite Schleife kann entfallen

Beispiel 286.

- wir betrachten die Rosenbrock-Funktion aus Beispiel 271
- für $x_0 = (-0.5, 1)^T$ bzw. $x_0 = (-1.2, 1)^T$ wenden wir das gedämpfte Newton-Verfahren vom Levenberg-Marquardt-Typ mit BFGS-Updates an



11.5 Nichtlineare Ausgleichsprobleme

- in Abschnitt 9.1 hatten wir lineare Ausgleichsprobleme untersucht:
 - zu gegebenen Punkten $(x_i, y_i) \in \mathbb{R}^2$, $i = 1, \dots, n$, soll ein Polynom p vom Grad $m - 1$ mit Koeffizienten p_1, \dots, p_m bestimmt werden, so dass die Summe der Fehlerquadrate

$$f(p) = \frac{1}{2} \sum_{i=1}^n (p(x_i) - y_i)^2$$

minimal wird

- $f(p)$ kann man schreiben als

$$f(p) = \frac{1}{2} F(p)^T F(p), \quad F(p) = Ap - y,$$

mit

$$A = \begin{pmatrix} 1 & x_1 & \dots & x_1^{m-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \dots & x_n^{m-1} \end{pmatrix} \in \mathbb{R}^{n \times m}, \quad p = \begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix} \in \mathbb{R}^m, \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n$$

- F ist also linear in p
- wie wir oben gesehen haben ist das zugehörige Ausgleichsproblem über die Normalgleichungen

$$A^T A p = A^T y$$

einfach lösbar, falls A vollen Rang hat

- hat A keinen vollen Rang, so ist der Minimierer von f nicht eindeutig (siehe **Pseudo-Inverse**)

- sollen die Punkte (x_i, y_i) statt mit einem Polynom p z.B. mit einer Funktion $p(x) = p_1 e^{p_2 x}$ durch Minimieren der Summe der Fehlerquadrate approximiert werden, so erhalten wir analog mit $p = (p_1, p_2)^T$

$$f(p) = \frac{1}{2} \|F(p)\|_2^2, \quad F(p) = \begin{pmatrix} p_1 e^{p_2 x_1} - y_1 \\ \vdots \\ p_1 e^{p_2 x_n} - y_n \end{pmatrix},$$

wobei F jetzt eine nichtlineare Funktion in p ist

Definition 287. Es sei $F : \mathbb{R}^m \rightarrow \mathbb{R}^n$ nichtlinear und hinreichend oft differenzierbar. Das Problem

$$\min_{x \in \mathbb{R}^m} f(x), \quad f(x) = \frac{1}{2} \|F(x)\|_2^2 = \frac{1}{2} F(x)^T F(x) \quad (11.5)$$

heißt **nichtlineares Ausgleichsproblem**

- das nichtlineare Ausgleichsproblem (11.5) ist damit ein nicht restriktives Minimierungsproblem, bei dem die Zielfunktion eine spezielle Struktur hat
- damit können wir prinzipiell die Verfahren aus den vorherigen Kapiteln anwenden
- wegen der speziellen Struktur von f ergeben sich teilweise Vereinfachungen
- die Verfahren aus den letzten Abschnitten greifen auf den Gradienten $f'(x)$ bzw. die Hesse-Matrix $f''(x)$ zu
- aus (11.5) erhalten wir

$$f'(x) = F'(x)^T F(x) \quad (11.6)$$

wobei $F'(x) \in \mathbb{R}^{n \times m}$ die Jacobi-Matrix von F ist, bzw.

$$f''(x) = F'(x)^T F'(x) + \sum_{i=1}^n F_i(x) F_i''(x), \quad (11.7)$$

wobei $F_i''(x) \in \mathbb{R}^{m \times m}$ die Hesse-Matrix von $F_i(x)$ ist

- zur Herleitung eines ersten iterativen Verfahrens nähern wir $F(x)$ lokal durch eine lineare Approximation

$$L_k(x) = F(x_k) + F'(x_k)(x - x_k)$$

an

- statt $f(x) = \frac{1}{2} \|F(x)\|_2^2$ minimieren wir nun

$$\tilde{q}_k(x) = \frac{1}{2} \|L_k(x)\|_2^2$$

und benutzen einen Minimierer als neue Näherung x_{k+1}

- diese Aufgabenstellung können wir auf zwei Arten interpretieren
- einerseits ist

$$L_k(x) = F'(x_k)x - (F'(x_k)x_k - F(x_k)) = A_kx - b_k$$

und wir erhalten

$$\tilde{q}_k(x) = \frac{1}{2}\|A_kx - b_k\|_2^2,$$

d.h. x_{k+1} ist Lösung eines linearen Ausgleichsproblems

- andererseits gilt

$$\begin{aligned} \tilde{q}_k(x) &= \frac{1}{2}L_k^T(x)L_k(x) \\ &= \frac{1}{2}(F(x_k)^TF(x_k) + F(x_k)^TF'(x_k)(x - x_k) \\ &\quad + (x - x_k)^TF'(x_k)^TF(x_k) \\ &\quad + (x - x_k)^TF'(x_k)^TF'(x_k)(x - x_k)) \\ &= \frac{1}{2}(F(x_k)^TF(x_k) + 2(F'(x_k)^TF(x_k))^T(x - x_k) \\ &\quad + (x - x_k)^TF'(x_k)^TF'(x_k)(x - x_k)) \end{aligned}$$

so dass wegen (11.6)

$$\tilde{q}_k(x) = f(x_k) + f'(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^TF'(x_k)^TF'(x_k)(x - x_k)$$

ein lokales quadratisches Modell für f ist mit einer Näherung

$$B_k = F'(x_k)^TF'(x_k) = A_k^TA_k$$

der Hesse-Matrix (11.7)

- hat $A_k = F'(x_k)$ vollen Rang, dann ist B_k spd und x_{k+1} ist eindeutig bestimmt durch

$$\begin{aligned} x_{k+1} &= (A_k^TA_k)^{-1}A_k^Tb_k \\ &= B_k^{-1}A_kb_k \\ &= (F'(x_k)^TF'(x_k))^{-1}F'(x_k)^T(F'(x_k)x_k - F(x_k)) \\ &= x_k - (F'(x_k)^TF'(x_k))^{-1}F'(x_k)^TF(x_k) \\ &= x_k - F'(x_k)^+F(x_k) \end{aligned}$$

Definition 288. Das **Gauß-Newton-Verfahren** für das nichtlineare Ausgleichsproblem (11.5) ist definiert durch

$$x_{k+1} = x_k - F'(x_k)^+F(x_k)$$

Bemerkung 289.

- ◊ pro Schritt muss einmal die Funktion F und die Jacobi-Matrix F' an x_k ausgewertet werden sowie ein lineares Ausgleichsproblem gelöst werden (Pseudo-Inverse)
- ◊ im allgemeinen erhält man lineare Konvergenz
- ◊ würden wir statt

$$B_k = F'(x_k)^T F'(x_k)$$

die exakte Hesse-Matrix

$$f''(x_k) = B_k + \sum_{i=1}^n F_i(x_k) F_i''(x_k)$$

verwenden, so würden wir statt \tilde{q}_k das exakte quadratische Modell q_k benutzen, also das Newton-Verfahren auf $f'(x) = 0$ anwenden und somit ein quadratisch konvergentes Verfahren erhalten

- ◊ ist $\sum_{i=1}^n F_i(x) F_i''(x)$ klein im Vergleich zu B_k , so kann man superlineare Konvergenz beobachten

Beispiel 290.

- wir betrachten das nichtlineare Ausgleichsproblem für $p(x) = p_1 e^{p_2 x}$, d.h.

$$f(p) = \frac{1}{2} \|F(p)\|_2^2, \quad F(p) = \begin{pmatrix} p_1 e^{p_2 x_1} - y_1 \\ \vdots \\ p_1 e^{p_2 x_n} - y_n \end{pmatrix}, \quad p = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

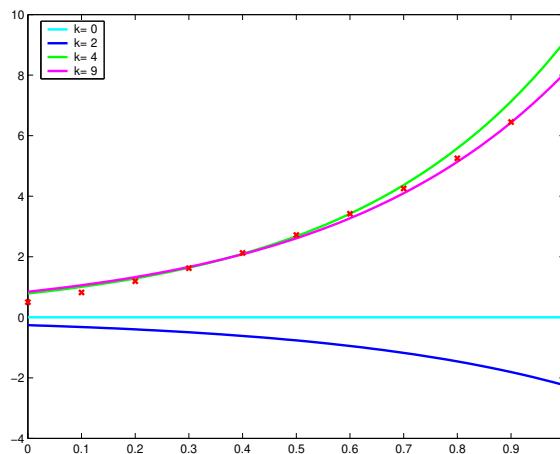
- die gegebenen Daten sind $x_i = \frac{i}{10}$, $i = 0, \dots, 10$ und

i	0	1	2	3	4	5	6	7	8	9	10
y_i	0.500	0.821	1.191	1.622	2.125	2.718	3.420	4.255	5.253	6.449	7.889

- benutzt man als Startwert $p_0 = (0, 0)^T$, so erhält man die Iterierten

k	0	1	2	3	4	5	6	7	8	9
$p_{k,1}$	0	3.2951	-0.2596	0.8398	0.7863	0.8459	0.8454	0.8453	0.8454	0.8454
$p_{k,2}$	0	0	2.1575	1.7993	2.4499	2.2569	2.2552	2.2553	2.2553	2.2553

- für verschiedene Iterationen k ist in der folgenden Abbildung die approximierende Kurve $p(x)$ dargestellt



- wenn $F'(x_k)$ keinen vollen Rang hat, dann ist $B_k = F'(x_k)^T F'(x_k)$ symmetrisch positiv semidefinit, also singulär, d.h. \tilde{q}_k hat keinen eindeutigen Minimierer
 - welchen Minimierer benutzen wir als x_{k+1} ?
 - primitive Lösung:
 - benutze Pseudo-Inverse $F'(x_k)^+$
 - wie wir aus Kapitel 9.6 wissen, kann die Multiplikation mit $F'(x_k)^+$ sehr schlecht konditioniert sein
 - zweite Lösung:
 - x_{k+1} soll \tilde{q}_k minimieren, wird also im Prinzip mit einem (Quasi-)Newtonschritt ermittelt, wobei statt $f''(x_k)$ die Näherung B_k benutzt wird
 - hat $F'(x_k)$ keinen vollen Rang hat, dann ist $B_k = F'(x_k)^T F'(x_k)$ symmetrisch positiv semidefinit, also
- $B_k + \mu I$
- spd für alle $\mu > 0$
- das entspricht der Idee des gedämpften Newton-Verfahrens
 - wir müssen nicht das selbe μ für alle k benutzen, sondern können μ von Schritt zu Schritt anpassen, indem wir z.B. die **Levenberg-Marquardt-Strategie** wie bei den gedämpften Newton-Verfahren benutzen
 - insgesamt erhalten wir das **Levenberg-Marquardt-Verfahren** für nichtlineare Ausgleichsprobleme:

- starte mit $x := x_0, \mu := \mu_0, x_0, \mu_0 > 0$ gegeben
- wiederhole bis Genauigkeit erreicht ist:
 - löse $(F'(x)^T F'(x) + \mu I)p = -f'(x) = -F'(x)^T F(x)$
 - $x_{\text{neu}} := x + p$
 - berechne

$$\rho := \frac{f(x) - f(x_{\text{neu}})}{\tilde{q}(x) - \tilde{q}(x_{\text{neu}})}$$

mit $\tilde{q}(y) = \frac{1}{2} \|F(x) + F'(x)(y - x)\|_2^2$

- ist $\rho > 0$:
 - $x := x_{\text{neu}}$
 - $\mu := \mu \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$
 - $\nu = 2$
- ist $\rho \leq 0$:
 - x_{neu} wegwerfen (aktuellen Schritt wiederholen)
 - $\mu := \nu\mu$
 - $\nu := 2\nu$

Bemerkung 291.

- ◊ der variable Parameter ν dient zum schnelleren Verändern von μ
- ◊ ersetzen wir $B_k = F'(x_k)^T F'(x_k)$ durch $B_k + \mu_k I$, dann bestimmen wir x_{k+1} durch

$$(F'(x_k)^T F'(x_k) + \mu_k I) x_{k+1} = F'(x_k)^T (F'(x_k)x_k - F(x_k))$$

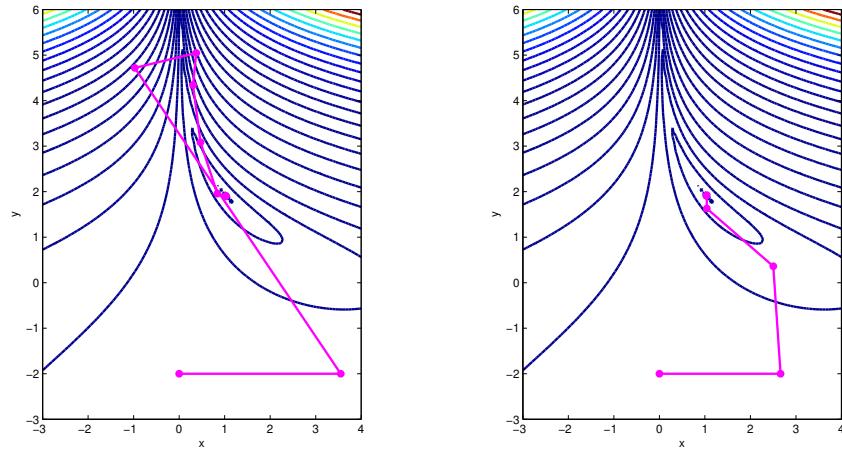
- ◊ das ist die **Tikhonov-Regularisierung** des Originalproblems

$$F'(x_k)^T F'(x_k) x_k = F'(x_k)^T (F'(x_k)x_k - F(x_k))$$

mit Parameter $\alpha^2 = \mu_k$

Beispiel 292.

- wir betrachten noch einmal Beispiel 290 und vergleichen das Gauß-Newton und das Levenberg-Marquardt Verfahren für $p_0 = (0, -2)^T$



Kapitel 12

Diskrete Fourier- und Wavelettransformation

12.1 Motivation

- aus der linearen Algebra ist bekannt, dass es in einem Vektorraum viele unterschiedliche Basen gibt
- besonders einfach zu handhaben sind orthonormale Basen $e_k, \langle e_k, e_l \rangle = \delta_{kl}, k, l = 1, \dots, n$
- für die Koordinaten \hat{u}_k eines Vektors $u \in \mathbb{C}^n$ bezüglich einer beliebigen orthonormalen Basis $e_k, k = 1, \dots, n$, gilt

$$\hat{u}_k = \langle e_k, u \rangle = \bar{e}_k^T u$$

und weiterhin

$$u = \sum_{k=1}^n \hat{u}_k e_k = \sum_{k=1}^n \langle e_k, u \rangle e_k$$

sowie

$$\|u\|^2 = \langle u, u \rangle = \sum_{k=1}^n |\hat{u}_k|^2 = \sum_{k=1}^n \langle e_k, u \rangle^2$$

- die Koordinaten \hat{u}_k geben also die Länge der Projektion von u auf e_k an
- $|\hat{u}_k|$ ist ein Indikator dafür, wie groß der "Anteil" von u in Richtung e_k ist
- im \mathbb{R}^n bzw. \mathbb{C}^n benutzt man als Standardbasis die Basis der orthonormalen Einheitsvektoren $e_k = (0, \dots, 0, 1, 0, \dots, 0)^T, k = 1, \dots, n$
- in Anwendungen ist es oftmals sinnvoll, andere, dem Problem angepasste Orthonormalbasen zu verwenden

Beispiel 293.

- wir wollen im \mathbb{R}^2 Vektoren daraufhin untersuchen, ob sie im Wesentlichen in Richtung einer der Winkelhalbierenden zeigen
- dazu betrachten wir die folgende Orthonormalbasis

$$e_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad e_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

deren Basisvektoren auf den Winkelhalbierenden liegen und berechnen die Koordinaten bezüglich dieser Basis

- betrachten wir $u = (4, 3)^T$, so erhalten wir zunächst $\|u\|^2 = 25$ und

$$\hat{u}_1 = \langle e_1, u \rangle = \frac{7}{\sqrt{2}}$$

bzw.

$$|\hat{u}_1|^2 = \frac{49}{2}$$

- nach oben gilt $\|u\|^2 = |\hat{u}_1|^2 + |\hat{u}_2|^2$ und somit

$$|\hat{u}_2|^2 = 25 - \frac{49}{2} = \frac{1}{2}$$

- da $|\hat{u}_1| \gg |\hat{u}_2|$ sieht man direkt, dass u im Wesentlichen in Richtung e_1 zeigt, also entlang der ersten Winkelhalbierenden

- über Basistransformationen kann man also zusätzliche Einblicke in die Struktur der Vektoren u erhalten
- im nächsten Abschnitt werden wir sehen, dass sich diese Methodik auch auf Funktionen verallgemeinern lässt

12.2 Fourierreihen

- im letzten Abschnitt haben wir gesehen, dass für eine beliebige Orthonormalbasis $e_k, k = 1, \dots, n$, in \mathbb{R}^n bzw. \mathbb{C}^n gilt

$$u = \sum_{k=1}^n \hat{u}_k e_k, \quad \hat{u}_k = \langle e_k, u \rangle, \quad \|u\|^2 = \langle u, u \rangle = \sum_{k=1}^n |\hat{u}_k|^2, \quad \forall u \in \mathbb{R}^n$$

- dieses Eigenschaft kann auch man auf bestimmte Funktionen erweitern

Satz 294 (Komplexe Fourierreihe). Es sei

$$\begin{aligned} L^2(0, 1) &= \left\{ v \mid v : (0, 1) \rightarrow \mathbb{C}, \int_0^1 |v(t)|^2 dt < \infty \right\}, \\ (u, v)_{L^2} &= \int_0^1 \bar{u}(t)v(t) dt, \\ \|u\|_{L^2}^2 &= \sqrt{(u, u)_{L^2}} = \sqrt{\int_0^1 |u(t)|^2 dt}. \end{aligned}$$

$L^2(0, 1)$ ist der Vektorraum der komplexwertigen, quadratintegrierbaren Funktionen auf $(0, 1)$ mit Skalarprodukt $(\cdot, \cdot)_{L^2}$ und dadurch induzierter Norm $\|\cdot\|_{L^2}$.

Die Funktionen

$$e_k(t) = e^{j2\pi kt}, \quad k \in \mathbb{Z}, \quad j = \sqrt{-1},$$

bilden eine Orthonormalbasis von $L^2(0, 1)$, d.h.

$$(e_k, e_l)_{L^2} = \int_0^1 \bar{e}_k(t)e_l(t) dt = \delta_{kl} \quad \forall k, l \in \mathbb{Z}$$

und jede Funktion $u \in L^2(0, 1)$ kann in die komplexe Fourierreihe

$$u(t) = \sum_{k=-\infty}^{\infty} \hat{u}_k e_k(t) \quad (\text{f.i.}),$$

$$\hat{u}_k = (e_k, u)_{L^2} = \int_0^1 e^{-j2\pi kt} u(t) dt,$$

mit

$$\|u\|_{L^2}^2 = (u, u)_{L^2} = \sum_{k=-\infty}^{\infty} |\hat{u}_k|^2$$

entwickelt werden.

- benutzen wir in der komplexen Fourierreihe den Zusammenhang

$$e_k(t) = e^{j2\pi kt} = \cos(2\pi kt) + j\sin(2\pi kt),$$

so erhalten wir mit $c_k(t) = \cos(2\pi kt)$ und $s_k(t) = \sin(2\pi kt)$

$$\begin{aligned} u(t) &= \sum_{k=-\infty}^{\infty} \hat{u}_k (c_k(t) + js_k(t)) \\ \hat{u}_k &= (c_k + js_k, u)_{L^2} = (c_k, u)_{L^2} - j(s_k, u)_{L^2} \end{aligned}$$

bzw. wegen $c_0(t) = 1, s_0(t) = 0, c_{-k}(t) = c_k(-t) = c_k(t), s_{-k}(t) = s_k(-t) = -s_k(t)$

$$\begin{aligned} u(t) &= \sum_{k=-\infty}^{-1} \hat{u}_k (c_k(t) + js_k(t)) + \hat{u}_0 + \sum_{k=1}^{\infty} \hat{u}_k (c_k(t) + js_k(t)) \\ &= \sum_{k=1}^{\infty} \hat{u}_{-k} (c_{-k}(t) + js_{-k}(t)) + \hat{u}_0 + \sum_{k=1}^{\infty} \hat{u}_k (c_k(t) + js_k(t)) \\ &= \underbrace{\hat{u}_0}_{\frac{a_0}{2}} + \sum_{k=1}^{\infty} \underbrace{(\hat{u}_k + \hat{u}_{-k})}_{a_k} c_k(t) + \sum_{k=1}^{\infty} \underbrace{j(\hat{u}_k - \hat{u}_{-k})}_{b_k} s_k(t) \end{aligned}$$

- außerdem ist

$$\begin{aligned} \hat{u}_k &= (e_k, u)_{L^2} = (c_k + js_k, u)_{L^2} = (c_k, u)_{L^2} - j(s_k, u)_{L^2}, \\ \hat{u}_{-k} &= (e_{-k}, u)_{L^2} = (c_{-k} + js_{-k}, u)_{L^2} = (c_k, u)_{L^2} + j(s_k, u)_{L^2} \end{aligned} \quad (12.1)$$

also

$$a_k = 2(c_k, u)_{L^2}, \quad b_k = 2(s_k, u)_{L^2} \quad (12.2)$$

- wegen (12.2) und (12.1) gilt

$$\hat{u}_k = \frac{a_k - jb_k}{2}, \quad \hat{u}_{-k} = \frac{a_k + jb_k}{2}$$

und somit

$$\begin{aligned} \|u\|_{L^2}^2 &= \sum_{k=-\infty}^{\infty} |\hat{u}_k|^2 \\ &= \sum_{k=1}^{\infty} |\hat{u}_{-k}|^2 + |\hat{u}_0|^2 + \sum_{k=1}^{\infty} |\hat{u}_k|^2 \\ &= \left| \frac{a_0}{2} \right|^2 + \sum_{k=1}^{\infty} \left| \frac{a_k + jb_k}{2} \right|^2 + \sum_{k=1}^{\infty} \left| \frac{a_k - jb_k}{2} \right|^2 \\ &= \frac{|a_0|^2}{4} + 2 \sum_{k=1}^{\infty} \frac{|a_k|^2 + |b_k|^2}{4} \end{aligned}$$

womit wir das folgende Ergebnis gezeigt haben

Satz 295. Jede Funktion $u \in L^2(0, 1)$ kann in die Fourierreihe

$$u(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(2\pi k t) + \sum_{k=1}^{\infty} b_k \sin(2\pi k t) \quad (f.ii.),$$

$$a_k = 2(c_k, u)_{L^2} = 2 \int_0^1 \cos(2\pi k t) u(t) dt, \quad k \in \mathbb{N}_0$$

$$b_k = 2(s_k, u)_{L^2} = 2 \int_0^1 \sin(2\pi k t) u(t) dt, \quad k \in \mathbb{N}$$

entwickelt werden und es gilt

$$\|u\|_{L^2}^2 = (u, u)_{L^2} = \frac{|a_0|^2}{4} + \sum_{k=1}^{\infty} \frac{|a_k|^2 + |b_k|^2}{2}$$

Bemerkung 296. Ist u reellwertig, dann sind alle a_k, b_k reell und es gilt

$$\hat{u}_{-k} = \bar{\hat{u}}_k, \quad a_k = 2\Re(\hat{u}_k) = 2\Re(\hat{u}_{-k}), \quad b_k = -2\Im(\hat{u}_k) = 2\Im(\hat{u}_{-k}), \quad k \in \mathbb{N}_0$$

Bemerkung 297.

- ◊ jede Funktion $u \in L^2(0, 1)$ ist also vollständig durch die Folge

$$\hat{u}_k = (e_k, u)_{L^2} = \int_0^1 e^{-j2\pi k t} u(t) dt \in \mathbb{C}, \quad k \in \mathbb{Z}$$

bzw.

$$a_k = 2(c_k, u)_{L^2} = 2 \int_0^1 \cos(2\pi k t) u(t) dt, \quad k \in \mathbb{N}_0$$

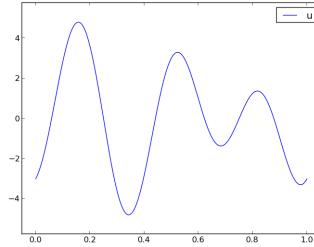
$$b_k = 2(s_k, u)_{L^2} = 2 \int_0^1 \sin(2\pi k t) u(t) dt, \quad k \in \mathbb{N}$$

beschrieben

- ◊ statt $u(t)$ kann man also auch einfach $\hat{u}_k \in \mathbb{C}$, $k \in \mathbb{Z}$, oder $a_k, k \in \mathbb{N}_0$, und $b_k, k \in \mathbb{N}$, betrachten, was wieder als "Basiswechsel" interpretiert werden kann
- ◊ \hat{u}_k bzw. a_k, b_k geben an, mit welchem "Gewicht" die Funktionen $e_k(t) = e^{j2\pi k t}$ bzw. $c_k(t) = \cos(2\pi k t)$, $s_k(t) = \sin(2\pi k t)$ in $u(t)$ vorkommen
- ◊ e_k, c_k und s_k sind periodische Funktionen mit Periodenlänge $1/k$ bzw. Frequenz k
- ◊ sind $|\hat{u}_k|, |\hat{u}_{-k}|$ bzw. $|a_k|, |b_k|$ groß, dann enthält u nennenswerte Anteile mit Frequenz k
- ◊ man kann an \hat{u}_k bzw. a_k, b_k also ganz einfach den "Frequenzinhalt" einer Funktion ablesen, was in der Signal- und Bildverarbeitung äußerst nützlich ist

Beispiel 298.

- wir bestimmen die Fourierreihe für $u(t) = 2 \sin(4\pi t) - 3 \cos(6\pi t)$



- wir berechnen nun nacheinander die komplexen Fourierkoeffizienten

$$\hat{u}_k = (e_k, u)_{L^2} = \int_0^1 e^{-j2\pi kt} u(t) dt$$

und erhalten zunächst

$$\hat{u}_0 = 0, \quad \hat{u}_{-1} = \hat{u}_1 = 0, \quad \hat{u}_{-2} = j, \quad \hat{u}_2 = -j, \quad \hat{u}_{-3} = \hat{u}_3 = -\frac{3}{2}$$

- nach oben muss

$$\|u\|_{L^2}^2 = (u, u)_{L^2} = \int_0^1 |u(t)|^2 dt = \sum_{k=-\infty}^{\infty} |\hat{u}_k|^2$$

gelten

- nun ist aber

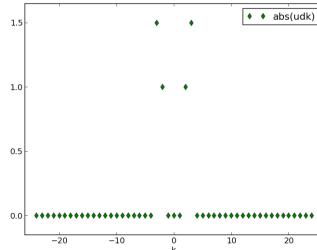
$$\int_0^1 |u(t)|^2 dt = \frac{13}{2}$$

und

$$\sum_{k=-3}^3 |\hat{u}_k|^2 = \left| -\frac{3}{2} \right|^2 + |j|^2 + |0|^2 + |0|^2 + |0|^2 + |-j|^2 + \left| -\frac{3}{2} \right|^2 = \frac{13}{2},$$

weshalb alle weiteren \hat{u}_k verschwinden müssen

- die folgende Abbildung zeigt $|\hat{u}_k|$ für $k = -25, \dots, 25$ (Amplitudenspektrum)



- man erkennt, dass nur $k = \pm 2$ und $k = \pm 3$ einen Beitrag liefern
- wir erhalten also die (endliche) komplexe Fourierreihe

$$u(t) = -\frac{3}{2}e_{-3}(t) + je_{-2}(t) - je_2(t) - \frac{3}{2}e_3(t) = -\frac{3}{2}(e^{-j6\pi t} + e^{j6\pi t}) + j(e^{-j4\pi t} - e^{j4\pi t})$$

- da u reellwertig ist gilt $\hat{u}_{-k} = \bar{\hat{u}}_k$, $a_k = 2\Re(\hat{u}_k)$, $b_k = -2\Im(\hat{u}_k)$, so dass wegen

$$\hat{u}_0 = 0, \quad \hat{u}_{-1} = \hat{u}_1 = 0, \quad \hat{u}_{-2} = j, \quad \hat{u}_2 = -j, \quad \hat{u}_{-3} = \hat{u}_3 = -\frac{3}{2}$$

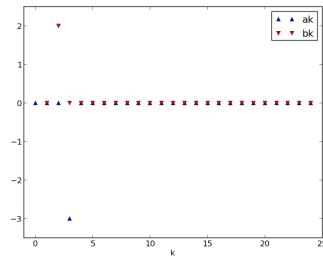
alle Koeffizienten außer

$$a_2 = 2\Re(-j) = 0, \quad a_3 = 2\Re\left(-\frac{3}{2}\right) = -3$$

bzw.

$$b_2 = -2\Im(-j) = 2, \quad b_3 = -2\Im\left(-\frac{3}{2}\right) = 0$$

verschwinden



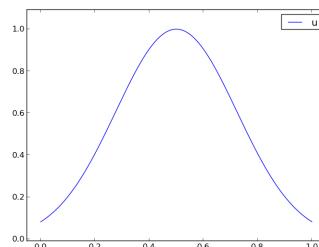
- in u tauchen also nur die Frequenzen $k = 2$ und $k = 3$ auf

- somit erhalten wir die (offensichtliche) Reihenentwicklung

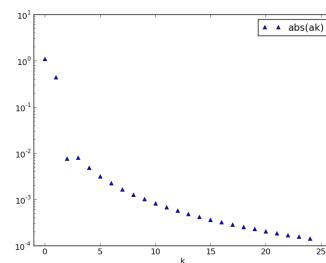
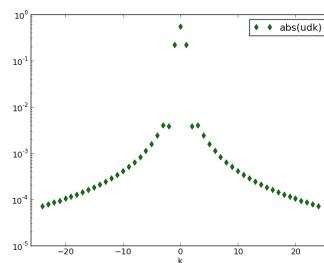
$$u(t) = 2s_2(t) - 3c_3(t) = 2\sin(2 \cdot 2\pi t) - 3\cos(3 \cdot 2\pi t)$$

Beispiel 299.

- wir betrachten die Fourierreihe von $u(t) = e^{-10(t-1/2)^2}$



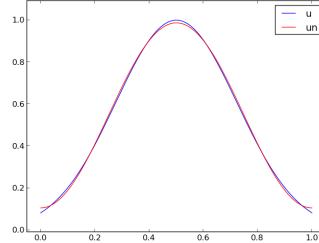
- die Reihe ist nicht endlich, für $|\hat{u}_k|$ bzw. $|\hat{a}_k|$ ($b_k = 0 \forall k$) erhalten wir in logarithmischem Maßstab



- man sieht, dass $k = 0$ und $k = \pm 1$ dominieren
- vernachlässigt man alle anderen Anteile, d.h. approximiert man u durch

$$\tilde{u}(t) = c_{-1}e_{-1}(t) + c_0e_0(t) + c_1e_1(t)$$

so ergibt sich folgendes Bild

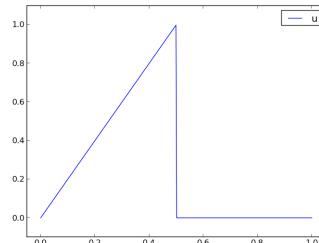


- durch Transformation und Approximation könnte man also eine (verlustbehaftete) Datenkompression erreichen

Beispiel 300.

- wir betrachten

$$u(t) = \begin{cases} 2t & t \leq \frac{1}{2} \\ 0 & \frac{1}{2} < t \end{cases}$$



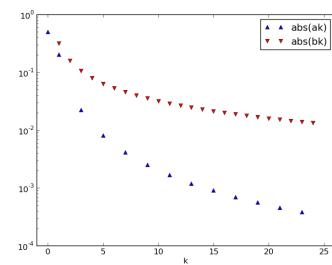
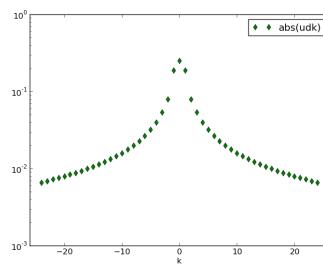
- die Fourierkoeffizienten sind

$$\hat{u}_0 = \frac{1}{4}, \quad \hat{u}_k = \frac{(-1)^k - 1}{2(k\pi)^2} + j \frac{(-1)^k}{2k\pi} \quad k \neq 0$$

bzw.

$$a_0 = \frac{1}{2}, \quad a_k = \frac{(-1)^k - 1}{(k\pi)^2}, \quad b_k = -\frac{(-1)^k}{k\pi}, \quad k \neq 0$$

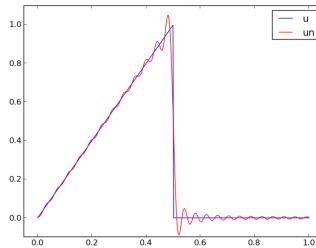
- die Reihe ist nicht endlich, die Koeffizienten fallen relativ langsam ab (logarithmischer Maßstab)



- approximiert man u durch

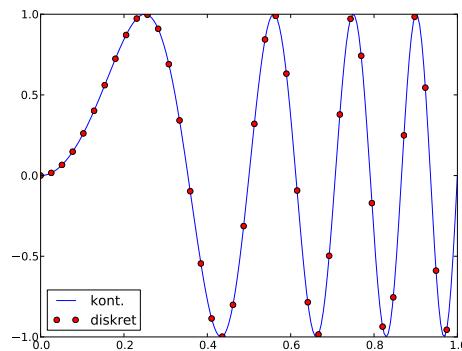
$$\tilde{u}(t) = \sum_{k=-25}^{25} c_k e_k(t)$$

so erhalten wir nur eine mäßige Approximationsqualität



12.3 Diskrete Fouriertransformation

- in der digitalen Signalverarbeitung liegen die Signale nicht als kontinuierliche Funktionen $u(t)$ sondern diskret in Form von äquidistant im Abstand $\Delta t = \frac{1}{N}$ abgetasteten Werten vor, wobei N die Anzahl der Abtastpunkte ist



- an $t_l = \frac{l}{N}$ liegt also ein Wert $u_l \in \mathbb{C}$ vor, $l = 0, \dots, N-1$
- statt $t \in (0, 1)$ haben wir also diskrete Zeitpunkte $\{t_0, \dots, t_{N-1}\}$, statt Funktionswerte $u(t) \in \mathbb{C}$ haben wir diskrete Werte $\{u_0, \dots, u_{N-1}\} \in \mathbb{C}^N$
- diese Analogien kann man auf alle Betrachtungen des letzten Kapitels ausweiten

kontinuierlich	diskret
$t \in (0, 1)$	$t_l \in \{t_0, \dots, t_{N-1}\}$
$u \in L^2(0, 1)$	$u = \{u_0, \dots, u_{N-1}\} \in \mathbb{C}^N$

kontinuierlich	diskret
$(u, v)_{L^2} = \int_0^1 \bar{u}(t)v(t) dt$	$\langle u, v \rangle_N = \sum_{l=0}^{N-1} \Delta t \bar{u}_l v_l = \frac{1}{N} \langle u, v \rangle$ (entspricht Integration mit der linksseitigen summierten Rechteckregel)
$e_k(t) = e^{j2\pi k t}$	$e_k = (e^{j2\pi k t_l})_{l=0,\dots,N-1}$
die Funktionen $e_k(t) = e^{j2\pi k t}$ sind orthonormal in $L^2(0,1)$ bzgl. $(\cdot, \cdot)_{L^2}$	die Vektoren $e_k = (e^{j2\pi k t_l})_{l=0,\dots,N-1}$ sind orthonormal in \mathbb{C}^N bzgl. $\langle \cdot, \cdot \rangle_N$
$\forall u \in L^2(0,1)$ gilt	$\forall u \in \mathbb{C}^N, N = 2n$, gilt
$u(t) = \sum_{k=-\infty}^{\infty} \hat{u}_k e_k(t) \quad (\text{f.ü.})$ $\hat{u}_k = (e_k, u)_{L^2}$ $\ u\ _{L^2}^2 = \sum_{k=-\infty}^{\infty} \hat{u}_k ^2$	$u = \sum_{k=-n}^{n-1} \hat{u}_k e_k$ $\hat{u}_k = \langle e_k, u \rangle_N$ $\ u\ _N^2 = \sum_{k=-n}^{n-1} \hat{u}_k ^2$
$c_k(t) = \cos(2\pi k t)$ $s_k(t) = \sin(2\pi k t)$	$c_k = (\cos(2\pi k t_l))_{l=0,\dots,N-1}$ $s_k = (\sin(2\pi k t_l))_{l=0,\dots,N-1}$
$\forall u \in L^2(0,1)$ gilt	$\forall u \in \mathbb{C}^N, N = 2n$, gilt
$u(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k c_k(t)$ $+ \sum_{k=1}^{\infty} b_k s_k(t) \quad (\text{f.ü.})$ $a_k = 2(c_k, u)_{L^2}$ $b_k = 2(s_k, u)_{L^2}$ $\ u\ _{L^2}^2 = \frac{ a_0 ^2}{4} + \sum_{k=1}^{\infty} \frac{ a_k ^2}{2} + \sum_{k=1}^{\infty} \frac{ b_k ^2}{2}$	$u = \frac{a_0}{2} + \sum_{k=1}^n a_k c_k$ $+ \sum_{k=1}^{n-1} b_k s_k$ $a_k = 2\langle c_k, u \rangle_N$ $b_k = 2\langle s_k, u \rangle_N$ $\ u\ _N^2 = \frac{ a_0 ^2}{4} + \sum_{k=1}^n \frac{ a_k ^2}{2} + \sum_{k=1}^{n-1} \frac{ b_k ^2}{2}$
ist u reell, dann gilt $a_k = 2\Re(\hat{u}_k) = 2\Re(\hat{u}_{-k}),$ $b_k = -2\Im(\hat{u}_k) = 2\Im(\hat{u}_{-k})$	ist u reell, dann gilt $a_k = 2\Re(\hat{u}_k) = 2\Re(\hat{u}_{-k}),$ $b_k = -2\Im(\hat{u}_k) = 2\Im(\hat{u}_{-k})$

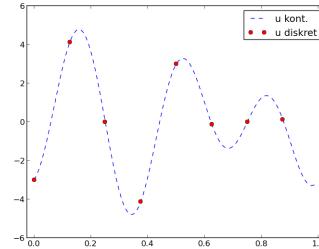
- diskret wird also ein Vektor $u \in \mathbb{C}^N$ in einen neuen Vektor $\hat{u} = (\hat{u}_{-n}, \dots, \hat{u}_{n-1})^T$ transformiert, dessen Komponenten die Koordinaten von u bezüglich der Orthonormalbasis $e_k = (e^{j2\pi k t_l})_{l=0,\dots,N-1}, k = 0, \dots, N-1$, sind

$$u = \sum_{k=-n}^{n-1} \hat{u}_k e_k, \quad \hat{u}_k = \langle e_k, u \rangle_N$$

Beispiel 301.

- wir betrachten $u(t) = 2 \sin(4\pi t) - 3 \cos(6\pi t)$ aus Beispiel 298, tasten diskret an $N = 8$ Punkten ab und erhalten somit den Vektor von Funktionswerten

$$u \approx (-3, 4.12, 0, -4.12, 3, -0.12, 0, 0.12)^T$$

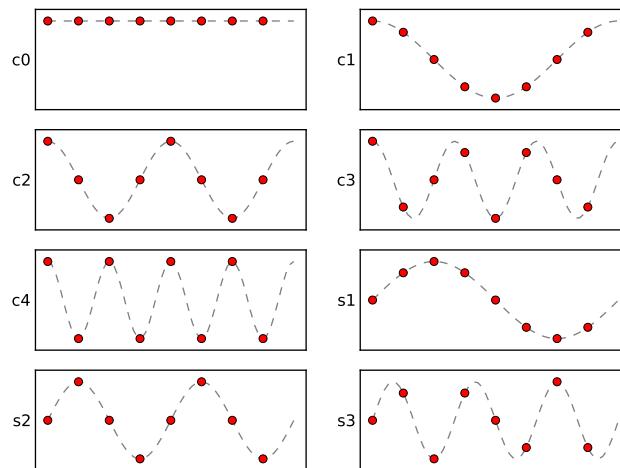


- wir transformieren u in die Form

$$u = \frac{a_0}{2} + \sum_{k=1}^n a_k c_k + \sum_{k=1}^{n-1} b_k s_k \quad a_k = 2 \langle c_k, u \rangle_N, \quad b_k = 2 \langle s_k, u \rangle_N$$

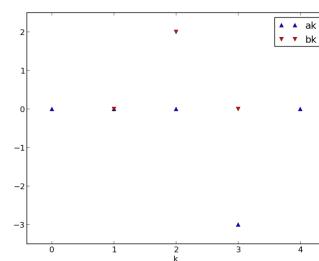
d.h. wir benutzen als neue Basen die c_k und s_k

- die c_k und s_k sind Vektoren aus \mathbb{R}^8 und haben folgende Form



- als Ergebnis erhalten wir

$$a = (0, 0, 0, -3, 0)^T, \quad b = (0, 2, 0)^T$$



- wir haben also den Vektor

$$u \approx (-3, 4.12, 0, -4.12, 3, -0.12, 0, 0.12)^T \in \mathbb{R}^8$$

in den Vektor

$$(0, 0, 0, -3, 0, 0, 2, 0)^T \in \mathbb{R}^8$$

transformiert

- benutzen wir die komplexe Variante

$$u = \sum_{k=-n}^{n-1} \hat{u}_k e_k$$

so transformieren wir analog

$$u \approx (-3, 4.12, 0, -4.12, 3, -0.12, 0, 0.12)^T \in \mathbb{C}^8$$

in

$$\hat{u} = (0, -3/2, j, 0, 0, 0, -j, -3/2)^T \in \mathbb{C}^8$$

- diese Darstellung wird in der Regel noch etwas umgeformt
- mit $t_l = l/N$ erhalten wir

$$e_{N-k} = (e^{j\frac{2\pi}{N}(N-k)l})_{l=0,\dots,N-1} = (\underbrace{e^{j2\pi l}}_1 e^{j\frac{2\pi}{N}(-k)l})_{l=0,\dots,N-1} = e_{-k}$$

also auch

$$\hat{u}_{N-k} = \langle e_{N-k}, u \rangle_N = \langle e_{-k}, u \rangle_N = \hat{u}_{-k}$$

- somit ergibt sich für $N = 2n$

$$\begin{aligned} u &= \sum_{k=-n}^{n-1} \hat{u}_k e_k \\ &= \sum_{k=-n}^{-1} \hat{u}_k e_k + \sum_{k=0}^{n-1} \hat{u}_k e_k \\ &= \sum_{k=1}^n \hat{u}_{-k} e_{-k} + \sum_{k=0}^{n-1} \hat{u}_k e_k \\ &= \sum_{k=0}^{n-1} \hat{u}_k e_k + \sum_{k=1}^n \hat{u}_{-k} e_{N-k} \\ &= \hat{u}_0 e_0 + \dots + \hat{u}_{n-1} e_{n-1} + \hat{u}_{-n} e_n + \hat{u}_{-n+1} e_{n+1} + \dots + \hat{u}_{-1} e_{N-1} \\ &= \hat{u}_0 e_0 + \dots + \hat{u}_{n-1} e_{n-1} + \hat{u}_n e_n + \hat{u}_{n+1} e_{n+1} + \dots + \hat{u}_{N-1} e_{N-1} \\ &= \sum_{k=0}^{N-1} \hat{u}_k e_k \end{aligned}$$

Definition 302. Es sei $u = (u_0, \dots, u_{N-1})^T \in \mathbb{C}^N$, $\hat{u} = (\hat{u}_0, \dots, \hat{u}_{N-1})^T \in \mathbb{C}^N$ mit

$$u = \sum_{k=0}^{N-1} \hat{u}_k e_k, \quad e_k = (e^{j\frac{2\pi}{N}kl})_{l=0,\dots,N-1}$$

\hat{u} ist die **Diskrete Fouriertransformation (DFT)** von u bzw. u ist die **Inverse Diskrete Fouriertransformation (IDFT)** von \hat{u} .

Bemerkung 303. Die DFT wird oft als Analyse, die IDFT als Synthese bezeichnet.

- betrachten wir die DFT/IDFT genauer, so können wir noch einige Vereinfachungen vornehmen
- mit der N -ten komplexen Einheitswurzel

$$\omega = e^{j \frac{2\pi}{N}}$$

erhalten wir

$$\hat{u}_k = \langle e_k, u \rangle_N, \quad e_k = (e^{j \frac{2\pi}{N} kl})_{l=0, \dots, N-1} = (\omega^{kl})_{l=0, \dots, N-1}, \quad k = 0, \dots, N-1,$$

also

$$\hat{u}_k = \frac{1}{N} \sum_{l=0}^{N-1} \bar{\omega}^{kl} u_l = \frac{1}{N} (\bar{\Omega} u)_k$$

mit der Matrix

$$\Omega = (\omega^{kl})_{k,l=0, \dots, N-1}$$

deren Spalten (und Zeilen) aus den Basisvektoren e_k bestehen

- die DFT ist also nichts weiter als ein Matrix-Vektor-Produkt

$$\hat{u} = \frac{1}{N} \bar{\Omega} u$$

- für die IDFT benutzen wir, dass die e_k eine Orthonormalbasis von \mathbb{C}^N bezüglich $\langle \cdot, \cdot \rangle_N$ bilden
- wegen

$$\delta_{km} = \langle e_k, e_m \rangle_N = \frac{1}{N} \sum_{l=0}^{N-1} \bar{e}_{k,l} e_{l,m} = \frac{1}{N} (\bar{\Omega} \Omega^T)_{km}$$

gilt

$$\frac{1}{N} \bar{\Omega} \Omega^T = I$$

also

$$\bar{\Omega}^{-1} = \frac{1}{N} \Omega^T = \frac{1}{N} \Omega$$

und somit

$$u = \left(\frac{1}{N} \bar{\Omega} \right)^{-1} \hat{u} = N \bar{\Omega}^{-1} \hat{u} = \Omega \hat{u}$$

Satz 304. Für die DFT bzw. IDFT in \mathbb{C}^N gilt

$$\hat{u} = \frac{1}{N} \bar{\Omega} u, \quad u = \Omega \hat{u}$$

mit der Matrix

$$\Omega = (\omega^{kl})_{k,l=0, \dots, N-1}, \quad \omega = e^{j \frac{2\pi}{N}}$$

Bemerkung 305.

- ◊ DFT bzw. IDFT sind also Matrix-Vektor-Produkte mit Matrix $\frac{1}{N}\bar{\Omega}$ bzw. Ω
- ◊ in den Standardimplementierungen wird in der Regel der Vorfaktor $1/N$ anders berücksichtigt
- ◊ dort werden Matrix-Vektor-Produkte mit Matrix $\bar{\Omega}$ bzw. $\frac{1}{N}\Omega$ durchgeführt, d.h. man berechnet statt \hat{u} den Vektor

$$\tilde{u} = \bar{\Omega}u = N\hat{u}$$

bzw. bei der IDFT

$$u = \frac{1}{N}\Omega\tilde{u} = \Omega\hat{u}$$

12.4 Schnelle Fouriertransformation (FFT)

- bei der DFT berechnet man (im wesentlichen) $\bar{\Omega}u$
- der Aufwand dafür liegt für $u \in \mathbb{C}^N$ bei $\mathcal{O}(N^2)$
- in vielen Anwendungen ist N sehr groß, so dass der Aufwand zu hoch ist
- den Aufwand kann man reduzieren, wenn man die spezielle Struktur der Matrix $\bar{\Omega}$ ausnutzt
- dazu betrachten wir die DFT noch einmal genauer
- setzen wir

$$\omega_m = e^{j\frac{2\pi}{m}}, \quad \Omega_m = (\omega_m^{kl})_{k,l=0,\dots,m-1},$$

so erhalten wir für die DFT von $u \in \mathbb{C}^N$

$$\tilde{u} = \bar{\Omega}_N u$$

- es gilt $\omega_m^m = 1$
- für $N = 2n$ erhalten wir

$$\begin{aligned} \omega_N^2 &= \left(e^{j\frac{2\pi}{N}}\right)^2 = e^{j\frac{2\pi}{2n}2} = e^{j\frac{2\pi}{n}} = \omega_n \\ \omega_N^n &= \left(e^{j\frac{2\pi}{N}}\right)^n = e^{j\frac{2\pi}{2n}n} = e^{j\pi} = -1 \end{aligned}$$

und somit auch

$$\bar{\omega}_N^2 = \bar{\omega}_n, \quad \bar{\omega}_N^n = -1$$

- wir betrachten nun $\tilde{u} = \bar{\Omega}_N u$, $N = 2n$, und sortieren die Spalten in $\bar{\Omega}$ und die Einträge in u so um, dass zunächst alle geraden, dann alle ungeraden Indizes auftreten

$$\tilde{u} = \bar{\Omega}_N u = (\bar{\omega}_N^{kl})_{k,l=0,\dots,N-1} \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix}$$

$$= \left(\bar{\omega}_N^{k2l} \mid \bar{\omega}_N^{k(2l+1)} \right)_{\substack{k=0, \dots, N-1 \\ l=0, \dots, n-1}} \begin{pmatrix} u_0 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_1 \\ u_3 \\ \vdots \\ u_{N-1} \end{pmatrix}$$

- mit $u_g = (u_0, u_2, \dots, u_{N-2})^T$, $u_u = (u_1, u_2, \dots, u_{N-1})^T$ und $\bar{\omega}_N^2 = \bar{\omega}_n$ erhalten wir

$$\begin{aligned} \tilde{u} &= \left(\bar{\omega}_N^{k2l} \mid \bar{\omega}_N^{k(2l+1)} \right)_{\substack{k=0, \dots, N-1 \\ l=0, \dots, n-1}} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \\ &= \begin{pmatrix} \left((\bar{\omega}_N^2)^{kl} \right)_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} & \left(\bar{\omega}_N^k (\bar{\omega}_N^2)^{kl} \right)_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} \\ \left((\bar{\omega}_N^2)^{kl} \right)_{\substack{k=n, \dots, 2n-1 \\ l=0, \dots, n-1}} & \left(\bar{\omega}_N^k (\bar{\omega}_N^2)^{kl} \right)_{\substack{k=n, \dots, 2n-1 \\ l=0, \dots, n-1}} \end{pmatrix} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \\ &= \begin{pmatrix} (\bar{\omega}_n^{kl})_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} & (\bar{\omega}_N^k \bar{\omega}_n^{kl})_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} \\ (\bar{\omega}_n^{kl})_{\substack{k=n, \dots, 2n-1 \\ l=0, \dots, n-1}} & (\bar{\omega}_N^k \bar{\omega}_n^{kl})_{\substack{k=n, \dots, 2n-1 \\ l=0, \dots, n-1}} \end{pmatrix} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \\ &= \begin{pmatrix} (\bar{\omega}_n^{kl})_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} & (\bar{\omega}_N^k \bar{\omega}_n^{kl})_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} \\ (\bar{\omega}_n^{(k+n)l})_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} & (\bar{\omega}_N^{k+n} \bar{\omega}_n^{(k+n)l})_{\substack{k=0, \dots, n-1 \\ l=0, \dots, n-1}} \end{pmatrix} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \\ &= \begin{pmatrix} (\bar{\omega}_n^{kl})_{\substack{k,l=0, \dots, n-1}} & (\bar{\omega}_N^k \bar{\omega}_n^{kl})_{\substack{k,l=0, \dots, n-1}} \\ (\underbrace{\bar{\omega}_n^{nl} \bar{\omega}_n^{kl}}_1)_{\substack{k,l=0, \dots, n-1}} & (\underbrace{\bar{\omega}_N^n \bar{\omega}_N^k \bar{\omega}_n^{nl} \bar{\omega}_n^{kl}}_1)_{\substack{k,l=0, \dots, n-1}} \end{pmatrix} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \\ &= \begin{pmatrix} (\bar{\omega}_n^{kl})_{\substack{k,l=0, \dots, n-1}} & (\bar{\omega}_N^k \bar{\omega}_n^{kl})_{\substack{k,l=0, \dots, n-1}} \\ (\bar{\omega}_n^{kl})_{\substack{k,l=0, \dots, n-1}} & (-\bar{\omega}_N^k \bar{\omega}_n^{kl})_{\substack{k,l=0, \dots, n-1}} \end{pmatrix} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \end{aligned}$$

- definieren wir nun die Diagonalmatrix

$$\bar{D}_n = \begin{pmatrix} 1 & & & \\ & \bar{\omega}_N & & \\ & & \ddots & \\ & & & \bar{\omega}_N^{n-1} \end{pmatrix}$$

und benutzen wir $\bar{\Omega}_n = (\bar{\omega}_n^{kl})_{k,l=0, \dots, n-1}$, so erhalten wir

$$\tilde{u} = \bar{\Omega}_n u = \begin{pmatrix} \bar{\Omega}_n & \bar{D}_n \bar{\Omega}_n \\ \bar{\Omega}_n & -\bar{D}_n \bar{\Omega}_n \end{pmatrix} \begin{pmatrix} u_g \\ u_u \end{pmatrix} \quad (12.3)$$

- um \tilde{u} zu bestimmen, muss also $\bar{\Omega}_n u_g$, $\bar{\Omega}_n u_u$ berechnet (und anschließend skaliert und addiert) werden

- der Aufwand reduziert sich von $\sim N^2$ auf $\sim 2n^2 = N^2/2$ (zuzüglich $\mathcal{O}(N)$ für das Skalieren und Addieren)
- ist n wieder gerade, so kann die Multiplikation mit $\bar{\Omega}_n$ analog reduziert werden (Rekursion)
- für $N = 2^s$ verringert sich der Gesamtaufwand von $\mathcal{O}(N^2)$ auf $\mathcal{O}(N \log(N))$
- aus (12.3) erhalten wir schließlich

Satz 306 (FFT-Algorithmus von Cooley und Tukey). Der bei der DFT auftretende Vektor \tilde{u} kann für $N = 2n$ mit der Rekursionsvorschrift

$$\tilde{u} = \bar{\Omega}_N u = \bar{\Omega}_{2n} u = \begin{pmatrix} v + w \\ v - w \end{pmatrix}, \quad v = \bar{\Omega}_n u_g, \quad w = \bar{D}_n \bar{\Omega}_n u_u$$

berechnet werden. Dabei ist

$$\begin{aligned} u_g &= (u_0, u_2, \dots, u_{N-2})^T \\ u_u &= (u_1, u_3, \dots, u_{N-1})^T \\ \bar{D}_n &= \text{diag}(1, \bar{\omega}_{2n}, \bar{\omega}_{2n}^2, \dots, \bar{\omega}_{2n}^{n-1}). \end{aligned}$$

Für $N = 2^s$ muss die Rekursion erst bei $n = 1$ ($\bar{\Omega}_1 = \bar{D}_1 = 1$) gestoppt werden. Der Aufwand beträgt dann $\mathcal{O}(N \log(N))$.

Bemerkung 307.

- ◊ Cooley und Tukey haben diesen Algorithmus 1965 veröffentlicht, die Grundidee findet sich aber auch schon in Arbeiten von Gauß (ca. 1805) wieder
- ◊ die Reduktion des Aufwands hat viele Anwendungen im Bereich der Signalverarbeitung möglich gemacht, die heute aus dem alltäglichen Leben nicht mehr wegzudenken sind
- ◊ der Cooley-Tukey-FFT Algorithmus wurde in die Top-10-Liste der wichtigsten Algorithmen des 20. Jahrhunderts gewählt

Lemma 308. Für die IDFT erhält man für $N = 2n$ analog

$$u = \frac{1}{N} \Omega_N \tilde{u} = \frac{1}{2n} \Omega_{2n} \tilde{u} = \frac{1}{2n} \begin{pmatrix} v + w \\ v - w \end{pmatrix}, \quad v = \Omega_n \tilde{u}_g, \quad w = D_n \Omega_n \tilde{u}_u,$$

$$\begin{aligned} \tilde{u}_g &= (\tilde{u}_0, \tilde{u}_2, \dots, \tilde{u}_{N-2})^T \\ \tilde{u}_u &= (\tilde{u}_1, \tilde{u}_3, \dots, \tilde{u}_{N-1})^T \\ D_n &= \text{diag}(1, \omega_{2n}, \omega_{2n}^2, \dots, \omega_{2n}^{n-1}). \end{aligned}$$

Für $N = 2^s$ muss die Rekursion erst bei $n = 1$ ($\Omega_1 = D_1 = 1$) gestoppt werden. Der Aufwand beträgt dann $\mathcal{O}(N \log(N))$.

Beispiel 309.

- wir wenden FFT zur verlustbehafteten Kompression eines Audiosignals an
- das Originalsignal u besteht aus $N = 2^{20} \approx 10^6$ Abtastwerten (entspricht ca. 24 Sekunden in Standard-CD-Qualität)



- zunächst wird u per FFT in \tilde{u} transformiert
 - in \tilde{u} werden die betragsmäßig kleinen Komponenten (aufsteigend nach Größe, angefangen bei den kleinsten) auf 0 gesetzt, wodurch ein komprimierter Vektor \tilde{u}_c entsteht (es müssen nur die Nicht-Null-Elemente in einem geeigneten Sparse-Format abgespeichert werden)
 - in diesem Beispiel wurde die Anzahl der eliminierten Komponenten so groß wie möglich gewählt, so dass
- $$\|\tilde{u}_c\|_2 \geq 0.98 \|\tilde{u}\|_2$$
- gilt, d.h. 98% der "Signalenergie" soll nach der Kompression erhalten bleiben
- die Anzahl der Nicht-Null-Elemente von \tilde{u}_c beträgt dann $N_c = 135\,292$ statt $N = 1\,048\,576$ (Faktor ≈ 7.75)
 - aus \tilde{u}_c kann durch IFFT ein Audiosignal u_c rekonstruiert werden
 - im Hörvergleich zum Original u ist u_c durchaus brauchbar

Zusammenfassung.

- DFT/FFT transformieren Vektoren (Signale) in eine neue Basis (e_k bzw. c_k, s_k)
- die neuen Koeffizienten haben eine anschauliche Bedeutung (Anteile einzelner Frequenzen im Signal)
- damit wird die Signalverarbeitung (Kompression, Filterung etc.) besonders einfach
- Grundlage für viele Technologien, wie z.B. MP3 und JPEG

12.5 Diskrete Wavelettransformation

- bei der DFT/FFT haben wir die Vektoren

$$e_k = (e^{j\frac{2\pi k l}{N}})_{l=0, \dots, N-1}, \quad k = 0, \dots, N-1$$

als neue Basis im \mathbb{C}^N benutzt

- seit einiger Zeit werden auch andere Basen in der Signalverarbeitung eingesetzt
- die Grundidee erläutern wir am Beispiel von $u \in \mathbb{C}^8$

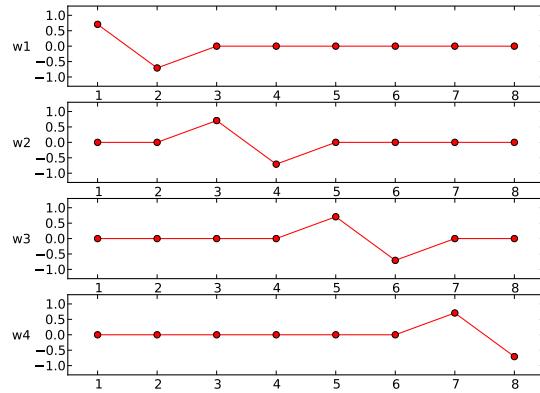
- wir betrachten zunächst die folgenden 4 Vektoren

$$w_1 = \frac{1}{\sqrt{2}} (1, -1, 0, 0, 0, 0, 0, 0)^T$$

$$w_2 = \frac{1}{\sqrt{2}} (0, 0, 1, -1, 0, 0, 0, 0)^T$$

$$w_3 = \frac{1}{\sqrt{2}} (0, 0, 0, 0, 1, -1, 0, 0)^T$$

$$w_4 = \frac{1}{\sqrt{2}} (0, 0, 0, 0, 0, 0, 1, -1)^T$$



- es gilt $\|w_i\| = 1$, $\langle w_i, w_j \rangle = \delta_{ij}$ (euklidische Norm und Skalarprodukt), d.h. die w_i sind orthonormal
- die w_i sehen aus wie kleine "Wellen" (daher der Name **Wavelets**) und sind sehr stark lokalisiert (sie variieren nur an zwei direkt benachbarten Einträgen, ansonsten sind sie identisch 0)
- sie zeigen also "hochfrequentes" Verhalten
- berechnen wir die Längen h_i der Projektionen von u auf die Vektoren w_i , so liefern diese uns Information über die hochfrequenten Anteile in u
- da die w_i orthonormal sind, berechnen sich die h_i wieder durch einfache Skalarprodukte, d.h.

$$h_i = \langle w_i, u \rangle = \frac{u_{2i-1} - u_{2i}}{\sqrt{2}}, \quad i = 1, \dots, 4$$

bzw.

$$h = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} u_1 - u_2 \\ u_3 - u_4 \\ u_5 - u_6 \\ u_7 - u_8 \end{pmatrix} \in \mathbb{C}^4$$

- um den niedrfrequenten Rest \tilde{u} von u zu ermitteln, subtrahieren wir von u die Projektionen

auf die w_i , d.h.

$$\tilde{l} = u - \sum_{i=1}^4 \langle w_i, u \rangle w_i = u - \sum_{i=1}^4 h_i w_i = \frac{1}{2} \begin{pmatrix} u_1 + u_2 \\ u_1 + u_2 \\ u_3 + u_4 \\ u_3 + u_4 \\ u_5 + u_6 \\ u_5 + u_6 \\ u_7 + u_8 \\ u_7 + u_8 \end{pmatrix} \in \mathbb{C}^8$$

- in \tilde{l} sind je zwei aufeinander folgende Einträge identisch, da die komponentenweisen Änderungen in den Projektionen auf die w_i stecken
- man kann damit ohne Informationsverlust jede zweite Komponente weglassen (**Downsampling**)
- damit der so erhaltene Vektor $l \in \mathbb{C}^4$ die selbe euklidische Norm wie \tilde{l} hat, wird noch mit einem Faktor $\sqrt{2}$ skaliert

$$l = \sqrt{2} \frac{1}{2} \begin{pmatrix} u_1 + u_2 \\ u_3 + u_4 \\ u_5 + u_6 \\ u_7 + u_8 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} u_1 + u_2 \\ u_3 + u_4 \\ u_5 + u_6 \\ u_7 + u_8 \end{pmatrix}$$

- zusammengefasst haben wir also folgendes Ergebnis:

- wir haben $u \in \mathbb{C}^8$ in $(l, h) \in \mathbb{C}^8$ transformiert durch

$$l_i = \frac{u_{2i-1} + u_{2i}}{\sqrt{2}}, \quad h_i = \frac{u_{2i-1} - u_{2i}}{\sqrt{2}}$$

- die h_i sind die Längen der Projektionen von u auf die hochfrequenten Orthonormalvektoren w_i
- es gilt $\|u\| = \|(l, h)\|$
- aus (l, h) kann u durch

$$u_{2i-1} = \frac{l_i + h_i}{\sqrt{2}}, \quad u_{2i} = \frac{l_i - h_i}{\sqrt{2}}$$

rekonstruiert werden

- analog kann nun l wieder in einen hoch- bzw. niederfrequenten Anteil zerlegt werden
- setzen wir $l^{(1)} = l, h^{(1)} = h, w_i^{(1)} = w_i$, so folgt durch rekursive Anwendung

$$l^{(1)} \longrightarrow (l^{(2)}, h^{(2)}), \quad l^{(2)}, h^{(2)} \in \mathbb{C}^2, \quad \|l^{(1)}\| = \|(l^{(2)}, h^{(2)})\|$$

bzw. im letzten Schritt

$$l^{(2)} \longrightarrow (l^{(3)}, h^{(3)}), \quad l^{(3)}, h^{(3)} \in \mathbb{C}, \quad \|l^{(2)}\| = \|(l^{(3)}, h^{(3)})\|$$

- mit $h_1^{(4)} = l_1^{(3)}$ erhalten wir insgesamt die Transformation

$$\mathbb{C}^8 \ni u \longrightarrow \hat{u} = (h_1^{(4)}, h_1^{(3)}, h_1^{(2)}, h_2^{(2)}, h_1^{(1)}, h_2^{(1)}, h_3^{(1)}, h_4^{(1)})^T \in \mathbb{C}^8$$

- wie wir oben gesehen haben, sind die $h_i^{(1)}$ die Längen der Projektionen von u auf die Orthonormalvektoren

$$\begin{aligned} w_1^{(1)} &= \frac{1}{\sqrt{2}}(1, -1, 0, 0, 0, 0, 0, 0)^T \\ w_2^{(1)} &= \frac{1}{\sqrt{2}}(0, 0, 1, -1, 0, 0, 0, 0)^T \\ w_3^{(1)} &= \frac{1}{\sqrt{2}}(0, 0, 0, 0, 1, -1, 0, 0)^T \\ w_4^{(1)} &= \frac{1}{\sqrt{2}}(0, 0, 0, 0, 0, 0, 1, -1)^T \end{aligned}$$

- aus der Rekursion ergibt sich direkt, dass die $h_i^{(2)}$ die Längen der Projektionen von u auf

$$\begin{aligned} w_1^{(2)} &= \frac{1}{2}(1, 1, -1, -1, 0, 0, 0, 0)^T \\ w_2^{(2)} &= \frac{1}{2}(0, 0, 0, 0, 1, 1, -1, -1)^T \end{aligned}$$

$h_1^{(3)}$ die Länge der Projektionen von u auf

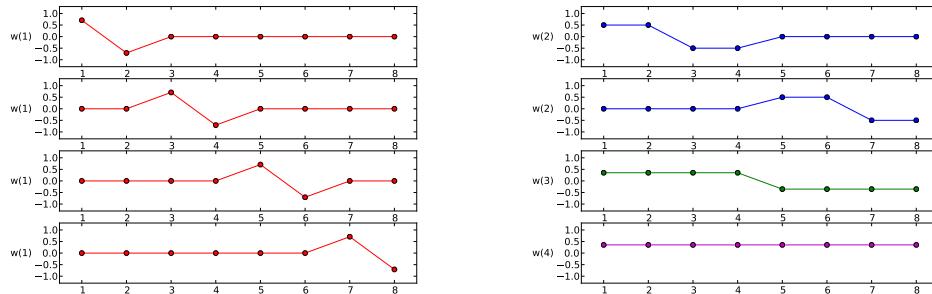
$$w_1^{(3)} = \frac{1}{\sqrt{8}}(1, 1, 1, 1, -1, -1, -1, -1)^T$$

und $h_1^{(4)}$ die Länge der Projektionen von u auf

$$w_1^{(4)} = \frac{1}{\sqrt{8}}(1, 1, 1, 1, 1, 1, 1, 1)^T$$

sind

- man rechnet direkt nach, dass die $w_i^{(j)}$ eine Orthonormalbasis in \mathbb{C}^8 bilden



- \hat{u} ist also die Transformation von u in diese Orthonormalbasis
- damit gilt $\|\hat{u}\| = \|u\|$
- die Basen $w_i^{(j)}$ können dabei wie folgt interpretiert werden:

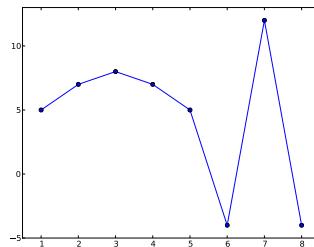
- der obere Index j beschreibt, auf welcher Skala sich die Basen verändern, bei $j = 1$ elementweise, bei $j = 2$ paarweise, bei $j = 3$ in Viererblöcken etc.
- damit ist $j = 1$ die feinste Skala, für $j > 1$ erhalten wir gröbere Skalen ("Auflösung" wird von Stufe zu Stufe halbiert)
- damit ist die Skala j das Analogon zur Frequenz bei der DFT (hohe Frequenz ist feine Skala, niedrige Frequenz grobe)
- innerhalb einer Skala erhalten wir alle Basen durch "Verschiebung" der jeweils ersten
- der untere Index i gibt also Auskunft darüber, wo in u Variationen auf der entsprechenden Skala vorhanden sind (diese "räumliche" Information gibt es bei der DFT nicht)
- Basen auf einer gröberen Skala erhält man einfach durch "Strecken" und Skalieren der Basen auf der nächst feineren Skala
- durch die Transformation von u auf

$$\hat{u} = (h_1^{(4)}, h_1^{(3)}, h_1^{(2)}, h_2^{(2)}, h_1^{(1)}, h_2^{(1)}, h_3^{(1)}, h_4^{(1)})^T$$

erhalten wir also eine **Multiskalarendarstellung** von u , d.h. die $h_i^{(j)}$ geben uns Informationen darüber, wie u sich auf verschiedenen Skalen verhält

Beispiel 310.

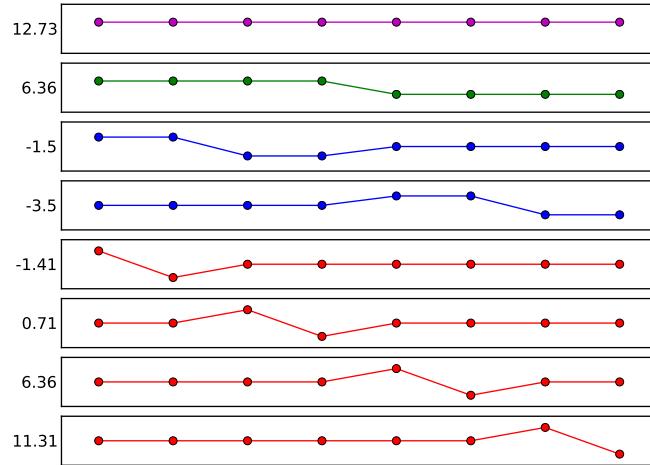
- wir betrachten den Vektor $u = (5, 7, 8, 7, 5, -4, 12, -4)^T$



- als Transformierte erhalten wir

$$\begin{aligned}\hat{u} &= (h_1^{(4)}, h_1^{(3)}, h_1^{(2)}, h_2^{(2)}, h_1^{(1)}, h_2^{(1)}, h_3^{(1)}, h_4^{(1)})^T \\ &\approx (12.73, 6.36, -1.5, -3.5, -1.41, 0.71, 6.36, 11.31)^T\end{aligned}$$

also



- ▶ auf der feinsten Skala (rot) sind die beiden letzten Koeffizienten groß, d.h. dort haben wir in u starke lokale Änderungen
- ▶ auf der nächst größeren Skala (blau) tut sich wenig
- ▶ die beiden größten Skalen liefern wieder einen nennenswerten Beitrag
- ▶ insgesamt heißt das, dass u sich im Wesentlichen aus einem konstanten Anteil (magenta), einem niederfrequenten leicht fallenden Anteil (grün) und einem stark veränderlichen Anteil (letzte rote Basis) zusammensetzt

- die Idee zur Konstruktion solcher Basen geht zurück auf Haar (1910) [7]
- Haar hat damit Orthonormalsysteme in $L^2(\mathbb{R})$ konstruiert
- wir betrachten hier nur \mathbb{C}^N mit $N = 2^s$
- die Transformation von oben nennt man **Diskrete Wavelettransformation (DWT)** mit Haar-Wavelet

Satz 311 (DWT Haar). Es sei $u \in \mathbb{C}^N$, $N = 2^s$. Dann ist der durch

$$\begin{aligned} l^{(0)} &= u, \\ l_i^{(j)} &= \frac{l_{2i-1}^{(j-1)} + l_{2i}^{(j-1)}}{\sqrt{2}}, & h_i^{(j)} &= \frac{l_{2i-1}^{(j-1)} - l_{2i}^{(j-1)}}{\sqrt{2}}, & i &= 1, \dots, 2^{s-j}, & j &= 1, \dots, s \\ h_1^{(s+1)} &= l_1^{(s)} \end{aligned}$$

definierte Vektor

$$\hat{u} = (h_1^{(s+1)}, \quad h_1^{(s)}, \quad h_1^{(s-1)}, h_2^{(s-1)}, \dots, h_1^{(1)}, \dots, h_{2^{s-1}}^{(1)})^T$$

die Transformation von u in die Orthonormalbasis $w_i^{(j)}$ mit

$$\begin{aligned} w_1^{(s+1)} &= \frac{1}{\sqrt{2^s}} (1, \dots, 1)^T, \\ w_{i,k}^{(j)} &= \frac{1}{\sqrt{2^j}} \begin{cases} 1 & 1 \leq k - (i-1)2^j \leq 2^{j-1} \\ -1 & 2^{j-1} + 1 \leq k - (i-1)2^j \leq 2^j \\ 0 & \text{sonst} \end{cases}, & i &= 1, \dots, 2^{s-j}, & j &= 1, \dots, s. \end{aligned}$$

- für die inverse Transformation (**IDWT**) erhalten wir analog

Satz 312 (IDWT Haar). Aus

$$\hat{u} = (h_1^{(s+1)}, \quad h_1^{(s)}, \quad h_1^{(s-1)}, h_2^{(s-1)}, \dots, h_1^{(1)}, \dots, h_{2^{s-1}}^{(1)})^T$$

kann mit

$$\begin{aligned} l_1^{(s)} &= h_1^{(s+1)} \\ l_{2i-1}^{(j-1)} &= \frac{l_i^{(j)} + h_i^{(j)}}{\sqrt{2}}, \quad l_{2i}^{(j-1)} = \frac{l_i^{(j)} - h_i^{(j)}}{\sqrt{2}}, \quad i = 1, \dots, 2^{s-j}, \quad j = s, \dots, 1 \\ u &= l^{(0)} \end{aligned}$$

u rekonstruiert werden.

Bemerkung 313.

- ◊ der Aufwand für DWT und IDWT ist $\mathcal{O}(n)$

Beispiel 314.

- wir wenden DWT nach Haar zur verlustbehafteten Kompression des Audiosignals aus Beispiel 309 an
- zunächst wird u per DWT in \hat{u} transformiert
- in \hat{u} werden wieder die betragsmäßig kleinen Komponenten (aufsteigend nach Größe, angefangen bei den kleinsten) eliminiert, so dass

$$\|\hat{u}_c\|_2 \geq 0.98 \|\tilde{u}\|_2$$

gilt, d.h. 98% der "Signalenergie" soll nach der Kompression erhalten bleiben

- die Anzahl der Nicht-Null-Elemente von \hat{u}_c beträgt dann $N_c = 136\,476$ statt $N = 1\,048\,576$ (Faktor ≈ 7.68)
- aus \hat{u}_c kann durch IDWT ein Audiosignal u_c rekonstruiert werden
- im Hörvergleich zum Original u ist u_c wieder sehr brauchbar

Zusammenfassung.

- die DWT transformiert Vektoren (Signale) in eine neue Orthonormalbasis
- die neuen Koeffizienten liefern sowohl "Frequenzinformation" (verschiedene Skalen) als auch "räumliche Information" (innerhalb der selben Skala)
- Haar-Wavelets sind die einfachsten Wavelets, es gibt mittlerweile einen regelrechten "Wavelet-Zoo"
- der Aufwand für DWT und IDWT ist $\mathcal{O}(n)$
- Wavelets werden z.B. bei JPEG2000 eingesetzt

Literaturverzeichnis

- [1] Peter Deuflhard, Andreas Hohmann, *Numerische Mathematik, Eine algorithmisch orientierte Einführung*, deGruyter, 2002
- [2] Martin Hanke-Bourgeois, *Grundlagen der numerischen Mathematik und des wissenschaftlichen Rechnens*, Teubner, 2002
- [3] Josef Stoer, *Numerische Mathematik*, Band 1, Springer, 1999
- [4] Josef Stoer, Roland Bulirsch, *Numerische Mathematik*, Band 2, Springer, 1999
- [5] Poul Erik Frandsen, Kristian Jonasson, Hans Bruun Nielsen, Ole Tingleff, *Unconstrained Optimization*, Lecture Notes, Technical University of Denmark, 2004
- [6] <https://storage.googleapis.com/springer-extras/zip/2002/978-3-642-62564-0.zip> (24.03.2024)
- [7] Alfred Haar, *Zur Theorie der orthogonalen Funktionensysteme*, Mathematische Annalen 69, Nr. 3, 1910, S. 331-371

Index

- $\frac{3}{8}$ -Regel, 216
- A-konjugiert, 253
- a-posteriori Abschätzung, 147
- a-priori Abschätzung, 147
- Abstiegsrichtung, 246
- Abstiegsverfahren
- striktes, 246
- Aitken-Neville-Schema, 233
- Äquilibrieren, 59
- Armijo-Bedingung, 248
- asymptotische Entwicklung, 228
- Ausgleichsgerade, 189
- Ausgleichsproblem
- nichtlineares, 268
- Auslöschung, 53
- average case analysis, 68
- Backtracking, 248
- Banachscher Fixpunktsatz, 147
- Basis, 42
- BFGS-Update, 266
- für Inverse, 266
- Broyden-Rang-1-Update, 265
- CG-Verfahren, 112, 253
- Fletcher-Reeves, 254
 - Polak-Ribiére, 254
 - vorkonditioniertes, 113
- CGLS-Verfahren, 196
- charakteristisches Polynom, 120
- Cholesky-Zerlegung, 32
- rationale, 32
- Deflation, 133
- diagonalisierbar, 121
- differentielle Fehleranalyse, 49, 50
- Diskrepanz-Prinzip nach Morozov, 211
- Diskretisierungsparameter, 51
- dividierte Differenzen, 168
- Downsampling, 290
- Dreiecksgestalt, 15
- Eigenfrequenzen, 118
- Eigenvektor, 121
- Eigenwert, 120
- mehrfacher, 121
- Eindeutigkeit, 36
- Einschließungsverfahren, 136, 138
- Einzelschritt-Verfahren, *siehe* Gauß-Seidel-Verfahren
- Ersatzproblem, 1, 7, 51
- euklidische Länge, 13
- Existenz, 36
- Exponent, 42
- Exponentialdarstellung, 41
- Extrapolation, 233
- Bulirsch-Folge, 236
 - Romberg-Folge, 236
- Fass-Regel, *siehe* $\frac{3}{8}$ -Regel
- Fehler, 38, 87
- absoluter, 38
 - akkumulierter, 41, 55
 - Akkumulierter Rundungsfehler, 56
 - Eingabefehler, 40, 55
 - lokaler, 55
 - relativer, 38
 - unvermeidbarer, 40
 - Verfahrensfehler, 40, 56
- Fehlerfortpflanzungsformeln, 49, 50
- Fehlerkriterium
- gemischtesgemischten Fehlerkriterium, 39
- Fehlerschätzer, 240
- a-posteriori, 60
 - a-priori, 60
- Fehlerverstärkung, 55
- Festkommadarstellung, 41
- FFT
- Algorithmus von Cooley und Tukey, 287
- Fixpunkteigenschaft, 89
- Fließkommadarstellung, 41
- normalisierte, 42
- Fließkommazahl, 42
- Floating-Point, 1
- Fourierreihe
- komplexe, 274
- Fouriertransformation
- diskrete (DFT), 283
 - diskrete inverse (IDFT), 283
- Frobeniusmatrizen, 19
- Frobeniusnorm, 14
- Gauß-Newton-Verfahren, 269
- Gauß-Quadratur, 224
- Gewichte, 224
- Gauß-Seidel-Verfahren, 100

symmetrisches, 107
 Gesamtschritt-Verfahren, *siehe* Jacobi-Verfahren
 Givens-Verfahren, 73
 Gradient, 244
 gut gestellt, 37
 Hesse-Matrix, 244
 Hessenberg-Form, 131
 Hilbertmatrix, 202
 Horner-Schema, 165
 Householder-Verfahren, 80
 IEEE 754, 44

- double, 44
- single, 44

 instabil, 55
 Interpolation, 163

- Zwischenstellen, 163

 Iterationsmatrix, 88
 Iterationsvektor, 88
 iteratives Verfahren, 86

- einstufig, 86
- instationär, 86
- Konvergenz, 87
- mehrstufig, 86
- stationär, 86
- Verfahrensvorschrift, 86

 Jacobi-Matrix, 50, 161
 Jacobi-Verfahren, 89, 97

- relaxiertes, 104

 Kirchhoffsche Regeln, 5
 Knotenregel, 4
 Kondition

- absolute, 46
- relative, 46

 Konditionszahl, 57
 Kontraktion, 146
 Konvergenz

- lineare, 153
- lokale, 135, 146
- Ordnung, 153
- quadratische, 154, 158
- superlineare, 154

 Konvergenzuntersuchung, 52
 Kurve, 184
 Kurvenparameter, 184
 Ladungserhaltung, 4
 Lagrange-Polynom, 166
 Levenberg-Marquardt-Verfahren, 271
 Liniensuche, 248

- exakte, 248

 Soft Linesearch, 248
 Mantisse, 42
 Maschenregel, 4
 Maschinengenauigkeit, 43
 Matrixnorm, 14

- induzierte, 14

 Minimum

- globales, 242
- lokales, 242

 Modes, 119
 Multiskalarendarstellung, 292
 Nachiteration, 103
 Nadelimpuls, 238
 Newton-Cotes-Formeln, 218

- adaptive, 239
- Gewichte, 214, 215
- summierte, 220

 Newton-Integration, *siehe* $\frac{3}{8}$ -Regel
 Newton-Interpolationspolynom, 168
 Newton-Verfahren, 142

- gedämpftes, 258
- Levenberg-Marquardt-Typ, 262

 Normalgleichungen, 190
 Nullstelle

- q -fache, 159
- einfache, 158
- mehrfache, 158

 Numerische Simulation, 3
 Ohmscher Widerstand, 4
 Optimierung

- nicht restringierte, 242

 Orthogonalpolynome, 222
 orthonormale Matrix, 70
 PCG-Verfahren, 113
 Penrose-Axiome, 197
 Permutationsmatrix, 24
 Picard-Iteration, 145
 Polygonzug, 171
 Pseudo-Inverse, 197
 QR-Verfahren, 130
 Quasi-Newton-Bedingung, 265
 Quasi-Newton-Verfahren, 264
 rückwärts einsetzen, 15
 Rückwärtsanalyse, 60, 67
 Regula-Falsi-Verfahren, 138
 Residueniteration, 87
 Residuum, 60, 87
 Rosenbrock-Funktion, 251
 Rundung

- nearest, 43
- nearest even, 45

 Satz von

- Banach, 147

Gershgorin, 121
 Kahan, 106
 Ostrowski, Reich, 106
 Prager, Oettli, 60
 Stein, Rosenberg, 102
 schlecht gestellt, 37
 schlecht konditioniert, 46
 Sekanten-Verfahren, 140
 Shift, 133
 Simpson-Regel, 216
 Streifenzahl, 221
 summierte, 221
 Singulärwerte, 198
 Singulärwertzerlegung, 198
 abgeschnittene, 207
 Skalarprodukt, 12
 SOR-Verfahren, 106
 Spalten-Pivot-Suche, 65
 Spaltensummennorm, 14
 spd, 30
 Spektralnorm, 14
 Spektralradius, 14, 91
 Spline
 Abschlussbedingungen, 171
 B-Splines, 178
 Basis-Splines, 179
 Hermite, 172
 Momente, 173
 natürlicher, 172
 periodischer, 172
 Polynomspline, 171
 Unterteilung, 178
 Stützstellen, 214
 stabil, 55
 stationärer Punkt, 244
 Steepest-Descent-Verfahren, 111, 249
 vorkonditioniertes, 111
 stetige Abhängigkeit, 36, 37
 Strafterm, 259
 streng diagonaldominant, 98
 Tikhonov-Regularisierung, 209
 transponierte Matrix, 12
 Trapez-Regel, 216
 summierte, 220
 Trust-Region, 260
 Trust-Region-Verfahren
 Gain-Faktor, 262
 TSVD, *siehe* abgeschnittene Singulärwertzerlegung
 Validierung, 3
 Vandermonde-Matrix, 164
 Vektoriteration, 123
 Vertrauensbereich, *siehe* Trust-Region
 Vertrauensradius, 260
 Vielfachheit