Einführung in die Numerik

Ole Klein

Universität Heidelberg
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Im Neuenheimer Feld 205, D-69120 Heidelberg
email: ole.klein@iwr.uni-heidelberg.de

basierend auf dem Vorlesungsskript von Peter Bastian

Sommersemester 2018

Abschnitt 1

1 Organisatorisches
- Vorlesung
- Programmieren
- Übungsbetrieb
- Scheinkriterien und Klausur
- Literatur und weiterführende Vorlesungen

Vorlesung

- Ole Klein, INF 205, R. 1/224, Sprechstunde nach Vereinbarung (ole.klein@iwr.uni-heidelberg.de)
- Übungsleitung: Marian Piatkowski (marian.piatkowski@iwr.uni-heidelberg.de)
- Vorlesungstermine: Di, Do 14–16, INF 227, HS1, **Beginn c.t.**
- Webseite zur Vorlesung (Materialien, Übungsaufgaben, . . .)

`https://conan.iwr.uni-heidelberg.de/teaching/numerik0_ss2018/`

- Vorlesung bestehend aus Folien und Tafelanschrieb.
- Inhalt der Folien ist als PDF auf Webseite erhältlich.

Programmieren

- Vorlesung und praktische Programmierübungen verwenden Programmiersprache C++.
- Zur Vorlesung werden C++-Klassen für Matrizen, Vektoren und Zeitmessung zur Verfügung gestellt.
- Wir empfehlen (und unterstützen) Programmieren in einer **Linux-Umgebung**.
- Mac user: analog zu Linux
- Windows user (**kein Support!**):
  - Virtuelle Maschine mit Linux (`http://www.virtualbox.org`)
  - Optional bieten wir Kopiermöglichkeit eines USB-Sticks

Programmierkurs

Einsteigerkurs für Programmieranfänger:

- Montags, 14–16 Uhr, INF 205, CIP Pool (dritter Stock)
- Beginnt nächste Woche (23.4.)
- Hilfe beim Einstieg in das Programmieren (Linux / C++)
- **kein** Ersatz für einen vollwertigen Programmierkurs
- **keine** Unterstützung bei den eigentlichen Aufgaben

Interessenten bitte in Liste eintragen!

Übung

Begleitend zur Vorlesung finden Übungen statt:

- Pro Woche wird ein Übungsblatt mit theoretischen und praktischen Übungen (Programmierübungen) ausgegeben.
- Punkteverhältnis ca. 70% (theoretisch) zu 30% (praktisch).
- Abgabe der praktischen Übungen per Mail an die Tutoren.
- Eine erfolgreiche Teilnahme an den Übungen ist Voraussetzung für die Teilnahme an der Klausur.
- Eine Abgabe in Gruppen von zwei bis drei Teilnehmern ist möglich und ausdrücklich erwünscht!

Übungsbetrieb

- Ausgabe Übungsblätter immer donnerstags nach der Vorlesung
- Ausgabe erstes Blatt diese Woche (19.4.)
- Bearbeitungszeit jeweils eine Woche
- Abgabe immer donnerstags im Hörsaal
- Abgabe muss **vor** Beginn der Vorlesung (c.t.) erfolgt sein!
- Beginn der Übungsgruppen ab 23.4.

Übungsgruppen

- **Anmeldung** über *Muesli*:
  `https://www.mathi.uni-heidelberg.de/muesli/`
  ist geöffnet
- Derzeit 6 Termine geöffnet
- **Anmeldung bis Donnerstag, 19.4.2018, 20:00 Uhr**
- Zuteilung erfolgt am Freitag

Scheinkriterien

Klausur:
Die Scheinvergabe und Benotung erfolgt aufgrund der Ergebnisse der verpflichtenden Klausur.

Klausurtermin: Donnerstag, 26.7.2018, 14–16 Uhr

Zulassung zur Klausur:

- 50 % der Punkte aus allen Übungen
- Vorführung mindestens einer Lösung in der Übungsgruppe (einmal pro **Teilnehmer**, nicht einmal pro Abgabegruppe).
- Wer meint, eine Klausurzulassung zu besitzen, klärt dies bitte in **den ersten beiden Wochen**!

Weiterführende Vorlesungen

- Numerik (gewöhnlicher Differentialgleichungen)
- Numerik partieller Differentialgleichungen
- Parallele Löser für große Gleichungssysteme
- Kontinuumsmechanik
- Strömungsmechanik
- Algorithmische Optimierung I/II
- Objektorientiertes Programmieren im Wissenschaftlichen Rechnen

Literatur

- Skript von Prof. Rannacher: `http://numerik.iwr.uni-heidelberg.de/~lehre/notes/num0/numerik0.pdf`
- Inzwischen auch als e-Book erhältlich: `http://heiup.uni-heidelberg.de/catalog/book/206`
- Klassiker: Stoer, Numerische Mathematik 1, Springer-Verlag
- H. R. Schwarz, N. Klöckler: Numerische Mathematik, Teubner-Verlag
- E. Süli, D. Mayers: An introduction to numerical analysis, Cambridge University Press

Abschnitt 2

❷ Motivation

- Die Wissenschaftliche Methode
- Ein Beispiel
- Lösungsansätze
- Numerische Resultate
- Weitere Anwendungen
- Ausblick

Warum Numerische Mathematik?

Die Wissenschaftliche Methode

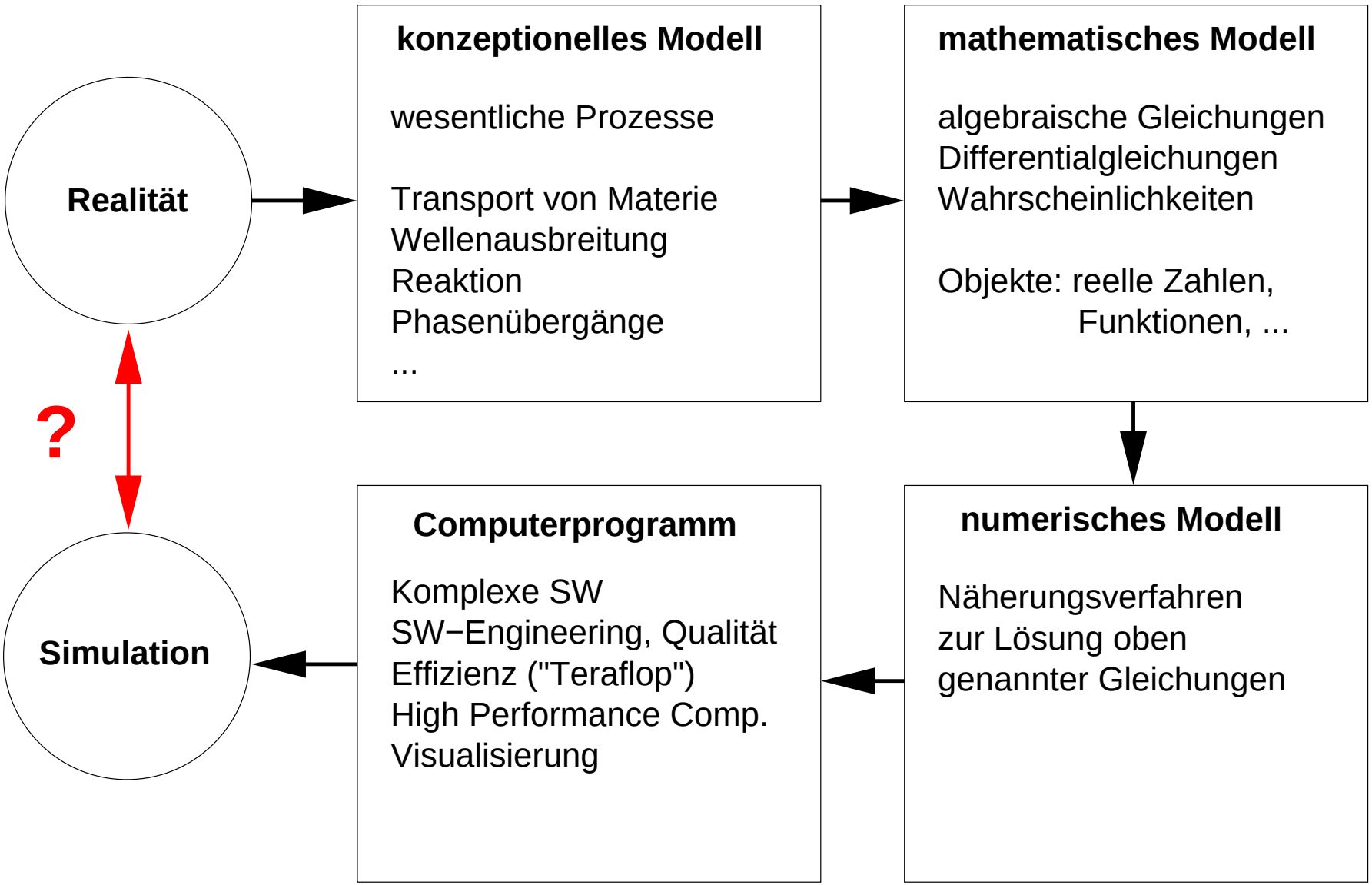
- Experiment: Beobachte (und messe) was passiert.
- Theorie: Versuche die Beobachtung mit Hilfe von Modellen zu erklären.
- Theorie und Experiment werden sukzessive verfeinert und verglichen, bis eine akzeptable Übereinstimmung vorliegt.
- In Naturwissenschaft und Technik liegen Modelle oft in Form mathematischer Gleichungen vor, z. B. gewöhnliche oder partielle Differentialgleichungen.
- Oft können die Modellgleichungen nicht geschlossen (mit Papier und Bleistift, oder Mathematica ...) gelöst werden.

$\Rightarrow$ **Numerische Simulation und Optimierung**

Simulation

- Simulation: Gleichungen numerisch lösen.
  - Undurchführbare Experimente ermöglichen (z. B. Galaxienkollisionen).
  - Einsparung teurer Experimente (z. B. Windkanal).
  - Parameterstudien schneller durchführbar.
  - (Automatische) Optimierung von Prozessen.
  - Identifikation von Modellparametern aus Messungen.
  - Abschätzung von Unsicherheiten.
- Vielfältiger Einsatz in Naturwissenschaft, Technik und Industrie: Strömungsberechnung (Wetter, Klima), Festigkeit von Bauwerken ...
- Grundlage für alle diese Anwendungen sind numerische Algorithmen!
- Auch Informatiker sind beteiligt: Software-Engineering, High-Performance (Parallel) Computing, ...

Wissenschaftliches Rechnen



Fehlerquellen

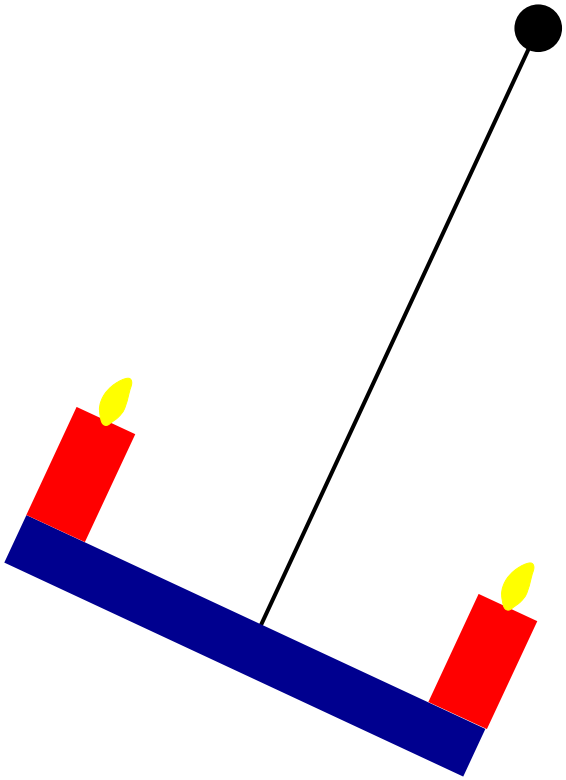
Unterschiede zwischen Experiment und Simulation haben verschiedene Gründe:

- *Modellfehler*: Ein relevanter Prozess wurde nicht oder ungenau modelliert (Temp. konstant, Luftwiderstand vernachlässigt, ...)
- *Datenfehler*: Messungen von Anfangsbedingungen, Randbedingungen, Werte für Parameter sind fehlerbehaftet.
- *Abschneidefehler*: Abbruch von Reihen oder Iterationsverfahren, Approximation von Funktionen (z.B. stückweise Polynome).
- *Rundungsfehler*: Reelle Zahlen werden im Rechner genähert dargestellt.

Untersuchung von Rundungsfehlern und Abschneidefehlern ist ein zentraler Aspekt der Vorlesung!

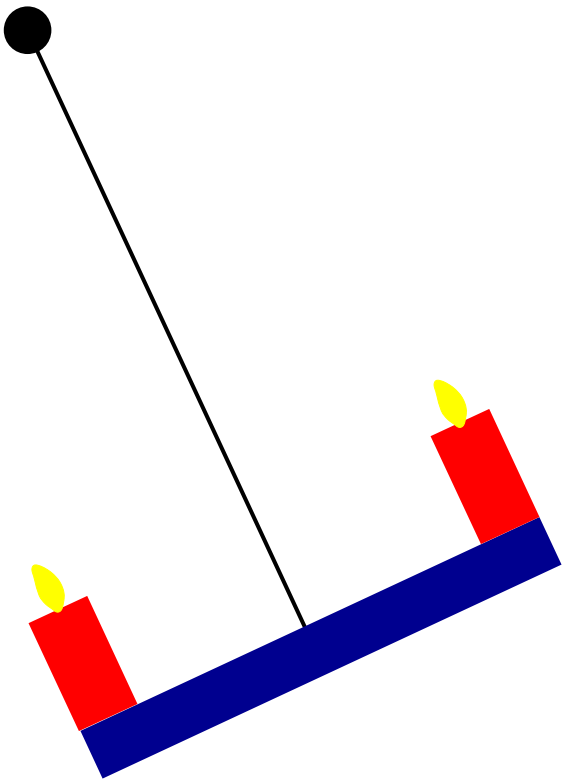
Pisa, 1582

Der Student Galileo Galilei sitzt in der Kirche und ihm ist langweilig. Er beobachtet den langsam über ihm pendelnden Kerzenleuchter und denkt: „Wie kann ich nur die Bewegung dieses Leuchters beschreiben?“.



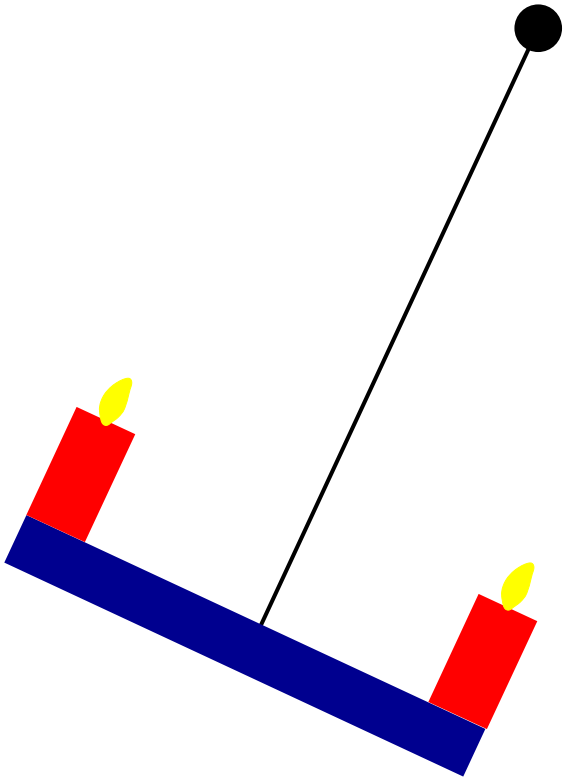
Pisa, 1582

Der Student Galileo Galilei sitzt in der Kirche und ihm ist langweilig. Er beobachtet den langsam über ihm pendelnden Kerzenleuchter und denkt: „Wie kann ich nur die Bewegung dieses Leuchters beschreiben?“.



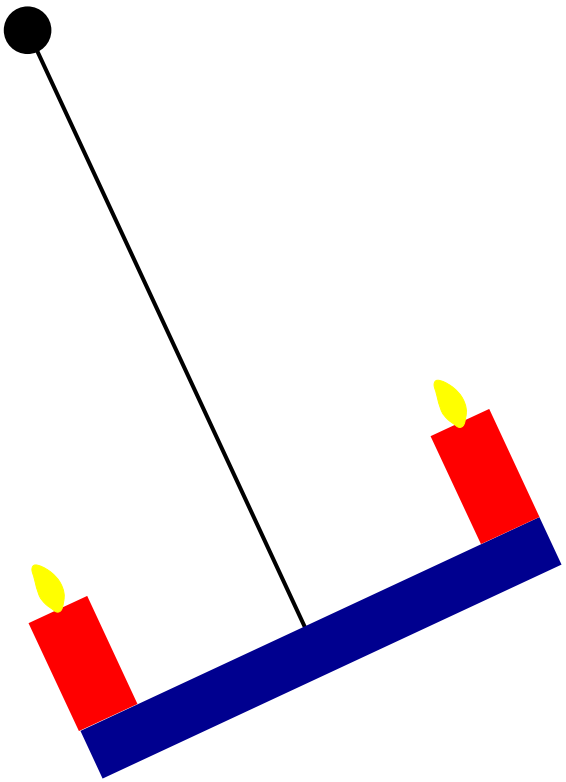
Pisa, 1582

Der Student Galileo Galilei sitzt in der Kirche und ihm ist langweilig. Er beobachtet den langsam über ihm pendelnden Kerzenleuchter und denkt: „Wie kann ich nur die Bewegung dieses Leuchters beschreiben?“.



Pisa, 1582

Der Student Galileo Galilei sitzt in der Kirche und ihm ist langweilig. Er beobachtet den langsam über ihm pendelnden Kerzenleuchter und denkt: „Wie kann ich nur die Bewegung dieses Leuchters beschreiben?“.

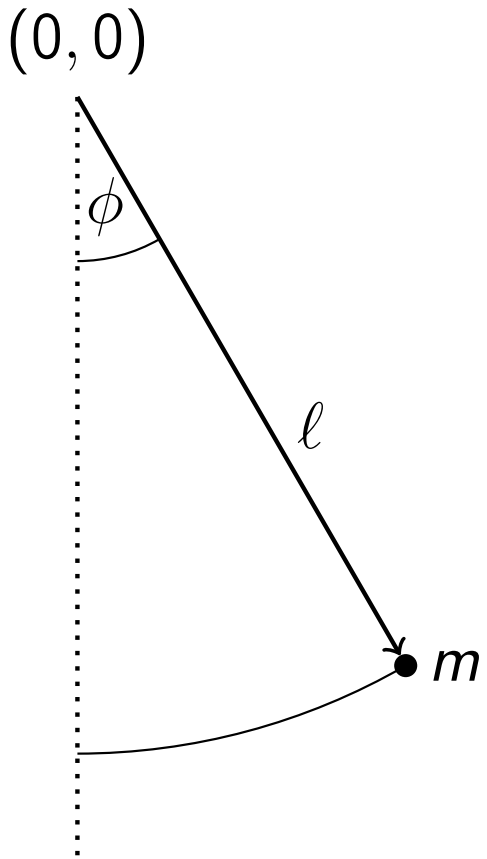


Konzeptionelles Modell

Welche Eigenschaften (physikalischen Prozesse) sind für die gestellte Frage relevant?

- Leuchter ist ein Massenpunkt mit der Masse m.
- Der Faden der Länge ℓ wird als rigide und masselos angenommen.
- Der Luftwiderstand wird vernachlässigt.

Nun entwickle mathematisches Modell.



Kräfte

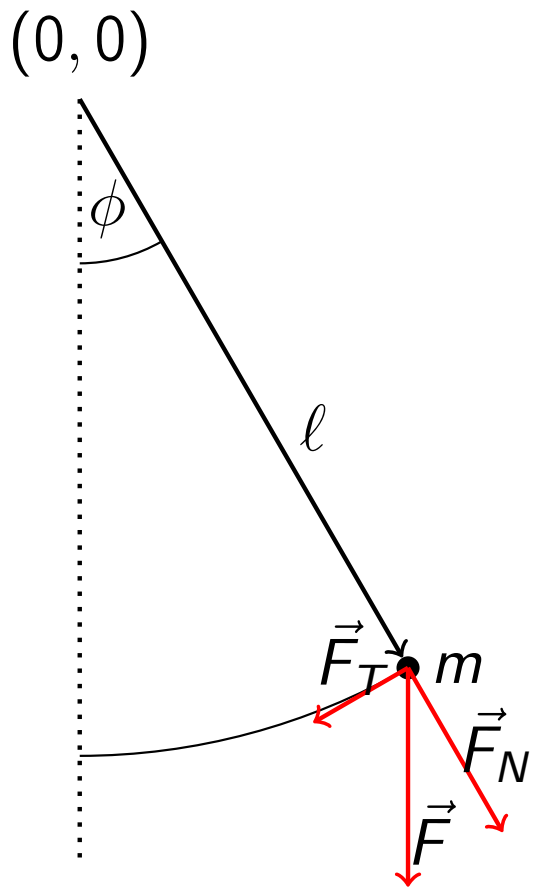
- *Annahme*: Pendel läuft auf Kreisbahn: Nur *Tangentialkraft* ist relevant.
- Tangentialkraft bei Auslenkung ϕ:

$$\vec{F}_T(\phi) = -m\,g\,\sin(\phi)\begin{pmatrix}\cos(\phi)\\ \sin(\phi)\end{pmatrix}.$$

- Also etwa:

$$\vec{F}_T(0) = -mg\begin{pmatrix}0\\0\end{pmatrix},$$

$$\vec{F}_T(\pi/2) = -mg\begin{pmatrix}0\\1\end{pmatrix}.$$



- Vorzeichen kodiert Richtung.

Weg, Geschwindigkeit, Beschleunigung

- Weg $s(t)$, Geschwindigkeit $v(t)$, Beschleunigung $a(t)$ erfüllen:

$$v(t) = \frac{ds(t)}{dt}, \qquad a(t) = \frac{dv(t)}{dt}.$$

- Für den zurückgelegten Weg (mit Vorzeichen!) gilt $s(t) = \ell\phi(t)$.
- Also für die Geschwindigkeit

$$v(t) = \frac{d\, s(\phi(t))}{dt} = \frac{d\, \ell\phi(t)}{dt} = \ell\frac{d\phi(t)}{dt}$$

- und die Beschleunigung

$$a(t) = \frac{d\, v(\phi(t))}{dt} = \ell\frac{d^2\phi(t)}{dt^2}.$$

Bewegungsgleichung

- Einsetzen in das 2. Newton'sche Gesetz $m\,a(t) = F(t)$ liefert nun:

$$m\ell \frac{d^2\phi(t)}{dt^2} = -mg\sin(\phi(t)) \qquad \forall t > t_0.$$

- Die Kraft ist hier skalar (vorzeichenbehafteter Betrag der Tangentialkraft), da wir nur den zurückgelegten Weg betrachten.
- Ergibt *gewöhnliche Differentialgleichung 2. Ordnung* für die Auslenkung $\phi(t)$:

$$\frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\sin(\phi(t)) \qquad \forall t > t_0. \tag{2.1}$$

- Eindeutige Lösung erfordert *zwei* Anfangsbedingungen ($t_0 = 0$):

$$\phi(0) = \phi_0, \qquad \frac{d\phi}{dt}(0) = u_0. \tag{2.2}$$

Lösung bei kleiner Auslenkung

- Allgemeine Gleichung für das Pendel ist schwer „analytisch“ zu lösen.
- Für *kleine* Winkel ϕ gilt

$$\sin(\phi) \approx \phi,$$

  z.B. $\sin(0,1) = 0,099833417$.
- Diese *Näherung* reduziert die Gleichung auf

$$\frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\phi(t).$$

- Ansatz $\phi(t) = A\cos(\omega t)$ liefert mit $\phi(0) = \phi_0$, $\frac{d\phi}{dt}(0) = 0$ dann die aus der Schule bekannte Formel

$$\phi(t) = \phi_0 \cos\left(\sqrt{\frac{g}{\ell}}t\right) \tag{2.3}$$

Volles Modell; Verfahren 1

- Löse das volle Modell mit zwei numerischen Verfahren.
- Ersetze Gleichung zweiter Ordnung durch *zwei* Gleichungen *erster* Ordnung:

$$\frac{d\phi(t)}{dt} = u(t), \qquad \frac{d^2\phi(t)}{dt^2} = \frac{du(t)}{dt} = -\frac{g}{\ell}\sin(\phi(t)).$$

- Ersetze Ableitungen durch Differenzenquotienten:

$$\frac{\phi(t+\Delta t)-\phi(t)}{\Delta t} \approx \frac{d\phi(t)}{dt} = u(t),$$
$$\frac{u(t+\Delta t)-u(t)}{\Delta t} \approx \frac{du(t)}{dt} = -\frac{g}{\ell}\sin(\phi(t)).$$

- Mit $\phi^n = \phi(n\Delta t)$, $u^n = u(n\Delta t)$ erhält man Rekursion (*Euler*):

$$\phi^{n+1} = \phi^n + \Delta t\, u^n \qquad \phi^0 = \phi_0 \qquad (2.4)$$
$$u^{n+1} = u^n - \Delta t\,(g/\ell)\sin(\phi^n) \qquad u^0 = u_0 \qquad (2.5)$$

Volles Modell; Verfahren 2

- Nutze Näherungsformel für die zweite Ableitung, sog. *Zentraler Differenzenquotient*:

$$\frac{\phi(t+\Delta t) - 2\phi(t) + \phi(t-\Delta t)}{\Delta t^2} \approx \frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\sin(\phi(t)).$$

- Auflösen nach $\phi(t+\Delta t)$ ergibt Rekursionsformel ($n \geq 2$):

$$\phi^{n+1} = 2\phi^n - \phi^{n-1} - \Delta t^2\,(g/\ell)\,\sin(\phi^n) \tag{2.6}$$

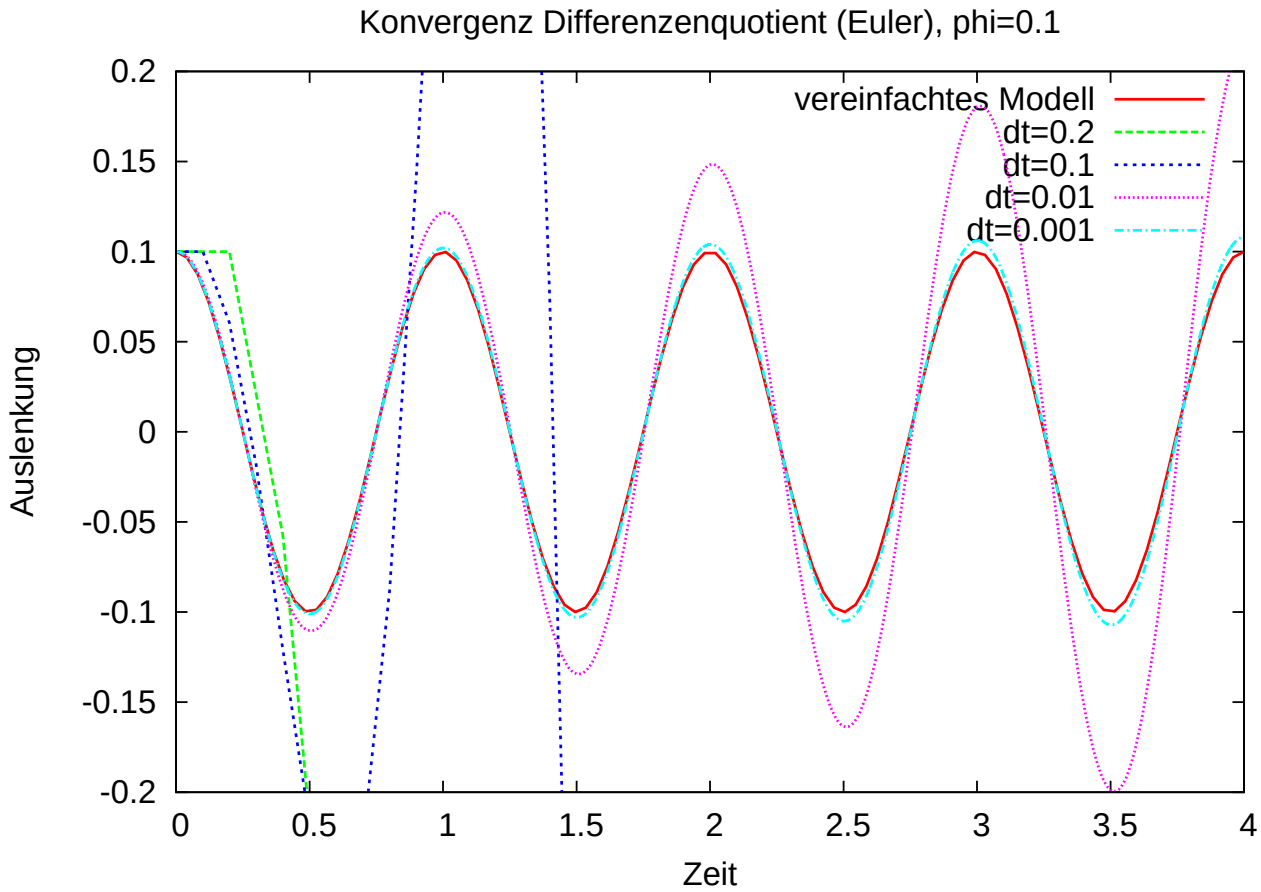
mit der Anfangsbedingung

$$\phi^0 = \phi_0, \qquad \phi^1 = \phi_0 + \Delta t\, u_0. \tag{2.7}$$

(Die zweite Bedingung kommt aus dem Eulerverfahren oben).

Experiment 1: $\phi_0 = 0.1 \approx 5.7°$

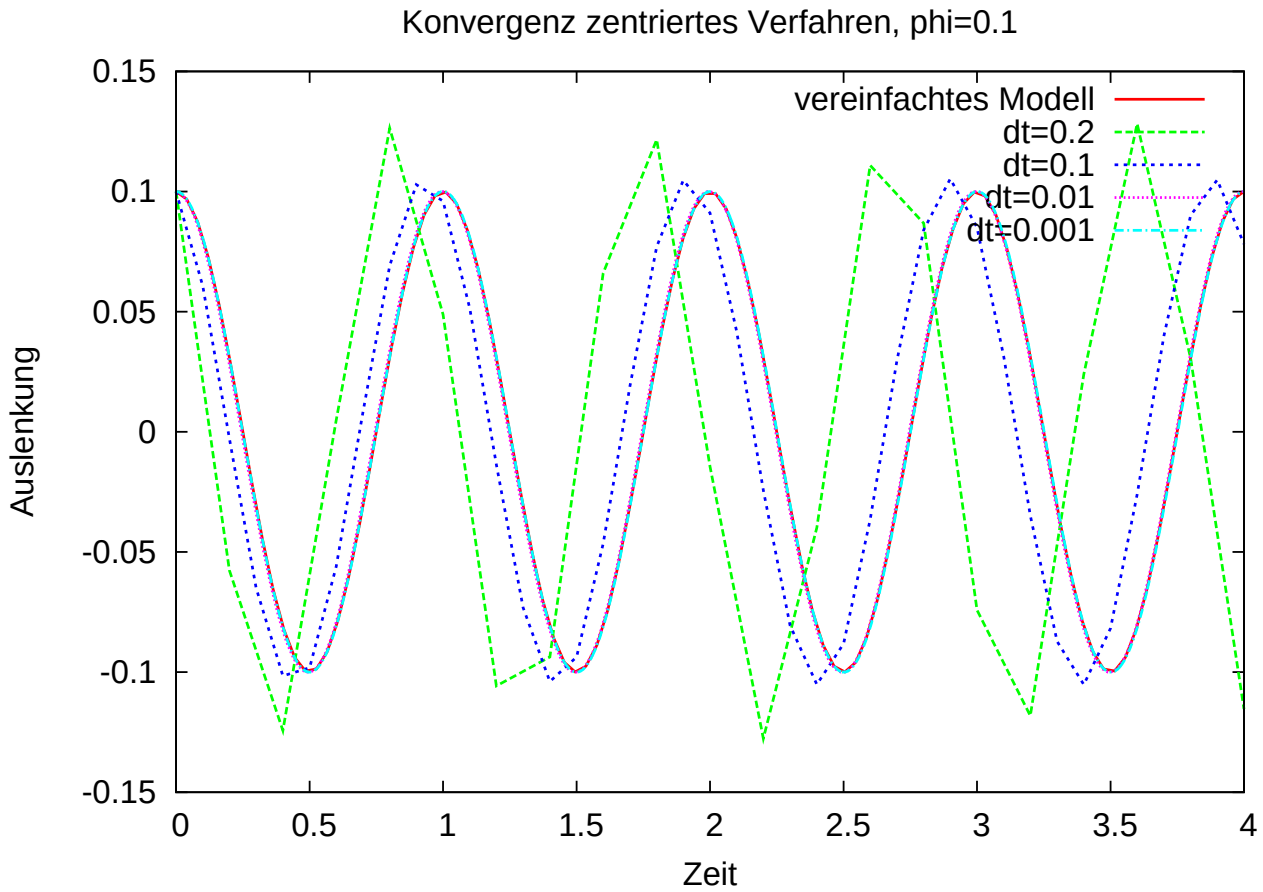
Differenzenquotient (Eulerverfahren)



Für festen Zeitpunkt t und $\Delta t \to 0$ *konvergiert* das Verfahren.
Für festes Δt und $t \to \infty$ wächst der Fehler an.

Experiment 2: $\phi_0 = 0.1 \approx 5.7°$

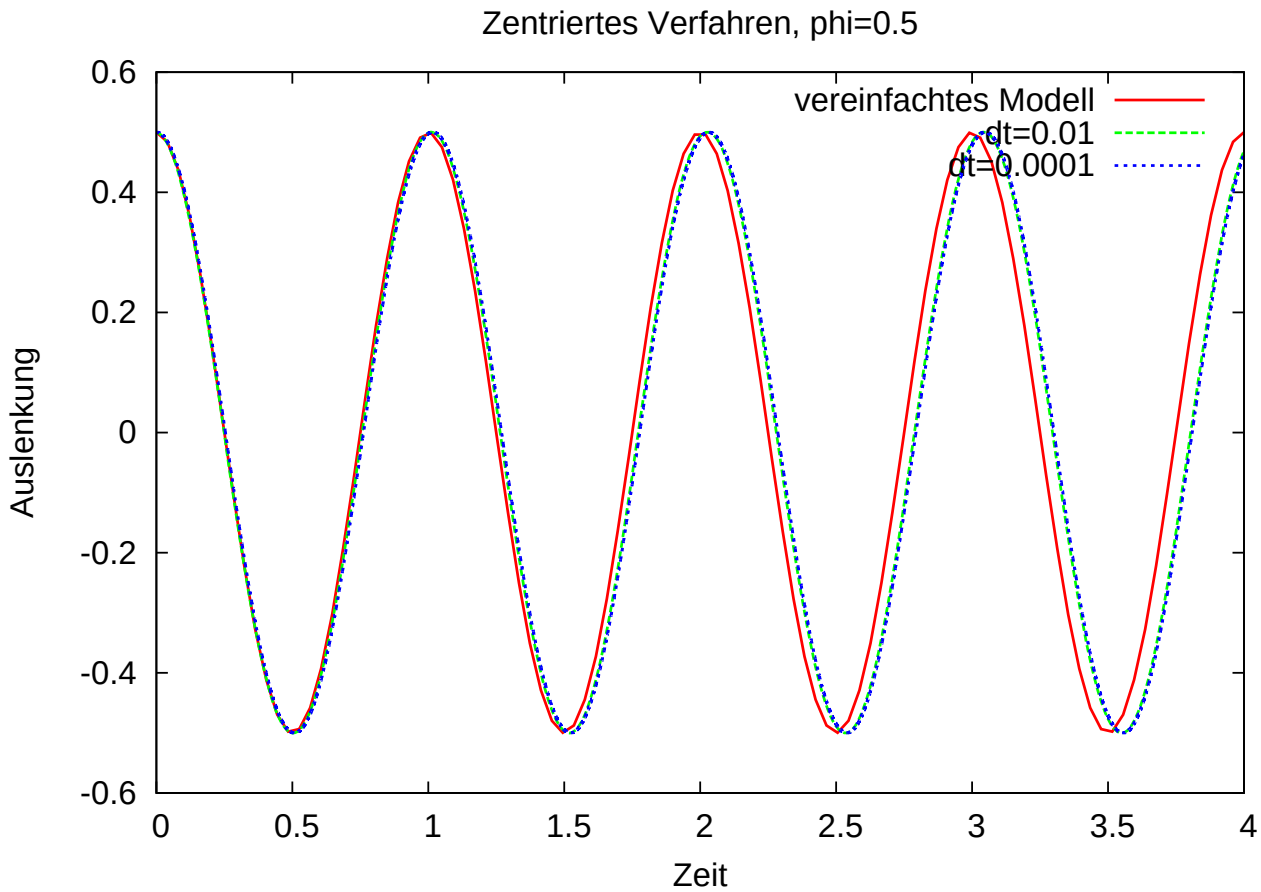
Zentrales Verfahren



Im Unterschied zum expliziten Euler scheint das Verfahren bei festem Δt und $t \to \infty$ nicht unbeschränkt zu wachsen.

Experiment 3: $\phi_0 = 0.5 \approx 28.6°$

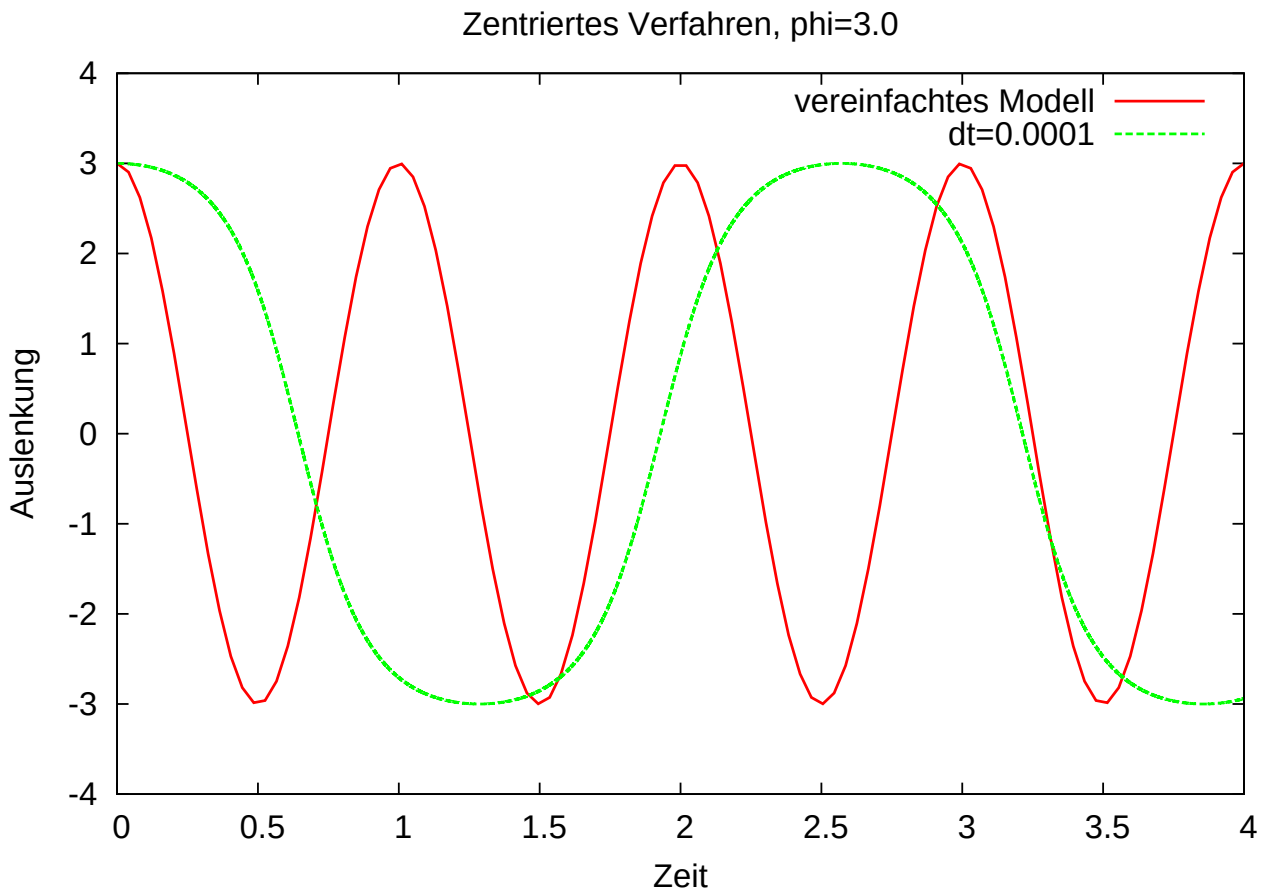
Zentrales Verfahren



Selbst bei $28.6°$ ist die Übereinstimmung noch einigermaßen passabel.

Experiment 4: $\phi_0 = 3.0 \approx 171°$

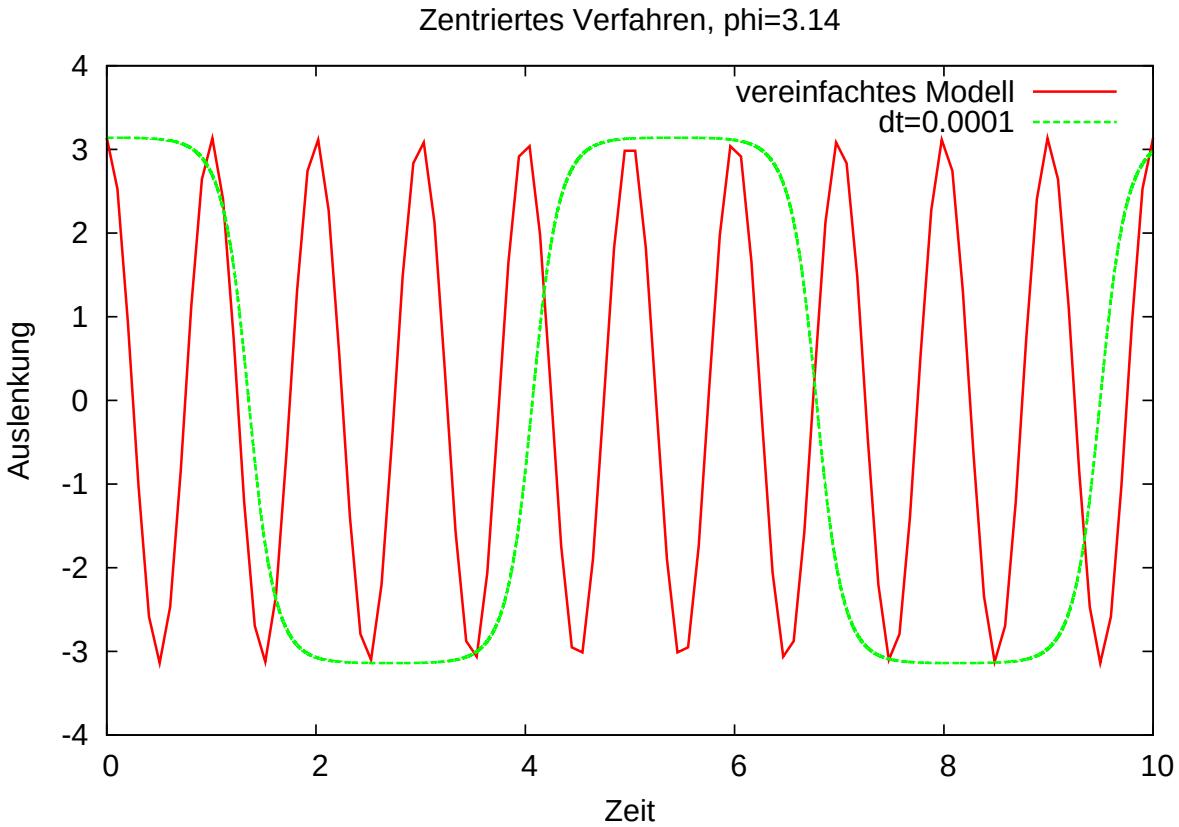
Zentrales Verfahren



Für große Auslenkungen ist das vereinfachte Modell völlig unbrauchbar. Die Form der Schwingung ist kein Kosinus mehr.

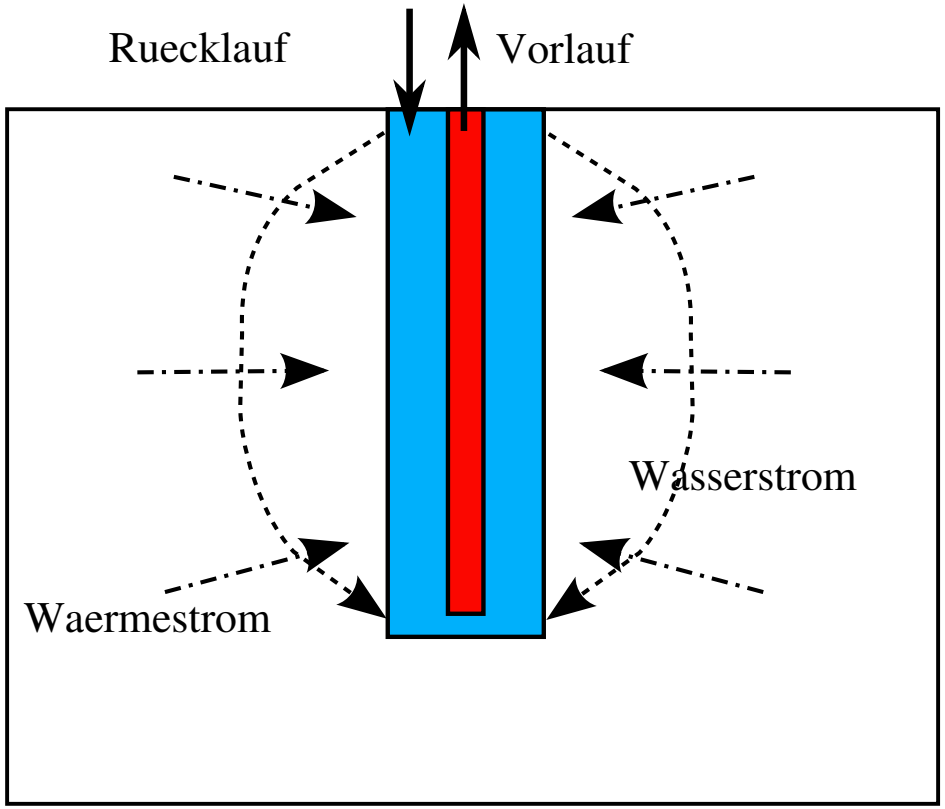
Experiment 5: $\phi_0 = 3.14 \approx 179.91°$

Zentrales Verfahren



Das Pendel wird nahe π immer langsamer. Das ist die Schiffschaukel, die fast auf dem Kopf steht. Wie würde denn die Kurve bei einer umlaufenden Schiffschaukel aussehen?

Eine Geothermieanlage



- Grundwasserströmung gekoppelt mit Wärmetransport.
- Welche Leistung erzielt so eine Anlage?

Modell für eine Geothermieanlage

- *Strömung des Wassers* in und um das Bohrloch

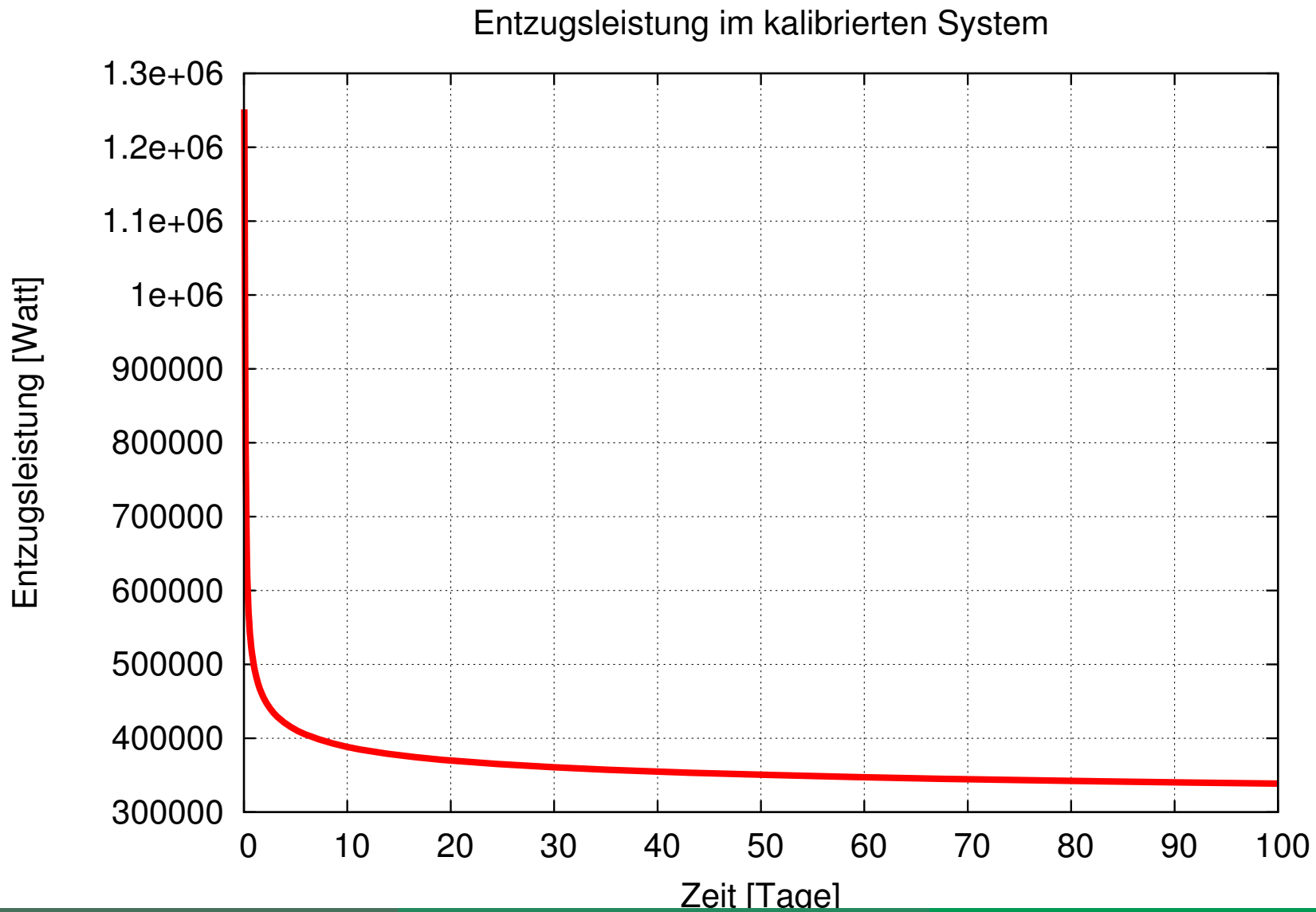
$$\nabla \cdot u = f,$$
$$u = -\frac{K}{\mu}(\nabla p - \varrho_w G)$$

- Transport der Wärme durch *Konvektion* und *Wärmeleitung*

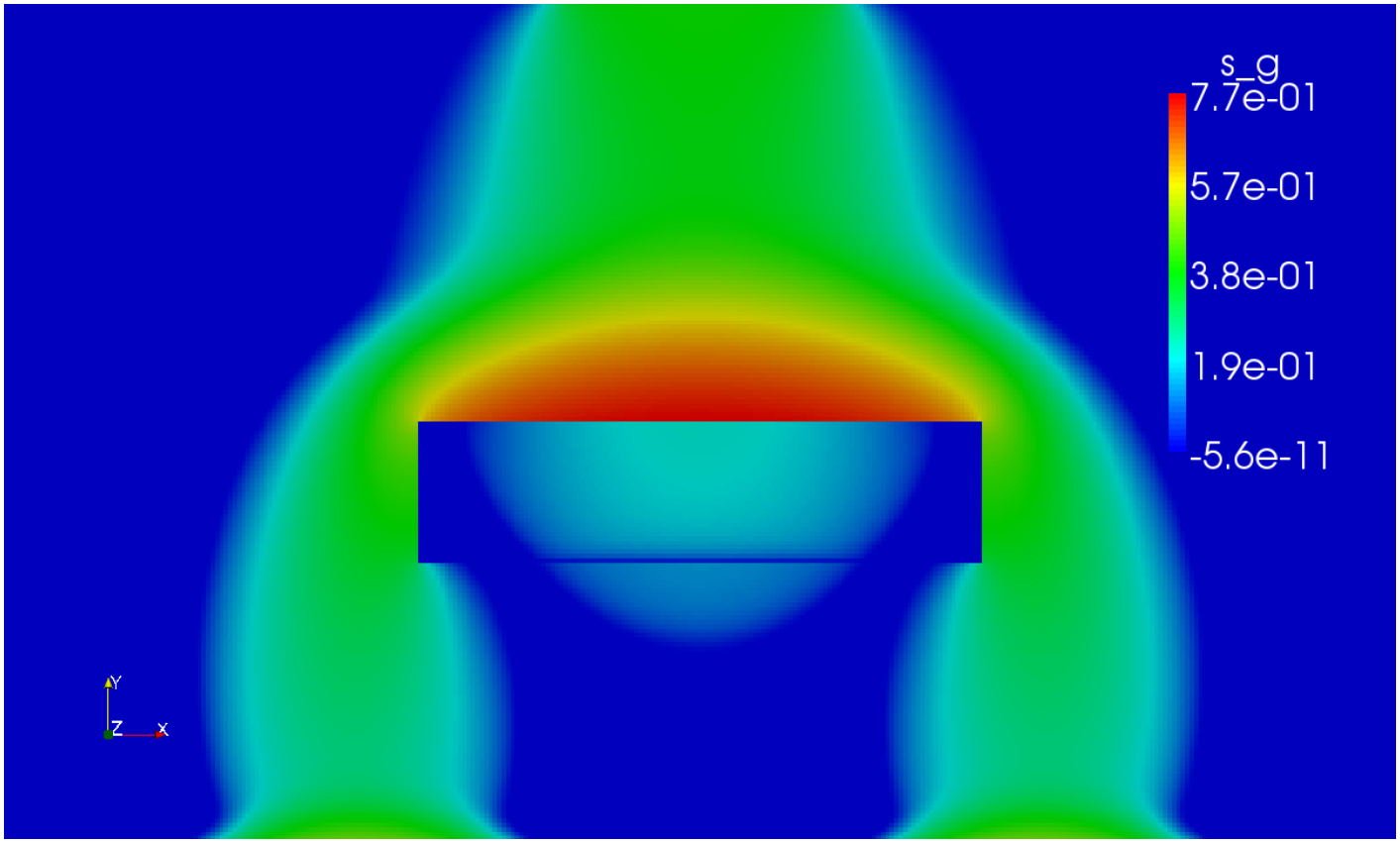
$$\frac{\partial(c_e \varrho_e T)}{\partial t} + \nabla \cdot q + c_w \varrho_w f\, T = g,$$
$$q = c_w \varrho_w u T - \lambda \nabla T$$

in Abhängigkeit diverser *Parameter*: Bodendurchlässigkeit, Wärmekapazität, Dichte, Wärmeleitfähigkeit, Pumprate sowie Rand- und Anfangsbedingungen.

Entzugsleistung



Schadstoffausbreitung



- Wo erreicht der Schadstoff welche Konzentrationen?
- Wie bekommt man den Schadstoff wieder weg?
- Wohin bewegt sich gelöster Schadstoff?

Inhalt der Vorlesung

Wir werden in dieser Vorlesung die folgenden Themengebiete behandeln:

- Grundbegriffe, Gleitpunktzahlen, Gleitpunktarithmetik
- Direkte Methoden zur Lösung linearer Gleichungssysteme
- Interpolation und Approximation
- Numerische Integration
- Iterationsverfahren zur Lösung linearer Gleichungssysteme
- Iterationsverfahren zur Lösung nichtlinearer Gleichungssysteme
- Eigenwerte und Eigenvektoren

Abschnitt 3

❸ **Fließkommazahlen**

- Zahlendarstellung
- Runden und Rundungsfehler
- Fließkommaarithmetik
- Fehleranalyse
- Die quadratische Gleichung
- Auslöschung
- Die Exponentialfunktion
- Rekursionsformel für Integrale

Stellenwertsystem

Definition 3.1 (Stellenwertsystem / Positionssystem)

System, das Zahlen $x \in \mathbb{R}$ wie folgt darstellt:

$$\begin{aligned} x &= \pm \ldots m_2 \beta^2 + m_1 \beta + m_0 + m_{-1} \beta^{-1} + m_{-2} \beta^{-2} \ldots \\ &= \sum_{i \in \mathbb{Z}} m_i \beta^i \end{aligned}$$

$\beta \in \mathbb{N}, \beta \geq 2$ heißt *Basis*, $m_i \in \{0, 1, 2, \ldots, \beta - 1\}$ heißen *Ziffern*

Geschichte: [Knuth, Band 2, S. 194]

- Babylonier (≈ -1750), $\beta = 60$
- Basis 10 ab ca. 1580
- Pascal: alle $\beta \geq 2$ möglich

Festkommazahlen

Festkommazahlen: brich Reihe nach endlich vielen Einträgen ab

$$x = \pm \sum_{i=-k}^{n} m_i \beta^i$$

Problem: wissenschaftl. Anwendungen brauchen Zahlen sehr unterschiedlicher Größe

Plancksches Wirkungsquantum:	$6.626093 \cdot 10^{-34}$ Js
Avogadrokonstante:	$6.021415 \cdot 10^{23}$ $\frac{1}{\text{mol}}$
Ruhemasse Elektron:	$9.109384 \cdot 10^{-31}$ kg
Lichtgeschwindigkeit:	$2.997925 \cdot 10^{8}$ $\frac{\text{m}}{\text{s}}$

Fließkommazahlen können unterschiedlich große Zahlen hinreichend genau darstellen

Fließkommazahlen

Definition 3.2 (Fließkommazahlen)

Sei $\beta, r, s \in \mathbb{N}$ und $\beta \geq 2$. Das *Fließkommagitter* $\mathbb{F}(\beta, r, s) \subset \mathbb{R}$ besteht aus den Zahlen mit folgenden Eigenschaften:

1. $\forall x \in \mathbb{F}(\beta, r, s)\colon x = m(x) \cdot \beta^{e(x)}$ mit

$$m(x) = \pm \sum_{i=1}^{r} m_i \beta^{-i}, \quad e(x) = \pm \sum_{j=0}^{s-1} e_j \beta^{j}$$

   mit Ziffern m_i und e_j. m heißt *Mantisse*, e heißt *Exponent*.
2. $\forall x \in \mathbb{F}(\beta, r, s)\colon x = 0 \vee m_1 \neq 0$. $m_1 \neq 0$ heißt *Normierung* und macht die Darstellung eindeutig.

Eigenschaften

Ist $x \in \mathbb{F}(\beta, r, s)$ und $x \neq 0$, dann gilt wegen Punkt 2:

$$\beta^{-1} \leq |m(x)| < 1 \quad \text{und damit} \quad \beta^{e(x)-1} \leq |x| < \beta^{e(x)} \tag{3.1}$$

Größte / kleinste darstellbare Zahl in $\mathbb{F}(\beta, r, s)$:

$$X_{+/-} = \pm(1 - \beta^{-r}) \cdot \beta^{\beta^s - 1}$$

Kleinste positive / größte negative Zahl:

$$x_{+/-} = \pm\beta^{-\beta^s}$$

$\rightsquigarrow$ Beweis: Tafel

Beispiel

Beispiel 3.3

1. $\mathbb{F}(10, 3, 1)$ besteht aus Zahlen der Form:

$$x = \pm(m_1 \cdot 0.1 + m_2 \cdot 0.01 + m_3 \cdot 0.001) \cdot 10^{\pm e_0}$$

mit $m_1 \neq 0 \vee (m_1 = m_2 = m_3 = 0)$, z.B. 0, $0.999 \cdot 10^4$, und $0.123 \cdot 10^{-1}$, aber *nicht* $0.140 \cdot 10^{-10}$, da Exponent zu klein

2. $\mathbb{F}(2, 2, 1)$ besteht aus Zahlen der Form:

$$x = \pm(m_1 \cdot 0.5 + m_2 \cdot 0.25) \cdot 2^{\pm e_0}$$

$$\Longrightarrow \mathbb{F}(2, 2, 1) = \left\{ -\frac{3}{2}, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{3}{8}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2} \right\}$$

$\rightsquigarrow$ Rechnung: Tafel

Abstände zwischen Fließkommazahlen

Sei $x' \in \mathbb{F}$ die nächstliegende Fließkommazahl zu $x \in \mathbb{F}, x > 0$.
Dann gilt:

1. Für $m(x) \neq \beta^{-1}$: $|x - x'| = \frac{|x|}{|m(x)|}\beta^{-r}$
2. Für $m(x) = \beta^{-1}$: $|x - x'| = |x|\beta^{-r}$

Für den *relativen* Abstand gilt:

$$\beta^{-r} \leq \frac{|x - x'|}{|x|} \leq \beta^{1-r}$$

Schwankt um Faktor β je nach Größe von $|x|$. Nennt man *wobble*.
Daher: kleine β besser!

$\rightsquigarrow$ Beweis: Tafel

Standard: IEEE 754 / IEC 559

Ziel: Portabilität von Programmen mit Fließkommaarithmetik
Verabschiedet 1985

$\beta = 2$, mit vier Genauigkeitsstufen und normierter Darstellung:

	single	single-ext	double	double-ext
$e_{\max}$	127	≥ 1024	1023	≥ 16384
$e_{\min}$	-126	$\leq$ -1021	-1022	$\leq$ -16381
Bits Expon.	8	≤ 11	11	≥ 15
Bits total	32	≥ 43	64	≥ 79

Der Standard definiert vier Rundungsarten:
nach $-\infty$, nach $+\infty$, nach 0, nearest

Seit 2008: zusätzlich half precision und quadruple precision

Double Precision

Betrachte double precision genauer:

- Insgesamt 64 bit
- 11 bit für Exponent, der vorzeichenlos als Zahl $c \in [1, 2046]$ gespeichert wird
- Setze $e := c - 1023 \implies e \in [-1022, 1023]$, kein Vorzeichen nötig
- Die Werte $c \in \{0, 2047\}$ werden anderweitig genutzt:
  - $c = 0 \wedge m = 0$ kodiert die Null
  - $c = 0 \wedge m \neq 0$ kodiert denormalisierte Darstellung
  - $c = 2047 \wedge m = 0$ kodiert ∞ (Überlauf)
  - $c = 2047 \wedge m \neq 0$ kodiert NaN = "not a number", z.B. bei Division durch Null

Double Precision

- $64 - 11 = 53$ bit für Mantisse, eines für Vorzeichen, bleiben 52 Nachkommastellen
- Wegen $\beta = 2$ gilt immer $m_1 = 1$
- Diese Ziffer heißt *hidden bit* und wird nicht gespeichert
- Somit $r = 53$ im Sinne der Definition von Fließkommazahlen

Bis auf die Spezialcodes entspricht double precision somit $\mathbb{F}(2, 53, 10)$.

Rundungsfunktion

Um $x \in \mathbb{R}$ in $\mathbb{F}(\beta, r, s)$ zu approximieren, brauchen wir

$$\mathrm{rd}\colon D(\beta, r, s) \to \mathbb{F}(\beta, r, s), \tag{3.2}$$

wobei $D(\beta, r, s) \subset \mathbb{R}$ der Bereich ist, der $\mathbb{F}(\beta, r, s)$ enthält:

$$D := [X_-, x_-] \cup \{0\} \cup [x_+, X_+]$$

Achtung: das setzt voraus, dass x im darstellbaren Bereich liegt! Bei Über-/Unterlauf sind r,s zu ändern.

Sinnvollerweise soll für rd gelten:

$$\forall x \in D\colon\ |x - \mathrm{rd}(x)| = \min_{y \in \mathbb{F}} |x - y| \quad \text{(Bestapproximation)}$$

Rundungsfunktion

Mit $l(x) := \max\{y \in \mathbb{F} | y \leq x\}$ und $r(x) := \min\{y \in \mathbb{F} | y \geq x\}$ gilt dann:

$$\mathrm{rd}(x) = \begin{cases} x & l(x) = r(x), x \in \mathbb{F} \\ l(x) & |x - l(x)| < |x - r(x)| \\ r(x) & |x - l(x)| > |x - r(x)| \\ ? & |x - l(x)| = |x - r(x)| \end{cases}$$

Im letzten Fall ist eine Rundung nötig. Dafür gibt es verschiedene Möglichkeiten.

Natürliche Rundung

Definition 3.4 (Natürliche Rundung)

Sei $x = \text{sign}(x) \cdot \left(\sum_{i=1}^{\infty} m_i \beta^{-i}\right) \beta^e$ die *normierte* Darstellung von $x \in D$. Setze

$$\text{rd}(x) := \begin{cases} l(x) = \text{sign}(x) \cdot \left(\sum_{i=1}^{r} m_i \beta^{-i}\right) \beta^e & \text{falls } 0 \leq m_{r+1} < \beta/2 \\ r(x) = l(x) + \beta^{e-r} \text{ (letzte Stelle)} & \text{falls } \beta/2 \leq m_{r+1} < \beta \end{cases}$$

Das ist die übliche Rundung, die jeder kennt.

Gerade Rundung

Definition 3.5 (Gerade Rundung)

Setze (mit Notation wie bei natürlicher Rundung)

$$\mathrm{rd}(x) := \begin{cases} l(x) & \text{falls } |x - l(x)| < |x - r(x)| \\ l(x) & \text{falls } |x - l(x)| = |x - r(x)| \wedge m_r \text{ gerade} \\ r(x) & \text{sonst} \end{cases}$$

Damit ist m_r in $\mathrm{rd}(x)$ immer gerade, wenn gerundet wurde.

- Für $\mathrm{rd}(x) = l(x)$ ist das nach Definition so.
- Sonst $\mathrm{rd}(x) = r(x) = l(x) + \beta^{e-r}$, m_r in $l(x)$ ungerade, und Addition von β^{e-r} ändert die letzte Stelle um Eins.

Diese Wahl von Rundung vermeidet eine mögliche Drift beim Aufrunden. Entspricht "round to nearest" im Standard.

Beispiel

Beispiel 3.6

Stelle die Zahl $x = (110)_{10}$ in $\mathbb{F}(4, 3, 2)$ dar, d.h. konvertiere ins Vierersystem ($\beta = 4$) und runde.

- $\log_4(110) = 3.39 \rightsquigarrow$ Exponent 4, damit Mantisse < 1
- Berechne $r + 1 = 4$ Ziffern, um runden zu können
  $\rightsquigarrow m_1 = 1, m_2 = 2, m_3 = 3, m_4 = 2$
- $\Longrightarrow (110)_{10} = 0.1232 \cdot (10)_4^{10}$ (exakt, Ziffern zur Basis 4)
- Natürliche Rundung: $(x')_4 = 0.130 \cdot 4^{10}$, da $m_{r+1} = 2 = 4/2$
- Gerade Rundung: $(x')_4 = 0.130 \cdot 4^{10}$, da $m_r = 3$ ungerade

Wichtig: Ergebnisse sind nur zufällig identisch!

$\rightsquigarrow$ Rechnung: Tafel

Absoluter und relativer Fehler

Definition 3.7 (absoluter und relativer Fehler)

Sei $x' \in \mathbb{R}$ eine Näherung von $x \in \mathbb{R}$. Dann heißt

$$\Delta x := x' - x \quad \text{absoluter Fehler}$$

und für $x \neq 0$

$$\epsilon_{x'} := \frac{\Delta x}{x} \quad \text{relativer Fehler}$$

Umformen liefert:

$$x' = x + \Delta x = x \cdot \left(1 + \frac{\Delta x}{x}\right) = x \cdot (1 + \epsilon_{x'})$$

Motivation

Motivation:
Es sei $\Delta x = x' - x = 100\,\text{km}$.
Für $x =$ Entfernung Erde—Sonne $\approx 1.5 \cdot 10^8\,\text{km}$ ist

$$\epsilon_{x'} = \frac{10^2\,\text{km}}{1.5 \cdot 10^8\,\text{km}} \approx 6.6 \cdot 10^{-7}$$

relativ klein.
Für $x =$ Entfernung Heidelberg—Paris $\approx 460\,\text{km}$ ist

$$\epsilon_{x'} = \frac{10^2\,\text{km}}{4.6 \cdot 10^2\,\text{km}} \approx 0.22 \quad (22\%)$$

dagegen relativ groß.

Fehlerabschätzung

Lemma 3.8 (Rundungsfehler)

Bei der Rundung in $\mathbb{F}(\beta, r, 2)$ gilt für den absoluten Fehler

$$|x - \mathrm{rd}(x)| \leq \frac{1}{2}\beta^{e(x)-r} \tag{3.3}$$

und für den relativen Fehler ($x \neq 0$)

$$\frac{|x - \mathrm{rd}(x)|}{|x|} \leq \frac{1}{2}\beta^{1-r}.$$

Diese Abschätzung ist scharf (d.h. der Fall "=" existiert).
Die Zahl eps $:= \frac{1}{2}\beta^{1-r}$ heißt *Maschinengenauigkeit*.

$\rightsquigarrow$ Beweis: Tafel

Fließkommaarithmetik

Wir benötigen eine Arithmetik auf $\mathbb{F}$:

$$\circledast : \mathbb{F} \times \mathbb{F} \to \mathbb{F} \quad \text{mit} \quad \circledast \in \{\oplus, \ominus, \odot, \oslash\}$$

die den bekannten Operationen $* \in \{+, -, \cdot, /\}$ auf $\mathbb{R}$ entsprechen.

Problem: in der Regel $x, y \in \mathbb{F} \not\Rightarrow x * y \in \mathbb{F}$

Daher muss das Ergebnis wieder gerundet werden. Wir *definieren*

$$\forall x, y \in \mathbb{F} : x \circledast y := \mathrm{rd}(x * y) \tag{3.4}$$

Man sagt: $\circledast$ ist "exakt gerundet". Das ist nicht trivial!

Guard digit

Beispiel 3.9 (Guard digit)

Sei $\mathbb{F} = \mathbb{F}(10, 3, 1), x = 0.215 \cdot 10^8, y = 0.125 \cdot 10^{-5}$. Wir betrachten die Subtraktion $x \ominus y = \mathrm{rd}(x - y)$.

1. Subtraktion und anschließendes Runden benötigt extrem hohe Stellenzahl $\mathcal{O}(\beta^s)$!
2. Runden vor der Subtraktion liefert das selbe Ergebnis. Gute Idee?
3. Aber: betrachte z.B. $x = 0.101 \cdot 10^1, y = 0.993 \cdot 10^0$
   $\Longrightarrow$ relativer Fehler 18% $\approx$ 35 eps
4. Mit ein, zwei zusätzlichen Stellen erreicht man exakte Rundung!
5. Diese Stellen heißen *guard digits* und werden auch in der Praxis (CPU) genutzt.

$\rightsquigarrow$ Rechnung: Tafel

Tabellenmacher-Dilemma

Algebraische Funktionen:
z.B. Polynome, $1/x$, $\sqrt{x}$, rationale Funktionen,...
grob: endliche Kombination der Grundrechenarten und Wurzeln

Tranzendente Funktionen: der Rest, z.B. $\exp(x), \ln(x), \sin(x), x^y, \ldots$

Tabellenmacher-Dilemma:
Es lässt sich nicht im Voraus entscheiden, wie viele guard digits für eine bestimmte Kombination von tranzendenter Funktion f und Argument x benötigt werden, um $f(x)$ exakt gerundet zu berechnen.

IEEE754 garantiert exakte Rundung für $\oplus, \ominus, \odot, \oslash$, ebenso $\sqrt{x}$.

Weitere Probleme / Eigenschaften

Die folgenden Punkte sind zu beachten:

- Assoziativ- und Distributivgesetz gelten nicht, es kommt auf die Reihenfolge von Operationen an!
- Es gibt $y \in \mathbb{F}, y \neq 0$, so dass $x \oplus y = x$
- Beispiel: $(\epsilon \oplus 1) \ominus 1 = 1 \ominus 1 = 0 \neq \epsilon = \epsilon \oplus 0 = \epsilon \oplus (1 \ominus 1)$
- Das Kommutativgesetz gilt jedoch:
  $x \circledast y = y \circledast x$ für $\circledast \in \{\oplus, \odot\}$
- Weitere einfache gültige Regeln:
  - $(-x) \odot y = -(x \odot y)$
  - $1 \odot x = x \oplus 0 = x$
  - $x \odot y = 0 \implies x = 0 \vee y = 0$
  - $x \odot z \leq y \odot z \quad \text{falls} \quad x \leq y \wedge z > 0$

Fehleranalyse

Rundungsfehler pflanzen sich in Rechnungen fort.

- Sei $F\colon \mathbb{R}^m \to \mathbb{R}^n$, in Komponenten $F(x) = \begin{pmatrix} F_1(x_1, \ldots, x_m) \\ \vdots \\ F_n(x_1, \ldots, x_m) \end{pmatrix}$
- Zur Berechnung von F im Rechner nutze *numerische Realisierung* $F'\colon \mathbb{F}^m \to \mathbb{F}^n$.
  F' wird durch einen *Algorithmus* realisiert, d.h. aus
  - endlich vielen (= Terminierung)
  - elementaren (= bekannten, d.h. $\oplus, \ominus, \odot, \oslash$)

  Rechenoperationen zusammengesetzt:

$$F'(x) = \varphi_l(\ldots \varphi_2(\varphi_1(x)) \ldots)$$

Fehleranalyse

Wichtig:

1. Zu einem F gibt es in der Regel viele Realisierungen, im Sinne unterschiedlicher *Reihenfolgen*

$$a + b + c \approx (a \oplus b) \oplus c \neq a \oplus (b \oplus c)!$$

2. Jeder Schritt φ_i steuert einen (unbekannten) Fehler bei.
3. Im Prinzip kann die Rechengenauigkeit beliebig gesteigert werden, d.h. eigentlich Folge $(F')^{(k)} : (\mathbb{F}^{(k)})^m \to (\mathbb{F}^{(k)})^n$. Das machen wir aber nicht so formal.

Fehleranalyse

$$F(x) - F'(\mathrm{rd}(x)) = \underbrace{F(x) - F(\mathrm{rd}(x))}_{\text{Konditionsanalyse}} + \underbrace{F(\mathrm{rd}(x)) - F'(\mathrm{rd}(x))}_{\text{Rundungsfehleranalyse}} \quad (3.5)$$

- $F(x)$: exaktes Ergebnis
- $F'(\mathrm{rd}(x))$: numerische Auswertung
- $F(\mathrm{rd}(x))$: exaktes Ergebnis für $\mathrm{rd}(x) \approx x$

Von hier aus:

- Analyse "in erster Näherung"
- absolute / relative Fehler
- Normen bilden, das lassen wir aber i.d.R. weg

Differentielle Konditionsanalyse

Wir nehmen an, dass $F\colon \mathbb{R}^m \to \mathbb{R}^n$ zweimal stetig differenzierbar.
Nach dem Satz von Taylor gilt für die Komponenten F_i:

$$F_i(x+\Delta x) = F_i(x) + \sum_{j=1}^{m} \frac{\partial F_i}{\partial x_j}(x)\Delta x_j + R_i^F(x;\Delta x) \quad i = 1,\ldots,n.$$

Für das Restglied gilt unter diesen Voraussetzungen

$$R_i^F(x;\Delta x) = \mathcal{O}\left(\|\Delta x\|^2\right),$$

d.h. der Fehler der Näherung ist quadratisch in Δx.

Landau-Symbole

Definition 3.10 (Landausche Symbole)

Man schreibt:

$$g(t) = \mathcal{O}(h(t)), \quad (t \to 0)$$

falls es $t_0 > 0$ und $c \geq 0$ gibt, so dass für alle $t \in (0, t_0]$ die Abschätzung $|g(t)| \leq c \cdot |h(t)|$ gilt.

Sprechweise: "$g(t)$ geht wie $h(t)$ gegen 0"
(quantifiziert also, "wie schnell" eine Funktion gegen Null geht)

Weiter bedeutet

$$g(t) = o(h(t)), \quad (t \to 0)$$

dass es $t_0 > 0$ und eine *Funktion* $c(t), \lim_{t\to 0} c(t) = 0$ gibt, so dass für alle $t \in (0, t_0]$ die Abschätzung $|g(t)| \leq c(t) \cdot |h(t)|$ gilt.

Bedeutung: "$g(t)$ geht schneller als $h(t)$ gegen 0" (falls $h(t) \to 0$)

Differentielle Konditionsanalyse

Somit können wir die Taylorformel umformen:

$$F_i(x + \Delta x) - F_i(x) = \underbrace{\sum_{j=1}^{m} \frac{\partial F_i}{\partial x_j}(x)\Delta x_j}_{\text{führende (erste) Ordnung}} + \underbrace{R_i^F(x; \Delta x)}_{\text{höhere Ordnungen}}$$

Oft lässt man die Terme höherer Ordnung weg und schreibt dann "$\doteq$" statt "$=$".

Sprechweise: "ist in erster Näherung gleich"

Differentielle Konditionsanalyse

Damit gilt:

$$\frac{F_i(x+\Delta x)-F_i(x)}{F_i(x)} \doteq \sum_{j=1}^{m} \frac{\partial F_i}{\partial x_j}(x)\frac{\Delta x_j}{F_i(x)} \tag{3.6}$$

$$\doteq \sum_{j=1}^{m} \underbrace{\left(\frac{\partial F_i}{\partial x_j}(x)\frac{x_j}{F_i(x)}\right)}_{\text{Verstärkungsfaktor } k_{ij}(x)} \cdot \underbrace{\left(\frac{\Delta x_j}{x_j}\right)}_{\leq \text{eps}},$$

d.h. die Verstärkungsfaktoren $k_{ij}(x)$ geben an, wie die (relativen) Eingabefehler $\frac{\Delta x_j}{x_j}$ in den (relativen) Fehler der i-ten Komponente von F eingehen!

Kondition

Definition 3.11 (Kondition)

Wir nennen die Auswertung $y = F(x)$ "schlecht konditioniert" im Punkt x, falls $|k_{ij}(x)| \gg 1$, andernfalls "gut konditioniert". $|k_{ij}(x)| < 1$ heißt Fehlerdämpfung, $|k_{ij}(x)| > 1$ heißt Fehlerverstärkung.

Das Symbol "$\gg$" steht für "viel größer als". Meist bedeutet das, dass die eine Zahl mehrere Größenordnungen größer als die andere ist (z.B. 1 Mio. $\gg$ 1).

Definition stellt ein Kontinuum dar: es gibt einen fließenden Übergang von "gut konditioniert" zu "schlecht konditioniert"!

Kondition

Warum *relative* Kondition?

Wegen Lemma 3.8 (Rundungsfehler) gilt

$$\left| \frac{x - \mathrm{rd}(x)}{x} \right| \leq \mathrm{eps} = \frac{1}{2}\beta^{1-r},$$

d.h. es gibt $\epsilon \in \mathbb{R}, |\epsilon| \leq \mathrm{eps}$, so dass

$$\frac{x - \mathrm{rd}(x)}{x} = \epsilon,$$

d.h. für die relativen Eingabefehler in Gl. (3.6) gilt gerade

$$\frac{\Delta x_j}{x_j} = \epsilon_j$$

Beispiel I

Beispiel 3.12

1. Addition: $F(x_1, x_2) = x_1 + x_2$, $\frac{\partial F}{\partial x_1} = \frac{\partial F}{\partial x_2} = 1$.
   Nach obiger Formel:

$$\frac{F(x_1 + \Delta x_1, x_2 + \Delta x_2) - F(x_1, x_2)}{F(x_1, x_2)}$$

$$\doteq \underbrace{1 \cdot \frac{x_1}{x_1 + x_2}}_{=k_1} \frac{\Delta x_1}{x_1} + \underbrace{1 \cdot \frac{x_2}{x_1 + x_2}}_{=k_2} \frac{\Delta x_2}{x_2}$$

   Schlecht konditioniert für $x_1 \to -x_2$!

Beispiel II

Beispiel 3.12

❷ $F(x_1, x_2) = x_1^2 - x_2^2$, $\frac{\partial F}{\partial x_1} = 2x_1$, $\frac{\partial F}{\partial x_2} = -2x_2$.

$$\frac{F(x_1 + \Delta x_1, x_2 + \Delta x_2) - F(x_1, x_2)}{F(x_1, x_2)}$$

$$\doteq \underbrace{2x_1 \cdot \frac{x_1}{x_1^2 - x_2^2}}_{=k_1} \frac{\Delta x_1}{x_1} + \underbrace{(-2x_2) \cdot \frac{x_2}{x_1^2 - x_2^2}}_{=k_2} \frac{\Delta x_2}{x_2}$$

$$\Longrightarrow \quad k_1 = \frac{2x_1^2}{x_1^2 - x_2^2}, k_2 = -\frac{2x_2^2}{x_1^2 - x_2^2},$$

schlecht konditioniert für $|x_1| \approx |x_2|$.

Rundungsfehleranalyse

Sogenannte "Vorwärtsrundungsfehleranalyse", rückwärts: später.

Nach Fehler-Aufspaltung, Gl. (3.5): $F(x) - F'(x)$ mit $x \in \mathbb{F}^m$, F' "zusammengesetzt" aus Einzeloperationen $\circledast \in \{\oplus, \ominus, \odot, \oslash\}$

Wegen Gl. (3.4) (exakt gerundete Arithmetik) und Lemma 3.8 (Rundungsfehler) gilt

$$\frac{(x \circledast y) - (x * y)}{(x * y)} = \epsilon \quad \text{mit } |\epsilon| \leq \text{eps}$$

Vorsicht, ϵ hängt von x und y ab und ist daher für jede Operation verschieden!

$$\Longrightarrow \quad x \circledast y = (x * y) \cdot (1 + \epsilon) \quad \text{für ein } |\epsilon(x, y)| \leq \text{eps}$$

Beispiel I

Beispiel 3.13

$F(x_1, x_2) = x_1^2 - x_2^2$ mit zwei Realisierungen:

1. $F_a(x_1, x_2) = (x_1 \odot x_1) \ominus (x_2 \odot x_2)$
2. $F_b(x_1, x_2) = (x_1 \ominus x_2) \odot (x_1 \oplus x_2)$

Erste Realisierung:

$$u = x_1 \odot x_1 = (x_1 \cdot x_1) \cdot (1 + \epsilon_1)$$
$$v = x_2 \odot x_2 = (x_2 \cdot x_2) \cdot (1 + \epsilon_2)$$
$$F_a(x_1, x_2) = u \ominus v = (u - v) \cdot (1 + \epsilon_3)$$
$$\frac{F_a(x_1, x_2) - F(x_1, x_2)}{F(x_1, x_2)} \doteq \frac{x_1^2}{x_1^2 - x_2^2}(\epsilon_1 + \epsilon_3) + \frac{x_2^2}{x_2^2 - x_1^2}(\epsilon_2 + \epsilon_3)$$

Beispiel II

Beispiel 3.13

Zweite Realisierung:

$$u = x_1 \ominus x_2 = (x_1 - x_2) \cdot (1 + \epsilon_1)$$
$$v = x_1 \oplus x_2 = (x_1 + x_2) \cdot (1 + \epsilon_2)$$
$$F_b(x_1, x_2) = u \odot v = (u \cdot v) \cdot (1 + \epsilon_3)$$
$$\frac{F_b(x_1, x_2) - F(x_1, x_2)}{F(x_1, x_2)} \doteq \frac{x_1^2 - x_2^2}{x_1^2 - x_2^2}(\epsilon_1 + \epsilon_2 + \epsilon_3) = \epsilon_1 + \epsilon_2 + \epsilon_3$$

$\Longrightarrow$ Zweite Realisierung besser als erste.

$\rightsquigarrow$ Rechnung: Tafel

Numerische Stabilität

Definition 3.14 (Numerische Stabilität)

Wir nennen einen numerischen Algorithmus "numerisch stabil", wenn die im Lauf der Rechnung akkumulierten Rundungsfehler den unvermeidbaren Problemfehler aus der Konditionsanalyse nicht übersteigen.

Mit anderen Worten:
Verstärkungsfaktoren aus Rundungsfehleranalyse $\leq$ denen aus Konditionsanalyse $\implies$ "numerisch stabil"

Beide Realisierungen a, b aus Bsp. 3.13 sind numerisch stabil.

Die quadratische Gleichung

Die Gleichung

$$y^2 - py + q = 0$$

hat für $p^2/4 > q \neq 0$ die beiden reellen und verschiedenen Lösungen

$$y_{1,2} = f_{\pm}(p, q) = \frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}. \quad \text{(Definiert zwei } f\text{!)}$$

Die Konditionsanalyse liefert mit $D := \sqrt{\frac{p^2}{4} - q}$:

$$\frac{f(p + \Delta p, q + \Delta q) - f(p, q)}{f(p, q)}$$

$$\doteq \left(1 \pm \frac{p}{2D}\right) \frac{p}{p \pm 2D} \frac{\Delta p}{p} - \frac{q}{D\,(p \pm 2D)} \frac{\Delta q}{q}$$

Die quadratische Gleichung

Das heißt:

- Für $\frac{p^2}{4} \gg q$ und $p < 0$ ist

$$f_-(p,q) = \frac{p}{2} - \sqrt{\frac{p^2}{4} - q}$$

  gut konditioniert
- Für $\frac{p^2}{4} \gg q$ und $p > 0$ ist

$$f_+(p,q) = \frac{p}{2} + \sqrt{\frac{p^2}{4} - q}$$

  gut konditioniert
- Für $\frac{p^2}{4} \approx q$ sind f_+ und f_- beide schlecht konditioniert, dieses Problem kann nicht vermieden werden

Die quadratische Gleichung

Numerisch geschickte Auswertung für den Fall $\frac{p^2}{4} \gg q$:

$p < 0$:

Berechne $y_2 = \frac{p}{2} - \sqrt{\frac{p^2}{4} - q}$, $y_1 = \frac{q}{y_2}$ nach Satz von Vieta $(q = y_1 \cdot y_2)$.

$p > 0$:

Berechne $y_1 = \frac{p}{2} + \sqrt{\frac{p^2}{4} - q}$, dann $y_2 = \frac{q}{y_1}$.

$\Longrightarrow$ Jedes Problem ist ein Spezialfall!

Auslöschung

Die obigen Beispiele enthalten das Phänomen der *Auslöschung*.
Dies tritt auf bei

- Addition $x_1 + x_2$ mit $x_1 \approx -x_2$
- Subtraktion $x_1 - x_2$ mit $x_1 \approx x_2$

Bemerkung 3.15

Bei der Auslöschung werden *vor* der entsprechenden Addition bzw. Subtraktion eingeführte Fehler extrem verstärkt.

Sind $x_1, x_2 \in \mathbb{F}$ *Maschinenzahlen*, so gilt wie oben gezeigt:

$$\left| \frac{(x_1 \ominus x_2) - (x_1 - x_2)}{(x_1 - x_2)} \right| \leq \text{eps}.$$

Das ist also kein Problem. Das Problem tritt erst ein, wenn x_1 und x_2 selbst schon mit Fehlern behaftet sind.

Beispiel

Beispiel 3.16

Betrachte $\mathbb{F} = \mathbb{F}(10, 4, 1)$.

$x_1 = 0.11258762 \cdot 10^2, x_2 = 0.11244891 \cdot 10^2$

$\implies \text{rd}(x_1) = 0.1126 \cdot 10^2, \text{rd}(x_2) = 0.1124 \cdot 10^2$

$x_1 - x_2 = 0.13871 \cdot 10^{-1}$, aber $\text{rd}(x_1) - \text{rd}(x_2) = 0.2 \cdot 10^{-1}$

Ergebnis hat keine gültige Ziffer! Relativer Fehler:

$$\frac{0.2 \cdot 10^{-1} - 0.13871 \cdot 10^{-1}}{0.13871 \cdot 10^{-1}} \approx 0.44 \approx 883 \cdot \underbrace{\frac{1}{2} \cdot 10^{-3}}_{=\text{eps}}!$$

Grundregel

Im Beispiel: Fehler durch Rundung der Eingangsgrößen.

Herkunft der Fehler ist egal, tritt ebenso auf, wenn x_1, x_2 mit Fehlern vorhergehender Rechenschritte behaftet sind.

Regel 3.17

Setze potentiell gefährliche Operationen möglichst früh im Algorithmus ein. (Vergleiche Bsp. 3.13).

Die Exponentialreihe

Die Funktion $\exp(x) = e^x$ lässt sich für alle $x \in \mathbb{R}$ als Potenzreihe darstellen:

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Naheliegender Ansatz: breche zur Berechnung nach n Gliedern ab, $\exp(x) \approx \sum_{k=0}^{n} \frac{x^k}{k!}$.

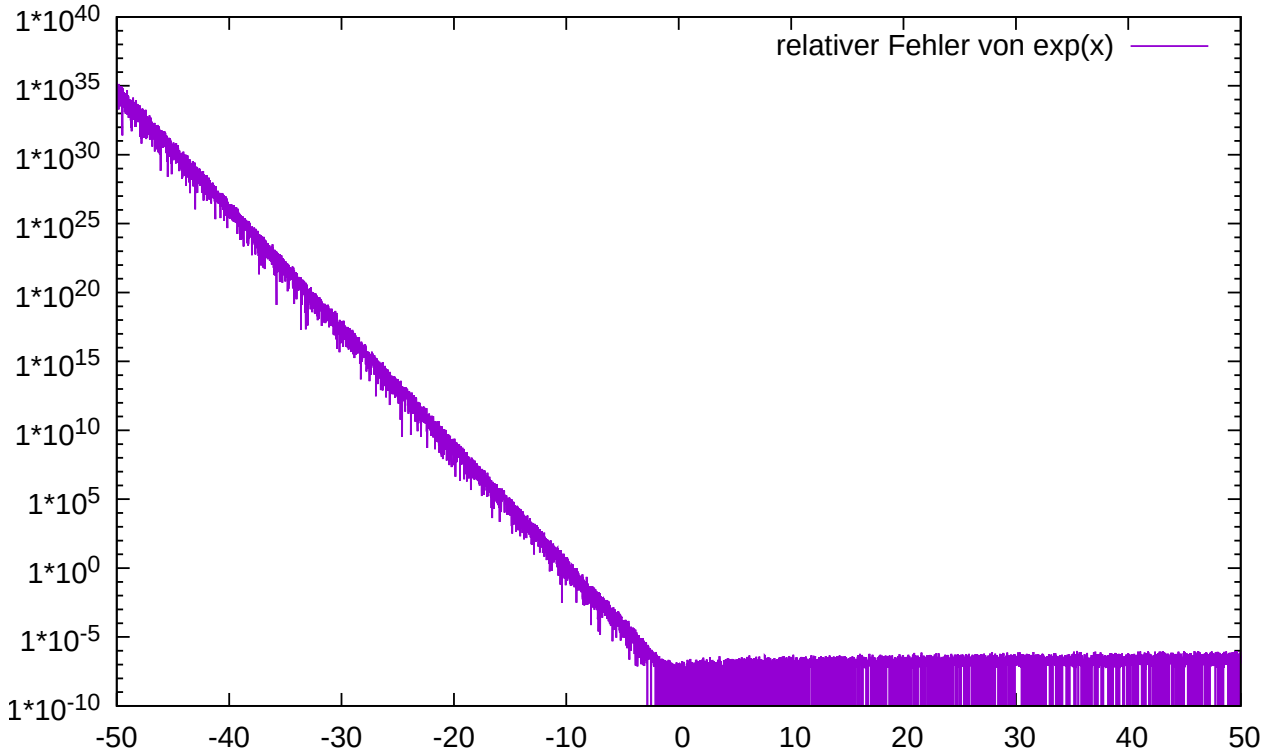
Verwende Rekursionsvorschrift:

$$y_0 := 1, \quad S_0 := y_0 = 1,$$
$$\forall k > 0: \quad y_k := \frac{x}{k} \cdot y_{k-1}, \quad S_k := S_{k-1} + y_k$$

y_n: Glieder der Reihe, S_n: Partialsummen

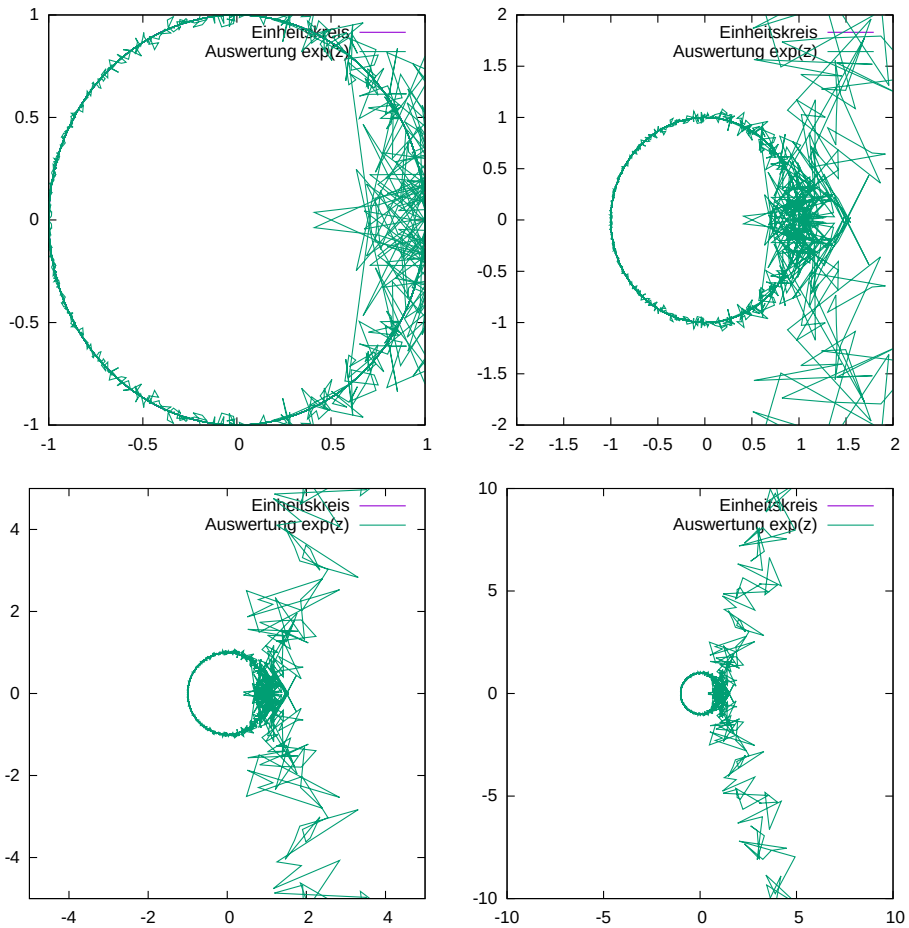
Fehler für verschiedene Werte von x

Ergebnisse für Rekursionsformel mit `float`, $n = 100$:



- Für negative Werte für x wird der Fehler beliebig groß
- Dieser Effekt hat *nichts* mit dem Abbruch der Reihe zu tun!

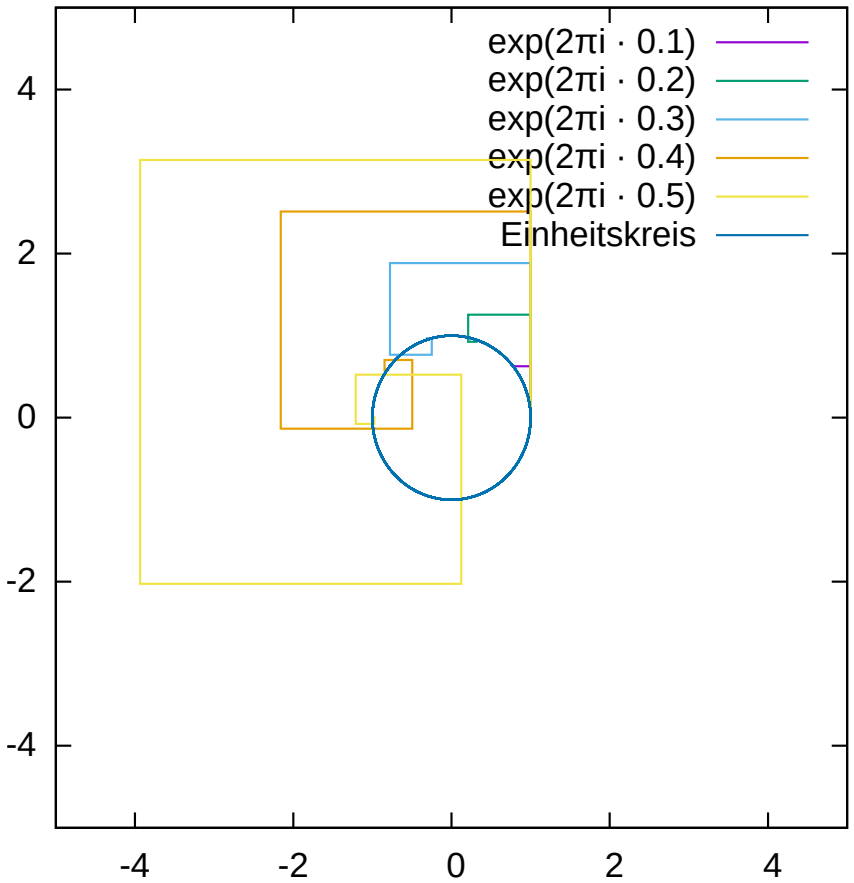
Abweichung für imaginäre Argumente



Ergebnisse für das imaginäre Intervall $[-50, 50] \cdot i$

- Für $|z| \leq \pi$ ist das Ergebnis noch halbwegs brauchbar
- Für $|z| \to 2\pi$ wächst der Fehler weiter an
- Danach entfernen sich die Werte vom Kreis (die Trajektorie nähert sich einer Geraden an und kommt auch nicht zurück)

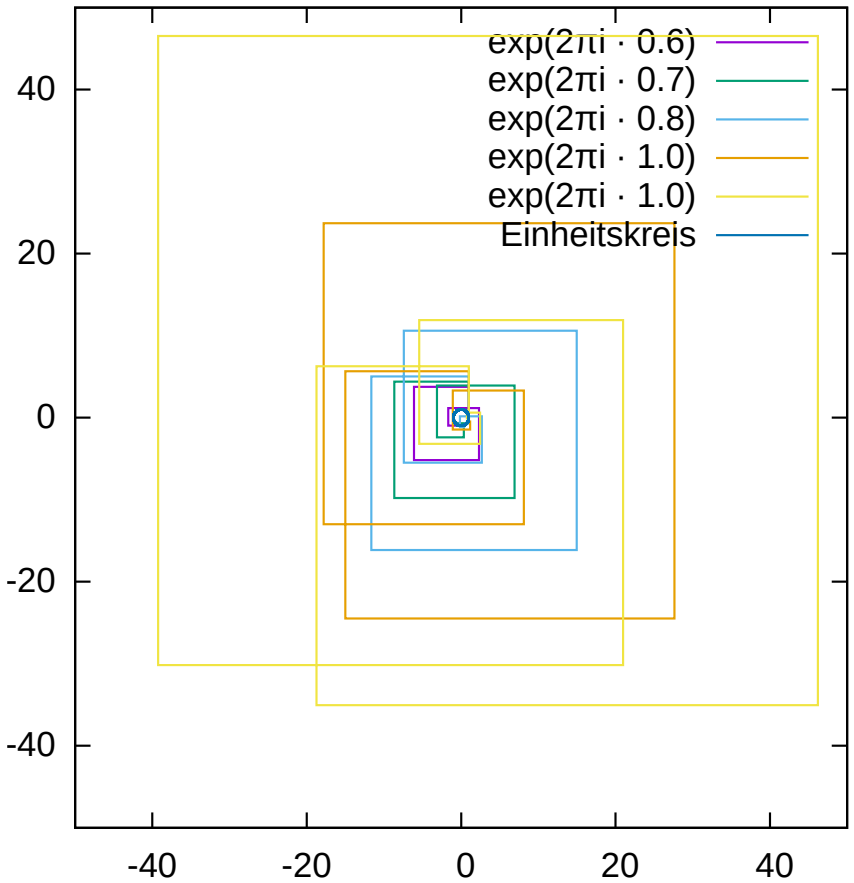
Visualisierung des Konvergenzverhaltens



- gerade Potenzen tragen zum Realteil von $\exp(2\pi i \cdot x)$ bei
- ungerade Potenzen tragen zum Imaginärteil bei

$\Longrightarrow$ Addieren eines Gliedes ändert abwechselnd Real- und Imaginärteil

Visualisierung des Konvergenzverhaltens



- Betrag der Zwischenergebnisse wächst exponentiell in x
- Form der Trajektorie nähert sich immer mehr Quadrat an

$\Longrightarrow$ Auslöschung

Konditionsanalyse für $\exp(x)$

Für die Funktion exp gilt wegen $\exp' = \exp$:

$$\frac{\exp(x + \Delta x) - \exp(x)}{\exp(x)} \doteq \left(\exp'(x)\frac{x}{\exp(x)}\right) \cdot \left(\frac{\Delta x}{x}\right) = \Delta x$$

$\Longrightarrow$ absoluter Fehler von x wird zu relativem Fehler von $\exp(x)$
(Vergleiche: exp ist Isomorphismus zwischen $(\mathbb{R}, +)$ und $(\mathbb{R}_+, \cdot)$.)

Wegen $k = x$ ist exp gut konditioniert, falls x nicht zu groß
$\Longrightarrow$ betrachteter Algorithmus ist instabil für $x < 0$

Gibt es stabileren Algorithmus? $\rightsquigarrow$ Übungsaufgabe

Rekursionsformel für Integrale

Integrale der Form

$$I_k = \int_0^1 x^k \exp(x)\, dx$$

kann man mithilfe einer Rekursionsformel berechnen:

$$I_0 = e - 1, \quad \forall k > 0 \colon I_k = e - k \cdot I_{k-1}$$

Für das erste Glied ist wegen $\exp'(x) = \exp(x)$ eine Stammfunktion bekannt, die weiteren Glieder kann man dann über die obige Formel bestimmen.

Wie gut funktioniert das in der Praxis?

Rekursionsformel für Integrale

Die ersten 26 Werte für I_k, mit endlicher Genauigkeit berechnet:

k	berechnetes I_k	Fehler $\lvert\Delta I_k\rvert$
0	1.718281828459050	$2.6 \cdot 10^{-15}$
1	1	(Null)
2	0.718281828459045	$1.5 \cdot 10^{-15}$
3	0.563436343081910	$5.5 \cdot 10^{-16}$
4	0.464536456131406	$1.0 \cdot 10^{-15}$
5	0.395599547802016	$6.0 \cdot 10^{-15}$
6	0.344684541646949	$3.8 \cdot 10^{-14}$
7	0.305490036930402	$2.7 \cdot 10^{-13}$
8	0.274361533015832	$2.1 \cdot 10^{-12}$
9	0.249028031316559	$1.9 \cdot 10^{-11}$
10	0.228001515293454	$1.9 \cdot 10^{-10}$
11	0.210265160231056	$2.1 \cdot 10^{-9}$
12	0.195099905686377	$2.5 \cdot 10^{-8}$
13	0.181983054536145	$3.3 \cdot 10^{-7}$
14	0.170519064953013	$4.6 \cdot 10^{-6}$
15	0.160495854163853	$7.0 \cdot 10^{-5}$
16	0.150348161837404	$1.1 \cdot 10^{-3}$
17	0.162363077223183	$1.9 \cdot 10^{-2}$
18	-0.204253561558257	$3.4 \cdot 10^{-1}$
19	6.59909949806592	$6.7 \cdot 10^{0}$
20	-129.263708132859	$1.3 \cdot 10^{1}$
21	2717.25615261851	$2.7 \cdot 10^{3}$
22	-59776.9170757787	$6.0 \cdot 10^{4}$
23	1374871.81102474	$1.4 \cdot 10^{6}$
24	-32996920.7463119	$3.3 \cdot 10^{7}$
25	824923021.376079	$8.2 \cdot 10^{8}$

Rekursionsformel $I_k = e - k \cdot I_{k-1}$ führt zu Fehlerverstärkung um Faktor k im k-ten Schritt!

Bessere Optionen

1. Alle I_k sind von der Form $a \cdot e + b$, mit $a, b \in \mathbb{Z}$. Berechne diese Zahlen über die Rekursionsformel, und werte erst ganz am Ende mit Fließkommazahlen aus.
2. Drehe die Rekursionsformel um: wenn $I_k \to I_{k+1}$ den Fehler um k verstärkt, wird er durch $I_{k+1} \to I_k$ um den Faktor k reduziert!

Wegen $0 \leq x^k \leq 1$ und $0 \leq \exp(x) \leq 3$ auf $[0, 1]$ muss $0 \leq I_k \leq 3$ gelten. Wenn wir also z.B. $I_{50} := 1.5$ setzen, kann der Fehler maximal 1.5 sein.

Benutze dann umgedrehte Rekursionsformel

$$I_k = (k+1)^{-1} \cdot (e - I_{k+1}).$$

Rekursionsformel für Integrale

Die Werte für I_k zwischen $k = 25$ und 50, rückwärts berechnet:

k	berechnetes I_k	Fehler $\lvert\Delta I_k\rvert$
50	1.5	$1.4 \cdot 10^{0}$
49	0.0243656365691809	$2.9 \cdot 10^{-2}$
48	0.0549778814671401	$5.9 \cdot 10^{-4}$
47	0.0554854988956647	$1.2 \cdot 10^{-5}$
46	0.0566552410545400	$2.6 \cdot 10^{-7}$
45	0.0578614475522718	$5.7 \cdot 10^{-9}$
44	0.0591204529090394	$1.3 \cdot 10^{-10}$
43	0.0604354858079547	$2.9 \cdot 10^{-12}$
42	0.0618103800616533	$6.7 \cdot 10^{-14}$
41	0.0632493201999379	$1.8 \cdot 10^{-15}$
40	0.0647568904453441	$1.4 \cdot 10^{-16}$
39	0.0663381234503425	$2.1 \cdot 10^{-16}$
38	0.0679985565386847	$1.5 \cdot 10^{-16}$
37	0.0697442966294832	$2.8 \cdot 10^{-16}$
36	0.0715820954548530	$1.9 \cdot 10^{-16}$
35	0.0735194370278942	$2.8 \cdot 10^{-17}$
34	0.0755646397551757	$2.2 \cdot 10^{-16}$
33	0.0777269761383491	$2.7 \cdot 10^{-18}$
32	0.0800168137066878	$1.8 \cdot 10^{-16}$
31	0.0824457817110112	$2.6 \cdot 10^{-16}$
30	0.0850269692499366	$2.9 \cdot 10^{-16}$
29	0.0877751619736370	$4.5 \cdot 10^{-17}$
28	0.0907071264305313	$4.3 \cdot 10^{-16}$
27	0.0938419536438755	$4.7 \cdot 10^{-16}$
26	0.0972014768450063	$1.3 \cdot 10^{-17}$
25	0.1008107827543860	$1.1 \cdot 10^{-16}$

Trotz völlig unbrauchbarer Schätzung für den Startwert I_{50} führt die neue Rekursionsformel $I_k = (k+1)^{-1} \cdot (e - I_{k+1})$ sehr schnell zu brauchbaren Ergebnissen!

Idee der Fehlerabschätzung

Nach Fehleranalyse gilt für Startwert $I_{k+m}, m \geq 1$:

$$|\Delta I_k| \approx \frac{k!}{(k+m)!}|\Delta I_{k+m}| \leq \frac{k!}{(k+m)!} \cdot 1.5 \leq (k+1)^{-m} \cdot 1.5$$

Idee: berechne benötigte Anzahl Schritte m aus gewünschter Genauigkeit $|\Delta I_k| < \text{tol}$.

$$(k+1)^{-m} \cdot 1.5 < \text{tol} \implies \exp(-m \cdot \ln(k+1)) < \frac{\text{tol}}{1.5}$$

$$\implies -m \cdot \ln(k+1) < \ln\left(\frac{\text{tol}}{1.5}\right) \implies m > \left|\frac{\ln(\text{tol}) - \ln(1.5)}{\ln(k+1)}\right|$$

Beispiel: $k = 25$, $\text{tol} = 10^{-8} \implies m > 5.7$

Idee der Fehlerabschätzung

Ergebnis für $m = 6$:

k	berechnetes I_k	Fehler $\lvert\Delta I_k\rvert$
31	1.5	$1.4 \cdot 10^{0}$
30	0.0392994138212595	$4.6 \cdot 10^{-2}$
29	0.0892994138212595	$1.5 \cdot 10^{-3}$
28	0.0906545660219926	$5.3 \cdot 10^{-5}$
27	0.0938438308013233	$1.9 \cdot 10^{-6}$
26	0.0972014073206564	$7.0 \cdot 10^{-8}$
25	0.1008107854284000	$2.9 \cdot 10^{-9}$

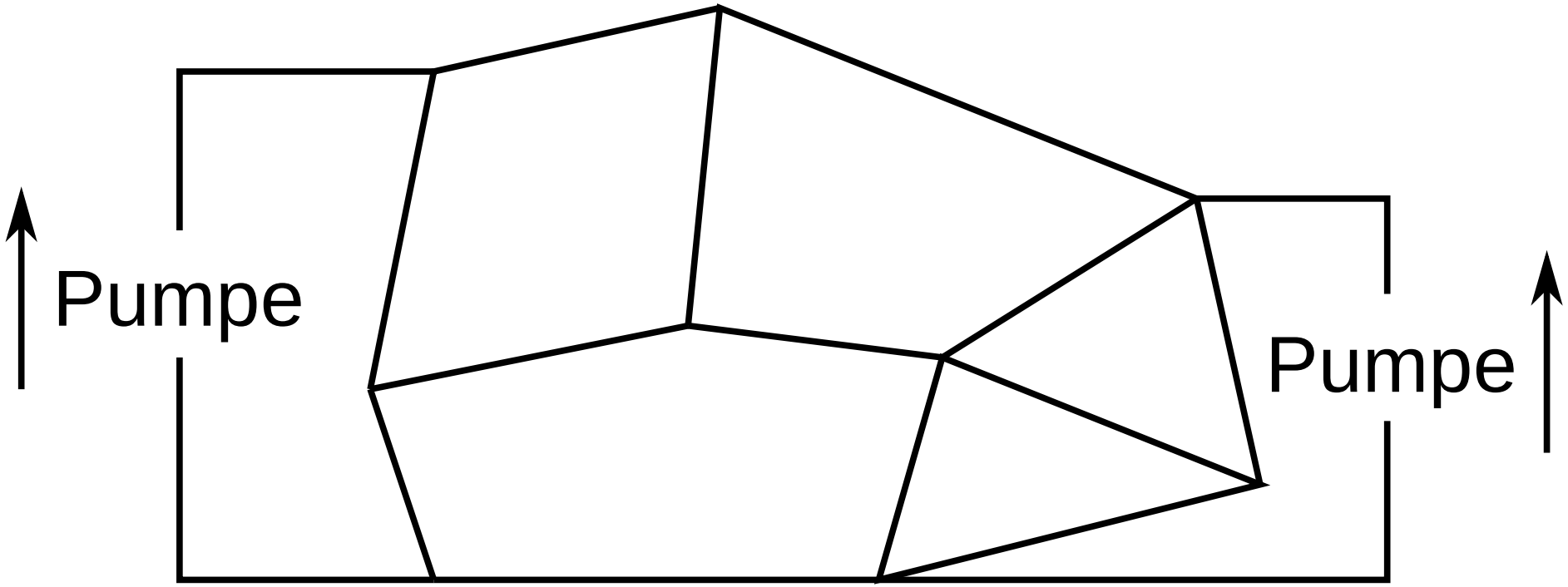
- Umgedrehte Rekursionsformel ist im Gegensatz zum naiven Ansatz stabil
- Fehlerabschätzung minimiert Aufwand für vorgegebene Genauigkeit

$\Longrightarrow$ stabil und effizient

Abschnitt 4

4. Motivation linearer Gleichungssysteme
   - Strömung in Rohrleitungsnetzen
   - Die Kirchhoffschen Gesetze
   - Das Knotenpotentialverfahren
   - Partielle Differentialgleichungen
   - Radiosity-Methode in der Computergraphik

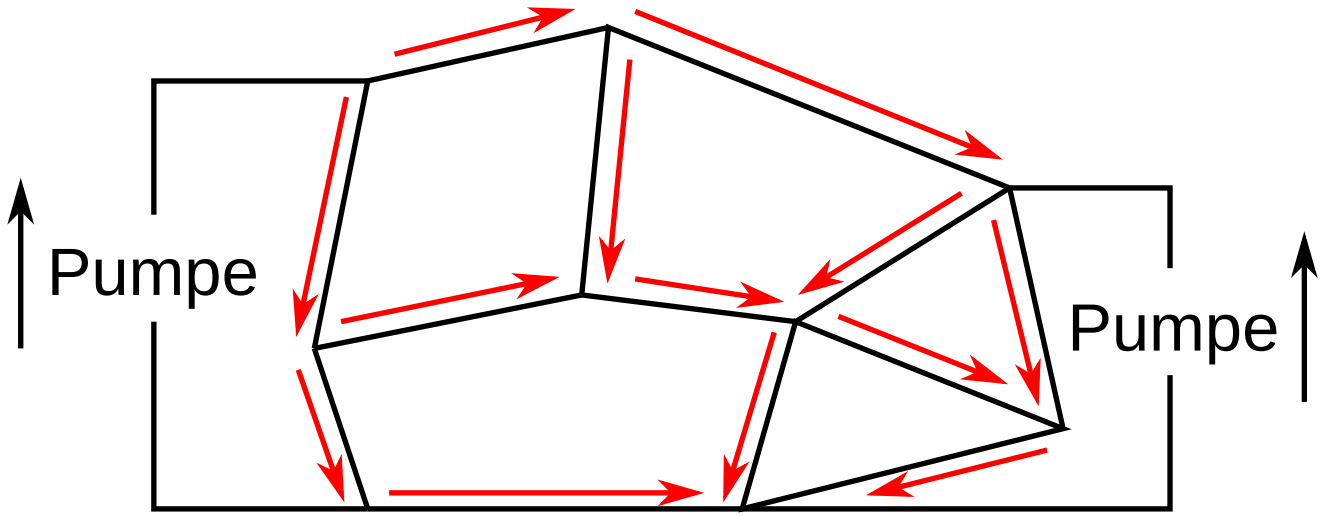
Netzwerk aus Röhren



Betrachte Netzwerk aus Röhren, in denen Flüssigkeit zirkuliert

- Um es einfach zu halten: nur durch Pumpen getrieben, keine Quellen und Senken im Kreislauf vorhanden
- Mögliche Anwendungen: Trinkwasserversorgung, elektrische Schaltkreise, Blutkreislauf (als Näherung)

Netzwerk aus Röhren



Beschrieben durch *gerichteten, (schwach) zusammenhängenden* Graphen $G = (V, E)$:

- Knotenmenge $V = \{v_1, \ldots, v_n\}$, $|V| = n$
- Kantenmenge $E = \{e_1, \ldots, e_M\} \subset V \times V$, $|E| = M$
- gerichtet "von v nach w": $(v, w) \in E \implies (w, v) \notin E$
- Röhren $E_R = \{e_1, \ldots, e_m\}$, Pumpen $E_P = \{e_{m+1}, \ldots, e_M\}$

Gesetz von Hagen-Poiseuille

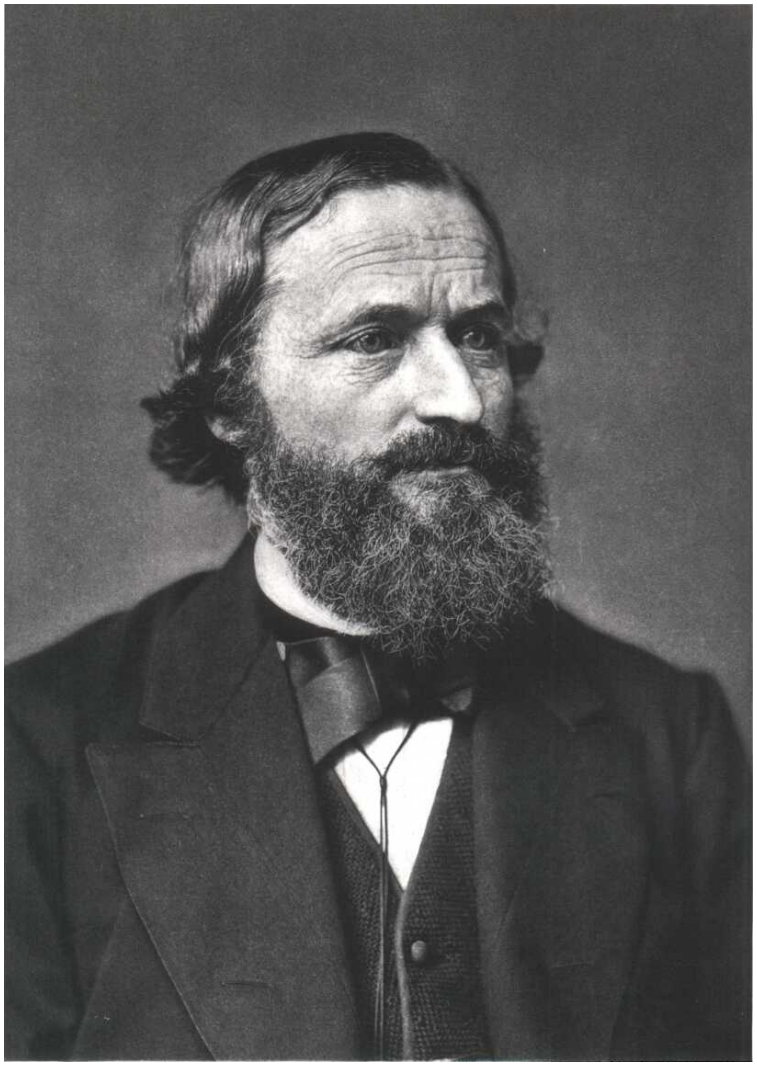
Fluss durch lange zylindrische Röhre $e = (v, w)$:

$$q_e = \frac{\pi r_e^4}{8\rho d_e} \Delta p_e$$

q_e:	gerichteter Volumenstrom [m^3/s]
Δp_e:	gerichtete Druckdifferenz [$\mathrm{Pa} = \mathrm{N}/\mathrm{m}^2$]
r_e:	Radius der Röhre e [m]
d_e:	Länge der Röhre e [m]
ρ:	dynamische Viskosität [Pa s]

$q_e > 0, \Delta p_e > 0$ falls Fluss von v nach w, $q_e < 0, \Delta p_e < 0$ sonst

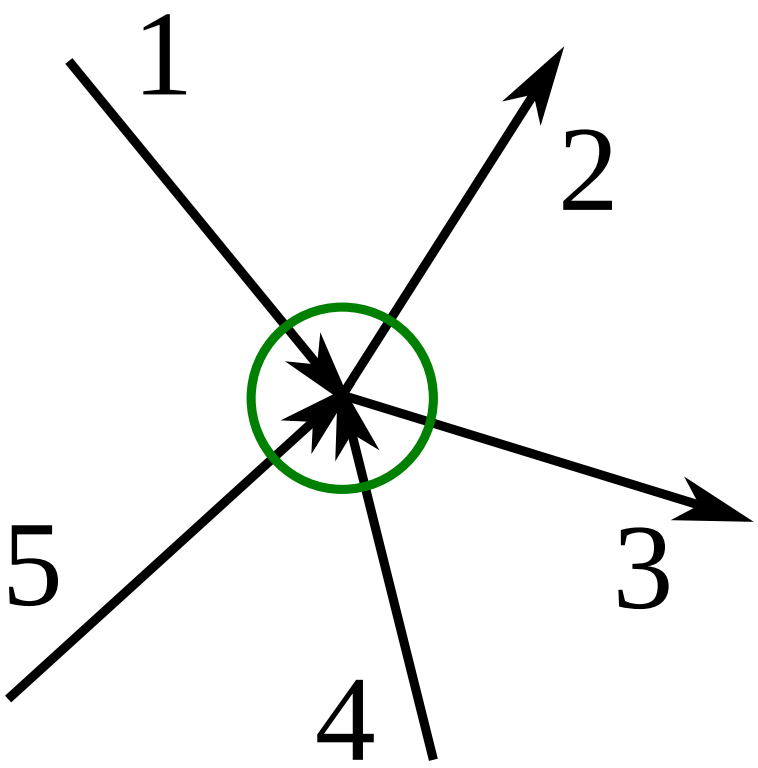
Die Kirchhoffschen Gesetze



Gustav Robert Kirchhoff (1824–1887):

- deutscher Physiker
- Beiträge zur Erforschung von
  - elektrischen Schaltkreisen
  - Spektralanalyse
  - Schwarzkörperstrahlung
- Kirchhoffsche Gesetze
  (Erhaltungsregeln für Schaltkreise)

Knotenregel (1. Kirchhoffsches Gesetz)



$$q_1 + q_4 + q_5 = q_2 + q_3$$

Knotenregel (Erhaltung der Ladung):
Die Summe der zufließenden Ströme ist für jeden Knoten eines elektrischen Netzwerks gleich der Summe der abfließenden Ströme.

Hier: Massenerhaltung
Flüssigkeit wird in den einzelnen Knoten weder vernichtet noch zwischengespeichert.

Knotenregel (1. Kirchhoffsches Gesetz)

Definiere für Knoten $v \in V$ *eingehende* und *ausgehende* Kanten:

$$E_v^+ := \{(u, w) \in E \mid w = v\}, \quad E_v^- := \{(u, w) \in E \mid u = v\}$$

Die Trennung in ein- und ausgehend basiert auf der Orientierung der Kanten und hat nichts mit dem physikalischen Fluss zu tun!

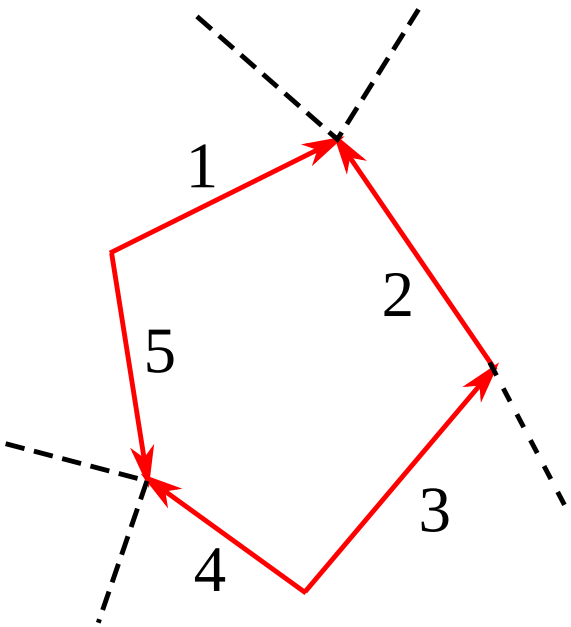
Knotenregel:

$$\sum_{e \in E_v^+} q_e - \sum_{e \in E_v^-} q_e = 0 \quad \forall v \in V \tag{4.1}$$

Das definiert $n - 1 = |V| - 1$ linear unabhängige Bedingungen, da sich aufgrund globaler Massenerhaltung die Erhaltung im letzten Knoten jeweils automatisch ergibt.

$\rightsquigarrow$ Beweis: Tafel

Maschenregel (2. Kirchhoffsches Gesetz)



$$\Delta p_3 + \Delta p_2 = \Delta p_4 - \Delta p_5 + \Delta p_1$$

Maschenregel (Energieerhaltung):
Die Summe der Spannungsdifferenzen entlang eines geschlossenen Pfades ist Null. Anders gesagt: Die Spannungsdifferenz zwischen zwei Knoten ist wegunabhängig.

(Sonst könnte man entlang des Pfades beliebig Energie erzeugen $\Longrightarrow$ Perpetuum Mobile)

Hier: Energieerhaltung
Es gibt keine spontanen (d.h. nicht durch Pumpen verursachte) Kreisbewegungen von Flüssigkeit.

Maschenregel (2. Kirchhoffsches Gesetz)

Betrachte $C \subset E$, c beliebiger *geschlossener* Pfad. Definiere:

$$C^+ := \{e \in C \mid e \text{ genauso wie } C \text{ orientiert}\}$$
$$C^- := \{e \in C \mid e \text{ entgegen } C \text{ orientiert}\}$$

(Anmerkung: wir vernachlässigen in der Notation, dass Kanten in C mehrfach vorkommen dürfen, evtl. mit geänderter Orientierung (!))

Maschenregel:

$$\sum_{e \in C^+} \Delta p_e - \sum_{e \in C^-} \Delta p_e = 0 \quad \forall \text{ geschl. Pfade } C \subset E \tag{4.2}$$

Knotenpotentialverfahren

Grundidee des Knotenpotentialverfahrens:

1. Wegen der Maschenregel ist Potentialdifferenz zwischen je zwei Knoten *wegunabhängig* und somit wohldefiniert
   $\rightsquigarrow$ Druckdifferenz zu (beliebigem) Referenzknoten als Variablen
2. Da das Potential nur bis auf Konstante eindeutig ist, können wir die Referenz als Null definieren
   $\rightsquigarrow$ $n-1$ Knotendrücke p_v als Unbekannte
3. Wegen der Knotenregel gibt es $n-1$ unabh. Nebenbedingungen
   $\rightsquigarrow$ $n-1$ lineare Gleichungen für die Unbekannten

Stelle System auf und löse nach Unbekannten (Druckdifferenzen) auf, daraus kann man dann die Flüsse berechnen.

Knotenpotentialverfahren

Wähle Referenzknoten r, ergibt Druckdifferenzen Δp_e der Röhren:

$$\forall e = (v, w) \in E_R \colon \Delta p_e = \begin{cases} p_w - p_v & v \neq r \wedge w \neq r \\ -p_v & w = r \,(\text{setze } p_w = 0) \\ p_w & v = r \,(\text{setze } p_v = 0) \end{cases}$$

Wähle feste *Anordnung* $e_k, k \in \{1, \ldots, m\}$, $v_i, i \in \{1, \ldots, n\}$ der Kanten und Knoten mit Wahl $r = v_n$

$$\implies p = (p_{v_1}, \ldots, p_{v_{n-1}})^T,\ \Delta p = (\Delta p_{e_1}, \ldots, \Delta p_{e_m})$$

Als lineares Gleichungssystem:

$$[\Delta p]_m = [B]_{m \times (n-1)} \cdot [p]_{n-1}$$

Knotenpotentialverfahren

Berechne resultierende Flüsse über Leitfähigkeiten (Hagen-Poiseuille):

$$\forall e \in E_R \colon q_e = L_e \cdot \Delta p_e$$

Als lineares Gleichungssystem mit Diagonalmatrix L:

$$[q]_m = [L]_{m \times m} \cdot [\Delta p]_m$$

Setze Umrechnung von p zu Δp ein:

$$[q]_m = [L]_{m \times m} \cdot \Big([B]_{m \times (n-1)} \cdot [p]_{n-1} \Big) = [L \cdot B]_{m \times (n-1)} \cdot [p]_{n-1}$$

Knotenpotentialverfahren

Verwende Knotenregel Gl. (4.1):

$$\sum_{e \in E_v^+ \cap E_R} q_e - \sum_{e \in E_v^- \cap E_R} q_e = \sum_{e \in E_v^- \cap E_P} q_e - \sum_{e \in E_v^+ \cap E_P} q_e$$

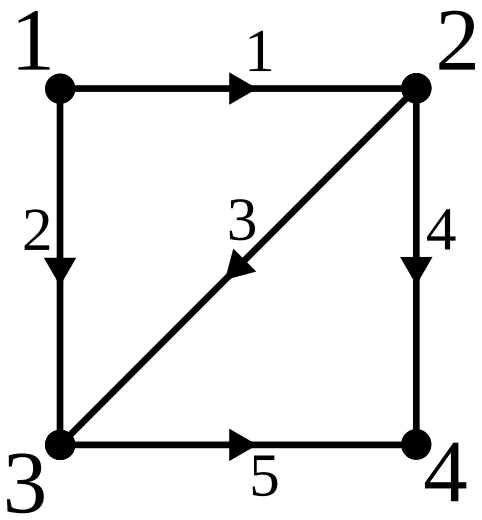
Als lineares Gleichungssystem formuliert:

$$[B^T]_{(n-1)\times m} \cdot [q]_m = [b]_{n-1},$$

wobei $b = (b_{v_1}, \ldots, b_{v_{n-1}})^T$ die Pumpströme enthält (die Einträge sind Null für innere Knoten)

Es handelt sich um B^T, weil eine Kante e genau dann zur Bilanz von v beisteuert, wenn v (mit gleichem Vorzeichen) zur Druckdifferenz entlang e beiträgt

Kleines Beispiel



Resultierende Matrix B für Beispielnetzwerk:

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \rightsquigarrow B = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

- Zeilen: Knoten, die von i-ter Kante verbunden werden (mit Vorzeichen)
- Spalten: Kanten, die für j-ten Knoten ein- / ausgehend sind (mit Vorzeichen)
- Streiche letzte Spalte wegen $p_4 := 0$ (für B) bzw. wegen redundanter Knotenregel (für B^T)

Knotenpotentialverfahren

Insgesamt:

$$\begin{aligned}[b]_{n-1} &= [B^T]_{(n-1)\times m} \cdot \Big([L \cdot B]_{m\times(n-1)} \cdot [p]_{n-1} \Big) \\ &= [B^T \cdot L \cdot B]_{(n-1)\times(n-1)} \cdot [p]_{n-1}\end{aligned}$$

Eigenschaften des Systems: $A := B^T L B$ ist

- symmetrisch und positiv definit
- damit insbesondere invertierbar
- dünn besetzt (Anzahl Kanten meist $\mathcal{O}(n)$, nicht $\mathcal{O}(n^2)$)

$\rightsquigarrow$ Beweis: Tafel

Anmerkungen

Das Ohmsche Gesetz besagt:

$$I_e = R_e^{-1}(U_v - U_w)$$

mit I_e dem elektrischen Strom entlang Leiter e, R_e dem Widerstand entlang e, und U_v der elektrischen Spannung im Knoten v.

$\Longrightarrow$ die hergeleitete Methode lässt sich völlig analog auf elektrische Netzwerke anwenden

Falls das Netzwerk Spulen oder Kondensatoren enthält (RLC-Netzwerk), werden die Zustandsgrößen komplex:

$$Ax = b \quad \text{mit } x, b \in \mathbb{C}^n,\ A \in \mathbb{C}^{n \times n}$$

Grenzübergang zum Kontinuum

Wenn man das Röhren- / Leitungsnetz immer feiner und feiner macht, beschreibt man im Limes ein unendlich feines Netzwerk, also ein Kontinuum. Die zugehörige Gleichung ist die Poisson-Gleichung:

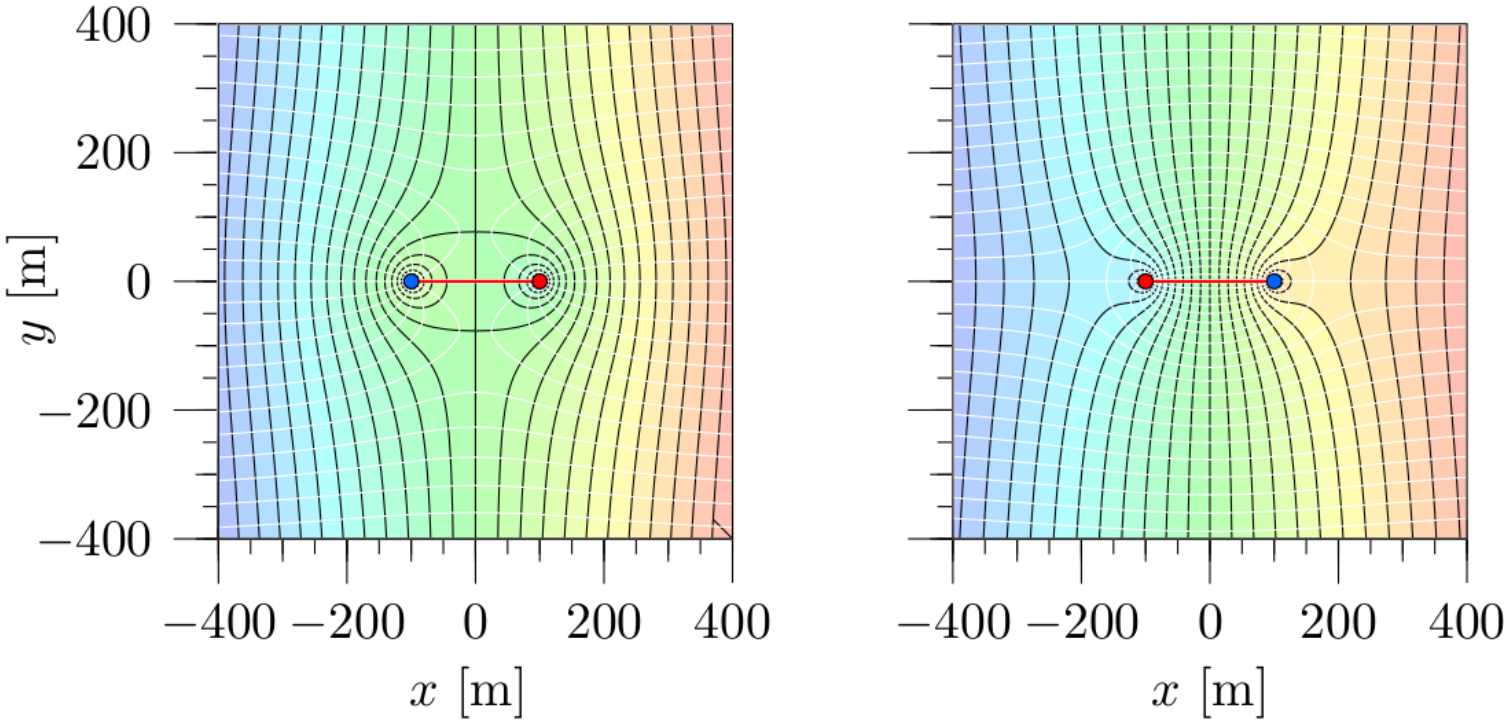
$$-\Delta u := -\nabla \cdot (\nabla u) := -\sum_{i=1}^{n} \partial_{x_i}^2 u := -\sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} = f$$

Das ist ebenfalls ein lineares Gleichungssystem, allerdings über einem *unendlich-dimensionalen Funktionenraum*.

Anwendungen der Gleichung sind zum Beispiel:

- Elektrostatik im Vakuum (f: Ladungsdichte, u: resultierendes Potential)
- Gravitation (f: Massendichte, u: Gravitationspotential)

Grenzübergang zum Kontinuum



Quelle: K. Roth, Soil Physics Lecture Notes, IUP, Universität Heidelberg

Extraktion und Injektion (roter / blauer Punkt) in einem homogenen Grundwasserleiter mit Hintergrundströmung.

Alternativ: Potential zweier Punktladungen in einem Kondensator.

Poröse Medien

Falls die Leitfähigkeit κ (Inverse des Widerstandes) ortsabhängig ist, ergibt sich die Darcy-Gleichung:

$$-\nabla \cdot (\kappa \nabla u) = -\sum_{i=1}^{n} \partial_{x_i} (\kappa \partial_{x_i} u) = f$$

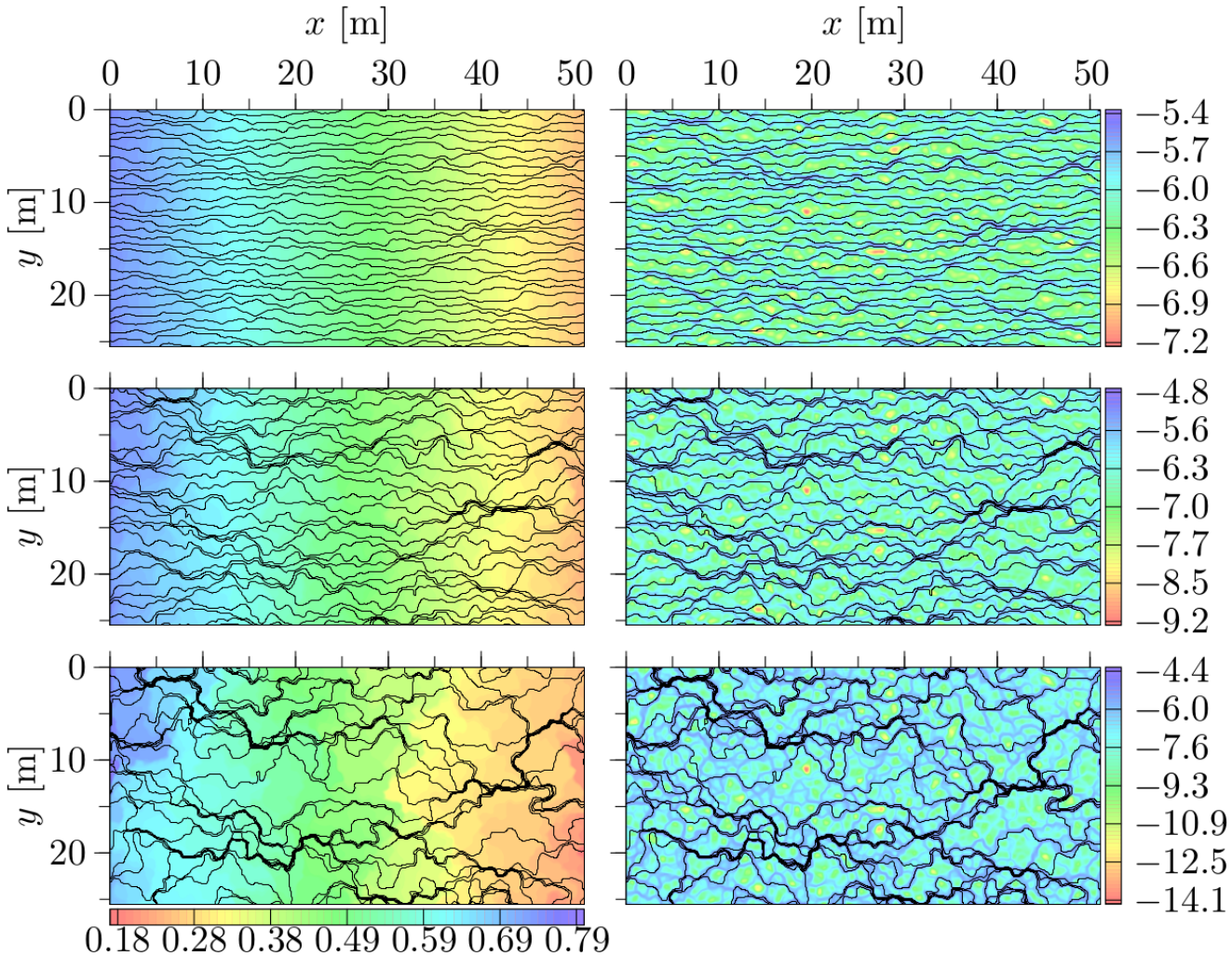
Im Allgemeinen lässt sich diese Gleichung nicht geschlossen lösen, daher verwendet man numerische Methoden. Die Diskretisierung führt dabei auf beliebig große lineare Gleichungssysteme.

Mögliche Anwendungen:

- Stationäre Strömungen in porösen Medien (z.B. Grundwasser)
- Elektrostatik in Bauteilen und Materialien
- Wärmeleitung durch heterogene Festkörper

Poröse Medien

Einfluss von Heterogenität auf die Struktur der Lösung:

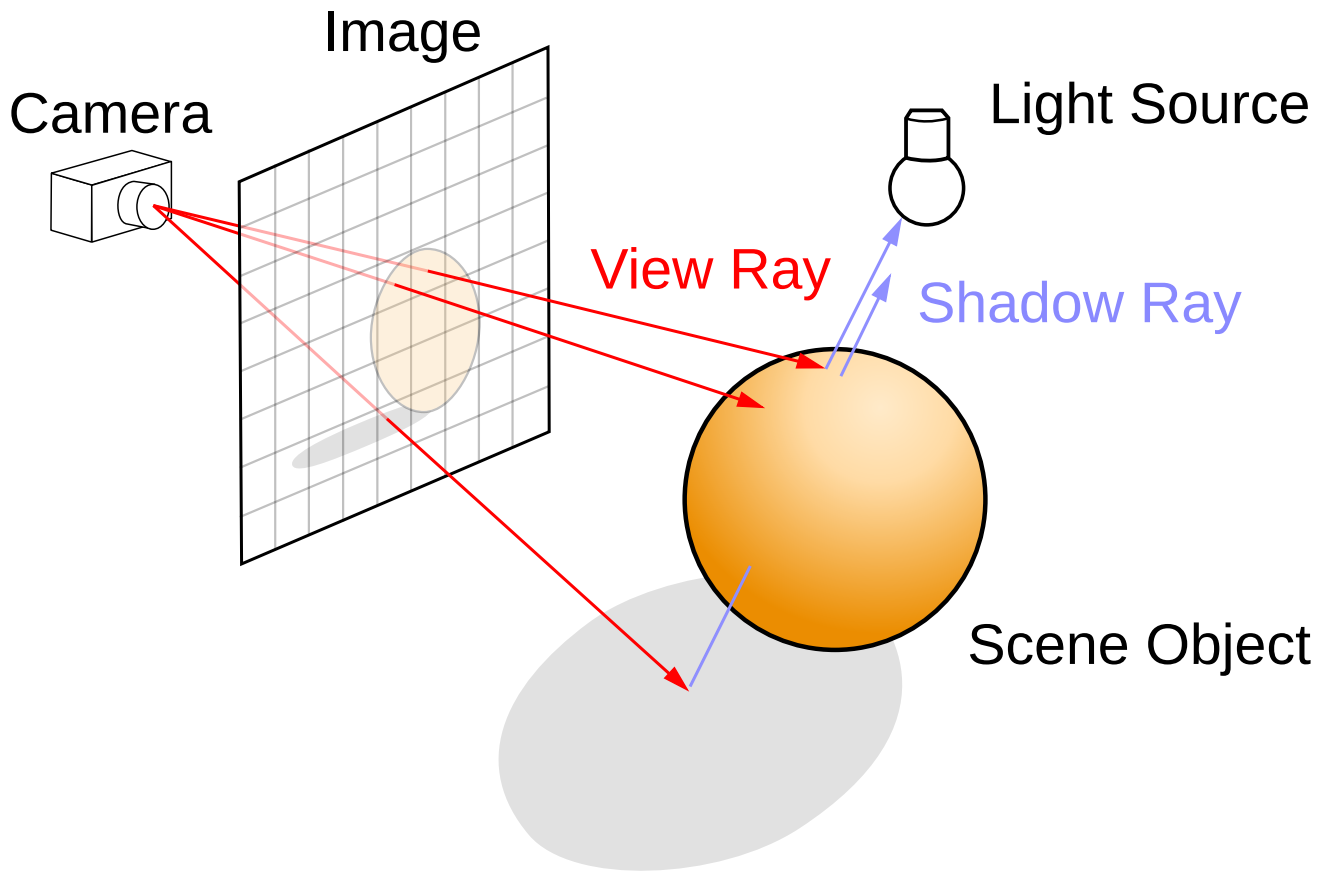


Quelle: K. Roth, Soil Physics Lecture Notes, IUP, Universität Heidelberg

Raytracing

Aufgabe: stelle beleuchtete Szene graphisch dar (Computergraphik).

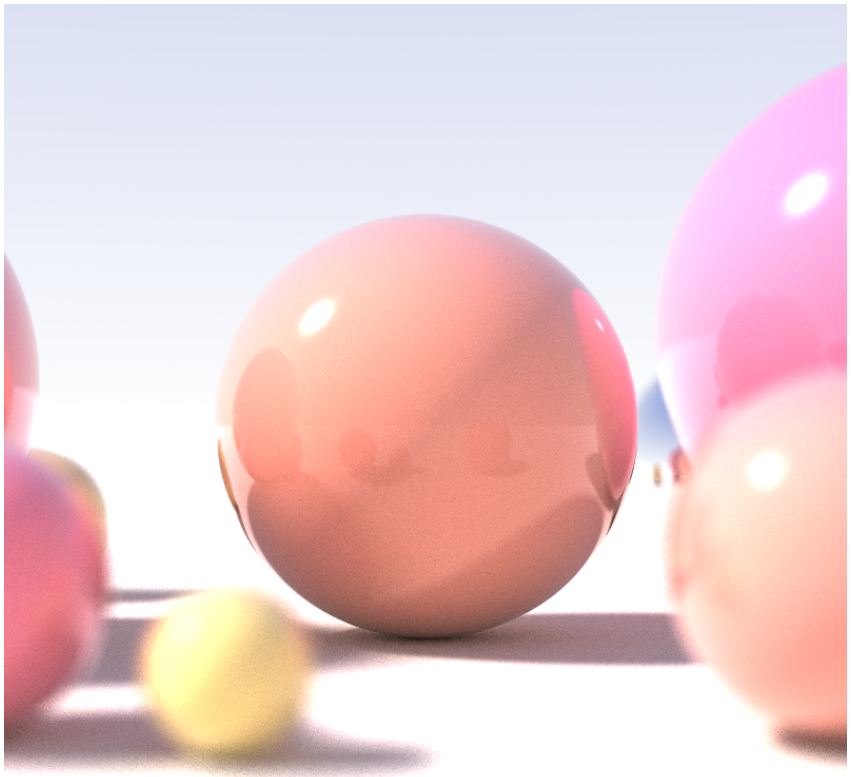
Grundidee des Raytracing-Algorithmus:



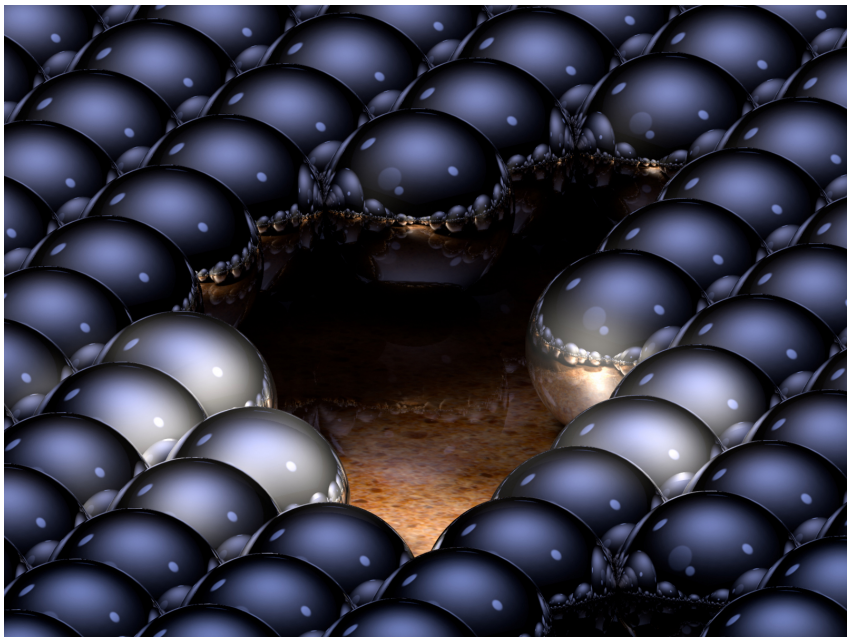
Quelle: Henrik, wikipedia.org, cc-by-sa

Raytracing

Beispiele von rekursivem Raytracing:



Quelle: Tim Babb, wikipedia.org, cc-by-sa



Quelle: Greg L, wikipedia.org, cc-by-sa

Raytracing vs. Radiosity-Methode

Idee von Raytracing:

- Verfolge "Sehstrahlen" vom Auge des Betrachters zu ihrem Ziel
- Berücksichtige dabei gerichtete Reflexion, diffuse Reflexion, Brechung, Wurf von Schatten

Vorteile / Nachteile von (rekursivem) Raytracing:

- Viele verschiedene optische Effekte darstellbar
- Aufwand wächst exponentiell mit der Anzahl der Rekursionen
- Ergebnis ist auf Blickwinkel festgelegt (!)

Raytracing vs. Radiosity-Methode

Idee der Radiosity-Methode:

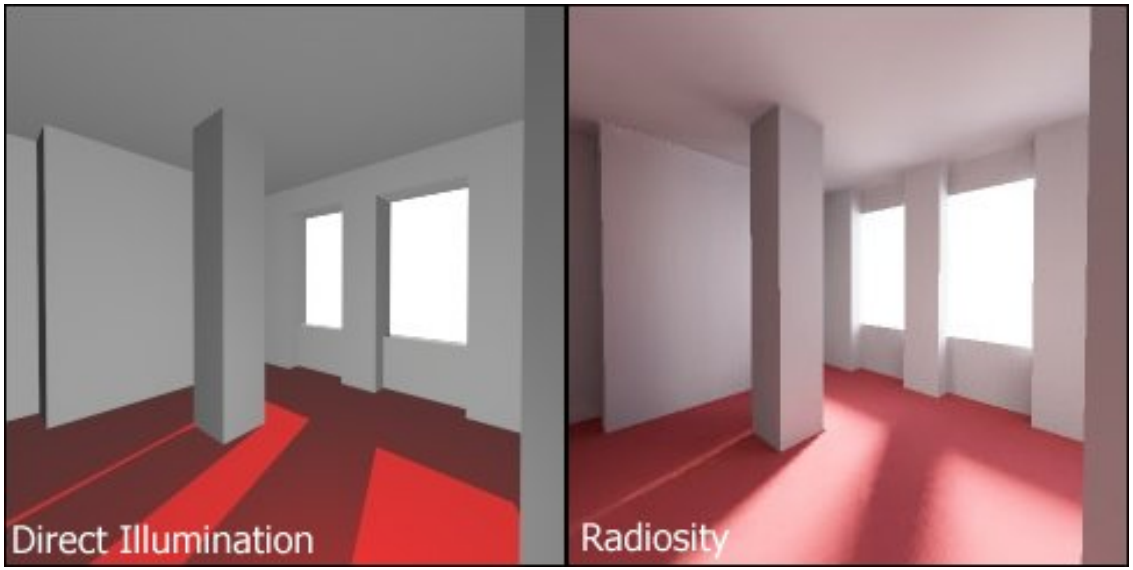
- Berechne das Strahlungsgleichgewicht für die Szene
- Beleuchtete Oberflächen erzeugen passive Beleuchtung, indem sie Licht in den Raum zurück werfen

Vorteile / Nachteile der Radiosity-Methode:

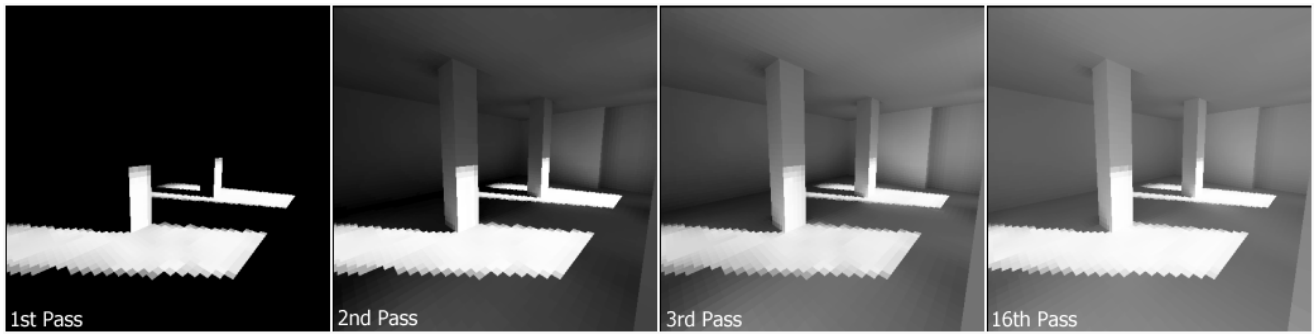
- Weniger Effekte berücksichtigt (z.B. nur diffuse Streuung)
- Aufwand ist auch bei dieser Methode sehr hoch
- Ergebnis ist unabhängig vom Betrachter ($\rightsquigarrow$ Animationen)

Radiosity-Methode

Anwendung der Radiosity-Methode:



Quelle: Hugo Elias, wikipedia.org, cc-by-sa



Quelle: Hugo Elias, wikipedia.org, cc-by-sa

Radiosity-Methode

Definiere Oberfläche der Szene:

$$S = \{x \in \mathbb{R}^3 \mid x \text{ auf Oberfläche eines Objekts}\}$$

(Lichtquellen sind dabei ebenfalls Objekte)

Bestimme "Energiedichte" $B\colon S \to \mathbb{R}$:

$$\int_\omega B \, dA = \text{von } \omega \subset S \text{ abgestrahlte Energie}$$

Benötigte Materialeigenschaften:

$E(x)$: Eigenstrahlung (bei Lichtquellen), Senke (bei Absorption)

$\rho(x)$: Reflexionsfaktor

Bestimmungsgleichung

Bestimmungsgleichung für $B(x), x \in S$:

$$B(x) = E(x) + \rho(x) \int_S B(x') \underbrace{\frac{\cos \varphi_{x,x'} \cos \varphi_{x',x}}{\|x - x'\|^2} V(x, x')}_{\text{“Kern”, “kernel” } \lambda(x, x')} \, dx'$$

mit

$$\varphi_{a,b} = \measuredangle(\nu_a, b - a)$$

$$\cos \varphi_{a,b} = \begin{cases} 1 & \text{Licht aus Richtung } \nu_a \\ 0 & \text{tangential zur Oberfläche} \end{cases}$$

$$V(x, x') = \begin{cases} 1 & x' \text{ von } x \text{ aus sichtbar} \\ 0 & \text{sonst} \end{cases}$$

(Eigenschaften der beleuchteten Szene)

Integralgleichung

Resultierende Integralgleichung:

$$B(x) - \rho(x) \int_S B(x')\lambda(x, x')\, dx' = E(x)$$

Alternative Formulierung mit Dirac-Delta $\delta(x, x')$:

$$\int_S B(x') \left(\delta(x, x') - \rho(x)\lambda(x, x')\right)\, dx' = E(x)$$

- Aufgabe: finde für gegebene Quellen / Senken $E(x)$ resultierende Energiedichte $B(x)$
- System ist erneut linear in Unbekannten $B(x)$, aber diesmal unendlichdimensional!

Kollokationsmethode

Zerlege die Oberfläche in eine endliche Menge von Teilgebieten $\mathcal{T}_h = \{t_1, \ldots, t_n\}$:

$$t_i \subset S \text{ offen}, \quad t_i \cap t_j = \emptyset \, \forall i \neq j, \quad \bigcup_{i=1}^{n} \overline{t_i} = S$$

Dieser Vorgang heißt *Diskretisierung* und ist üblich bei Integral- und Differentialgleichungen.

Definiere außerdem x_i jeweils als den Mittelpunkt von t_i.
(Existenz eines solchen Punktes schränkt implizit Wahl der t_i ein, aber wir sind sowieso nur an "simplen" Teilgebieten t_i interessiert!)

Kollokationsmethode

Approximiere $B\colon S \to \mathbb{R}$ durch diskrete Funktion $B_h\colon S \to \mathbb{R}$:

$$B_h(x) = \sum_{j=1}^{m} z_j \varphi_j(x)$$

mit $z_j \in \mathbb{R}$ *Koeffizienten* und $\varphi_j\colon S \to \mathbb{R}$ *Basisfunktionen*.
Insbesondere: $\varphi_1, \ldots, \varphi_m$ linear unabhängig.

Wir verwenden stückweise konstante Funktionen ($m = n$):

$$\varphi_j(x) = \begin{cases} 1 & \text{falls } x \in t_j \\ 0 & \text{sonst} \end{cases}$$

Kollokationsmethode

Erfülle Integralgleichung nur für $x \in X_h = \{x_1, \ldots, x_n\}$:

$$B_h(x_i) - \rho(x_i) \int_S B_h(x')\lambda(x_i, x')\,dx = E(x_i) \quad i = 1, \ldots, n$$

$$\iff \sum_{j=1}^{n} z_j \varphi_j(x_i) - \rho(x_i) \int_S \sum_{j=1}^{n} z_j \varphi_j(x')\lambda(x_i, x')\,dx = E(x_i)$$

$$\iff \sum_{j=1}^{n} z_j \underbrace{\left\{ \varphi_j(x_i) - \rho(x_i) \int_S \varphi_j(x')\lambda(x_i, x')\,dx \right\}}_{=:a_{ij}} = E(x_i)$$

$$\iff Az = b$$

mit $A = [a_{ij}]_{i,j=1}^{n,n}$, $z = [z_j]_{j=1}^{n}$, $b = [E(x_i)]_{i=1}^{n}$.

Anmerkungen

- Das Integral in a_{ij} wird i.d.R. ebenfalls numerisch berechnet (Stoff der Vorlesung).
- Man begeht dabei Diskretisierungsfehler

$$\|B_h - B\| = \mathcal{O}(h^\alpha)$$

mit $\|\cdot\|$ Norm auf Funktionenräumen, $h = \max_{t \in \mathcal{T}_h} \operatorname{diam}(t)$, und α "Konvergenzordnung".

$\Longrightarrow$ bessere Approx. B_h von B bei feinerer Unterteilung $\mathcal{T}_h$

$\Longrightarrow$ Gleichungssysteme können beliebig groß werden

- Die Matrix A ist in diesem Fall *nicht* dünn besetzt, sondern voll besetzt! Das ist wichtig bei der Wahl des Lösers.

Abschnitt 5

5 Kondition linearer Gleichungssysteme
- Lösbarkeit linearer Gleichungssysteme
- Vektornormen
- Matrixnormen
- Eigenwerte und Eigenvektoren
- Die Spektralnorm
- Positiv definite Matrizen
- Störungstheorie

Lösbarkeit linearer Gleichungssysteme

Gegeben seien Matrix $A \in \mathbb{K}^{m \times n}$ und Vektor $b \in \mathbb{K}^m$:

$$A = [a_{ij}]_{i,j=1}^{m,n} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, \quad b = [b_i]_{i=1}^{m} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Gesucht ist $[x_j]_{j=1}^{n} = [x_1, \ldots, x_n]^T \in \mathbb{K}^n$, so dass

$$\forall i: \sum_{j=1}^{n} a_{ij} x_j = b_i,$$

beziehungsweise in Kurzschreibweise:

$$Ax = b \tag{5.1}$$

Im folgenden ist stets $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$.

Lösbarkeit linearer Gleichungssysteme

Das Gleichungssystem (5.1) heißt

- *unterbestimmt*, falls $m < n$
- *quadratisch*, falls $m = n$
- *überbestimmt*, falls $m > n$

Es gibt mindestens eine Lösung, falls

$$\operatorname{rang}(A) = \operatorname{rang}([A|b]) = \operatorname{rang}\left(\begin{bmatrix} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{bmatrix}\right)$$

(Rang = Dimension Spaltenraum = Dimension Zeilenraum)

Lösbarkeit linearer Gleichungssysteme

Für *quadratische* Matrizen sind folgende Aussagen äquivalent:

1. $Ax = b$ ist für jedes b eindeutig lösbar
2. $\text{rang}(A) = n$ (Vollrang)
3. $\det(A) \neq 0$ (regulär)
4. A hat keinen Eigenwert Null

Vektornormen

Die Quantifizierung von Fehlern erfordert Normen.

Definition 5.1 (Vektornorm)

Eine Abbildung $\|\cdot\|: \mathbb{K}^n \to \mathbb{R}_+$ heißt *Norm*, falls gilt:

- **(N1)** Definitheit: $\|x\| > 0$ für $x \neq 0$
- **(N2)** Positive Homogenität:
  $\|\alpha \cdot x\| = |\alpha| \cdot \|x\|$, $x \in \mathbb{K}^n$, $\alpha \in \mathbb{K}$
- **(N3)** Subadditivität: $\|x + y\| \leq \|x\| + \|y\|$, $x, y \in \mathbb{K}^n$
  (Dreiecksungleichung)

Häufig verwendete Normen

Beispiel 5.2 (Häufig verwendete Normen)

Euklidische Norm, ℓ_2-Norm:

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$$

Maximumsnorm, ℓ_∞-Norm:

$$\|x\|_\infty = \max_{1,\ldots,n} |x_i|$$

ℓ_1-Norm:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

Folgerungen

Konvergenz in der Maximumsnorm entspricht komponentenweiser Konvergenz:

$$\forall i = 1, \ldots, n\colon \lim_{t\to\infty} x_i^{(t)} = x_i \iff \lim_{t\to\infty} \|x^{(t)} - x\|_\infty = 0$$

Gilt auch im Unendlichdimensionalen, heißt dort aber aus naheliegenden Gründen Supremumsnorm.

Direkte und wichtige Folge von (N3) ist die *inverse Dreiecksungleichung* für Normen $\|\cdot\|$:

$$\|x - y\| \geq |\|x\| - \|y\||$$

$\Longrightarrow$ Stetigkeit der Norm als Abbildung von $\mathbb{K}^n$ nach $\mathbb{R}_+$

Äquivalenz aller Normen

Satz 5.3 (Äquivalenz aller Normen)

Seien $\|\cdot\|, \|\cdot\|'$ Normen auf dem endlich-dimensionalen (!) $\mathbb{K}^n$. Dann gibt es Zahlen $m, M > 0$ aus $\mathbb{R}$, so dass gilt:

$$m \cdot \|x\|' \leq \|x\| \leq M \cdot \|x\|' \quad \forall x \in \mathbb{K}^n$$

Anmerkung: die Zahlen m und M sind nicht nur von den betrachteten Normen, sondern auch von der Dimension n abhängig. Insbesondere gilt die Aussage im Unendlichen nicht!

$\rightsquigarrow$ Beweis: Tafel

Schlussfolgerung

Im *Endlichdimensionalen* entspricht Konvergenz bezgl. einer beliebigen Norm $\|\cdot\|$ also Konvergenz in der Maximumsnorm, so dass wir sagen können:

Konvergenz bzgl. einer Norm $\|\cdot\|$ entspricht komponentenweiser Konvergenz:

$$\forall i = 1, \ldots, n\colon \lim_{t\to\infty} x_i^{(t)} = x_i \iff \lim_{t\to\infty} \|x^{(t)} - x\| = 0$$

Es gibt also für endlichdimensionale Vektorräume nur einen Konvergenzbegriff. Das ist für nicht endlich erzeugte Vektorräume i.A. falsch!

Normen auf $\mathbb{K}^{n\times m}$

Der $\mathbb{K}^{n\times m}$ ist ein Vektorraum, der mit $\mathbb{K}^{n\cdot m}$ identifiziert werden kann:

$$\mathbb{K}^{n\times m} \cong \mathbb{K}^{n\cdot m}$$

Daher definiert jede Norm auf $\mathbb{K}^{n\cdot m}$ auch eine Norm auf $\mathbb{K}^{n\times m}$.

$$A^{(t)} \to A\,(t\to\infty) \quad \text{meint dann} \quad a_{ij}^{(t)} \to a_{ij}\,(t\to\infty)\,\forall\, 1\le i,j\le n$$

Nicht jede dieser Normen ist mit der zusätzlichen Verknüpfung, der Matrixmultiplikation, verträglich. Wir betrachten daher Normen mit zusätzlichen Eigenschaften.

Matrixnormen

Definition 5.4 (Matrixnormen)

Eine Norm $\|\cdot\|$ auf $\mathbb{K}^{n\times n}$ heißt *verträglich* mit einer Vektornorm $\|\cdot\|$ auf $\mathbb{K}^n$, falls gilt:

$$\|Ax\| \leq \|A\| \cdot \|x\| \quad A \in \mathbb{K}^{n\times n}, x \in \mathbb{K}^n$$

Sie heißt *Matrixnorm*, wenn sie submultiplikativ ist:

$$\|AB\| \leq \|A\| \cdot \|B\| \quad A, B \in \mathbb{K}^{n\times n}$$

Die Submultiplikativität ergänzt die Subadditivität (Dreiecksungleichung), die für alle Normen gilt.

Beispiel

Beispiel 5.5

Die *Frobeniusnorm*

$$\|A\|_{\mathrm{Fr}} := \left(\sum_{i,j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

ist eine Matrixnorm, die mit der euklidischen Norm verträglich ist.

Die Eigenschaften einer Matrixnorm (Definitheit, positive Homogenität, Subadditivität, Submultiplikativität) und die Verträglichkeit kann man direkt nachrechnen ($\rightsquigarrow$ Übung).

Zugeordnete Matrixnormen

Definition 5.6 (Zugeordnete Matrixnorm)

Es sei $\|\cdot\|$ eine beliebige Vektornorm auf $\mathbb{K}^n$. Dann heißt

$$\|A\| := \sup_{x \in \mathbb{K}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|} = \sup_{x \in \mathbb{K}^n, \|x\|=1} \|Ax\|$$

die $\|\cdot\|$ *zugeordnete (oder natürliche) Matrixnorm*. Sie ist verträglich und submultiplikativ.

$\rightsquigarrow$ Beweis: Tafel

Zugeordnete Matrixnormen

Hilfssatz 5.7

Die zugeordneten Matrixnormen zu $\|\cdot\|_\infty$ und $\|\cdot\|_1$ sind:

$$\|A\|_\infty := \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}| \qquad \text{(Zeilensummennorm)}$$

$$\|A\|_1 := \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}| \qquad \text{(Spaltensummennorm)}$$

$\rightsquigarrow$ Beweis: [Rannacher, S. 104], Fall $\|\cdot\|_\infty$: Tafel

Eigenwerte und Eigenvektoren

Sei $A \in \mathbb{K}^{n\times n}$, $\lambda \in \mathbb{K}$, und $e \in \mathbb{K}^n$, so dass $Ae = \lambda e$, dann heißt λ *Eigenwert* von A und e zugehöriger *Eigenvektor*. (λ, e) heißt auch *Eigenpaar*.

Die Gleichung $(A - \lambda I)e = 0$ ist für $e \neq 0$ nur für

$$P(\lambda) = \det(A - \lambda I) = 0$$

erfüllbar. Das *charakteristische Polynom* $P(\lambda)$ hat genau n Nullstellen in $\mathbb{C}$ (Vielfachheit mitgezählt). Zu jedem Eigenwert λ gibt es mindestens einen Eigenvektor e.

Eigenwerte und Eigenvektoren

Für alle Matrizennormen $\|\cdot\|$ und Eigenwerte λ von A gilt

$$|\lambda| \leq \|A\|. \tag{5.2}$$

Also: alle Eigenwerte $\lambda \in \mathbb{C}$ liegen innerhalb eines Kreises um Null mit Radius $\|A\|$. Mit $\|A\|_\infty$ erhält man z.B. eine konkrete Abschätzung für den betragsmäßig größten Eigenwert.

Folgerung aus (5.2):

$$\|A\| \geq \rho(A) := \max\{|\lambda| \mid \lambda \text{ ist Eigenwert von } A\}$$

Die Zahl $\rho(A)$ heißt *Spektralradius* von A (leider ist das keine Norm).

$\rightsquigarrow$ Beweis: Tafel

Beispiele I

Beispiel 5.8

- Vandermonde-Matrix für die Stützstellen $2, 3, 4$:

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} \rightsquigarrow \begin{cases} \det(A) = (3-2)\cdot(4-2)\cdot(4-3) = 2 \\ \lambda_1 \approx 18.67, \lambda_2 \approx 1.25, \lambda_3 \approx 0.0859 \end{cases}$$

$$\rho(A) \approx 18.67, \|A\|_{\mathrm{Fr}} \approx 19.62, \|A\|_\infty = 21, \|A\|_1 = 29$$

- Drehstreckung um $/$ mit $(\theta = 45, r = 2^{1/2})$:

$$A = 2^{1/2} \cdot \begin{bmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \rightsquigarrow \begin{cases} \lambda_{1,2} = 1 \pm i \\ e_{1,2} = (\pm i, 1)^T \end{cases}$$

$$\rho(A) = 2^{1/2}, \|A\|_{\mathrm{Fr}} = \|A\|_\infty = \|A\|_1 = 2$$

Beispiele II

Beispiel 5.8

- Nilpotente Matrix:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \rightsquigarrow \lambda = 0 \,(\text{geometrische Vielfachheit } 1)$$

$$\rho(A) = 0, \|A\|_{\mathrm{Fr}} = \|A\|_\infty = \|A\|_1 = 1$$

- $A \neq 0, \rho(A) \implies \rho$ ist nicht definit
- $\rho(A + A^T) = 1 > 0 = \rho(A) + \rho(A^T) \implies \rho$ ist nicht subadditiv

Idee: nutze Spektralradius als eine Art "Grenz-Maß" für Matrizen
Da ρ wie gezeigt keine Norm ist, funktioniert das leider nicht

Gerschgorin-Kreise

Man kann aus den Einträgen einer Matrix noch genauere Abschätzungen der Eigenwerte erhalten:

Definition 5.9 (Gerschgorin-Kreise)

Für $A \in \mathbb{C}^{n \times n}$ heißen die Mengen

$$S_i := \left\{ z \in \mathbb{C} \mid |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\}$$

Gerschgorin-Kreise von A. Jeder komplexe Eigenwert λ von A liegt in mindestens einem Gerschgorin-Kreis. (Analog kann man Kreise für die Spalten definieren.)

$\rightsquigarrow$ Beweis: Tafel

Gerschgorin-Kreise

Beispiel 5.10

$$A = \begin{bmatrix} 5 & 1 & 0 \\ 1 & 10 & 1 \\ 0 & 1 & 2 \end{bmatrix} \rightsquigarrow \lambda_1 \approx 10.31, \lambda_2 \approx 4.82, \lambda_3 \approx 1.87$$

$$\rho(A) \approx 10.31, \|A\|_{\mathrm{Fr}} \approx 11.53, \|A\|_\infty = \|A\|_1 = 12$$

Vorgehensweise: alle Eigenwerte liegen...

- Normen: ...im Kreis um Null mit Radius 11.53
- Kreise: ...in den Gerschgorin-Kreisen um 2, 5 und 10 (mit den Radien 1, 1 und 2)
- Symmetrie: ...auf der reellen Achse

$\Longrightarrow \lambda_{1,2,3} \in [1, 3] \cup [4, 6] \cup [8, 11.53]$

Spezielle Matrizen

Definition 5.11 (Spezielle Matrizen)

Zu $A \in \mathbb{K}^{n\times m}$ heißt A^T mit $(A^T)_{ij} = (A)_{ji}$ die *transponierte* Matrix und $\overline{A}$ mit $(\overline{A})_{ij} = \overline{(A)_{ij}}$ die *komplex konjugierte* Matrix.

$A \in \mathbb{K}^{n\times n}$ heißt *hermitesch*, falls $A = \overline{A}^T$, d.h. $a_{ij} = \overline{a}_{ji}$.
Reelle hermitesche Matrizen heißen *symmetrisch* (dann ist $A = A^T$).

Eine komplexw. Matrix $A \in \mathbb{C}^{n\times n}$ heißt *normal*, falls $A\overline{A}^T = \overline{A}^T A$.
Sie heißt *unitär*, falls zusätzlich $A\overline{A}^T = \overline{A}^T A = I$, also $A^{-1} = \overline{A}^T$.
Reelle unitäre Matrizen heißen *orthogonal* (dann ist $A^{-1} = A^T$).

Skalarprodukte

Definition 5.12 (Skalarprodukte)

Eine Abbildung $(\cdot,\cdot)\colon \mathbb{K}^n \times \mathbb{K}^n \to \mathbb{K}$ heißt *Skalarprodukt*, falls gilt:

(S1) Symmetrie: $(x,y) = \overline{(y,x)}$

(S2) Linearität:
$(\alpha x + \beta y, z) = \alpha(x,z) + \beta(y,z),\ x,y,z \in \mathbb{K}^n,\ \alpha,\beta \in \mathbb{K}$

(S3) Definitheit: $(x,x) > 0$ für $x \in \mathbb{K}^n \setminus \{0\}$

- (S3) nutzt, dass aus (S1) $(x,x) \in \mathbb{R}$ folgt
- Aus (S1) und (S2) folgt $(z, \alpha x + \beta y) = \overline{\alpha}(z,x) + \overline{\beta}(z,y)$, also sind Skalarprodukte *Bilinearformen* für $\mathbb{K} = \mathbb{R}$ und *Sesquilinearformen* für $\mathbb{K} = \mathbb{C}$.

$\rightsquigarrow$ Rechnung: Tafel

Skalarprodukte

Ein Skalarprodukt erzeugt immer eine zugehörige Norm

$$\|x\| := \sqrt{(x,x)},\ x \in \mathbb{K}^n.$$

Es gilt die Cauchy-Schwarzsche Ungleichung

$$|(x,y)| \leq \|x\| \cdot \|y\|,\ x,y \in \mathbb{K}^n.$$

Das Euklidische Skalarprodukt lautet

$$(x,y)_2 = \sum_{i=1}^{n} x_i \overline{y}_i = x^T \overline{y}$$

(wird im folgenden oft verwendet). Damit gilt:

$$A = \overline{A}^T \iff (Ax,y)_2 = (x,Ay)_2 \quad \forall x,y \in \mathbb{K}^n$$

(andere Schreibweise für hermitesch, heißt auch *selbstadjungiert*).

Die Spektralnorm

Die der Euklidischen Norm zugeordnete Matrixnorm $\|A\|_2$ heißt *Spektralnorm*.

Hilfssatz 5.13

Für die Spektralnorm *hermitescher* Matrizen gilt

$$\|A\|_2 = \max\{|\lambda| \mid \lambda \text{ ist Eigenwert von } A\} = \rho(A)$$

Für jede Matrix $A \in \mathbb{K}^{n\times n}$ gilt

$$\|A\|_2 = \max\left\{|\lambda|^{1/2} \mid \lambda \text{ ist Eigenwert von } \overline{A}^T A\right\}$$

$\rightsquigarrow$ Beweis: Tafel

Positiv definite Matrizen

Definition 5.14 (Positiv definite Matrizen)

Eine Matrix $A \in \mathbb{K}^{n\times n}$ heißt positiv definit (in $\mathbb{K}$), falls die beiden folgenden Eigenschaften erfüllt sind:

- $(Ax, x)_2 \in \mathbb{R} \, \forall x \in \mathbb{K}^n$ (trivial falls $\mathbb{K} = \mathbb{R}$)
- $(Ax, x)_2 > 0 \, \forall x \in \mathbb{K}^n \setminus \{0\}$ (eigentliche Bedingung)

Das definiert eine wichtige Klasse von Matrizen mit vorteilhaften Eigenschaften. Ziel: Charakterisierung positiv definiter Matrizen.

Eigenschaft 5.15

Für $A \in \mathbb{C}^{n\times n}$ gilt $(Ax, x)_2 \in \mathbb{R} \, \forall x \in \mathbb{C}^n$ genau dann, wenn A hermitesch ist.

$\rightsquigarrow$ Beweis: Übung

Positiv definite Matrizen

Positiv definite Matrizen in $\mathbb{C}^{n\times n}$ sind also immer hermitesch, aber: positiv definite Matrizen in $\mathbb{R}^{n\times n}$ sind *nicht* notwendigerweise symmetrisch.

Lemma 5.16

Eine hermitesche Matrix A ist genau dann positiv definit, wenn alle ihre (reellen) Eigenwerte positiv sind. Alle Hauptdiagonalelemente sind (reell und) positiv.

$\rightsquigarrow$ Beweis: Tafel

Positiv definite Matrizen

Lemma 5.17

Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Dann liegt der betragsmäßig größte Eintrag auf der Hauptdiagonalen.

Normalerweise beschränkt man sich auf symmetrisch positiv definite Matrizen (kurz auch s.p.d.). Manche Autoren verlangen das daher direkt in der Definition. Wir fordern das aber extra.

$\rightsquigarrow$ Beweis: Tafel

Störungstheorie

Seien $A \in \mathbb{K}^{n\times n}$ regulär und $x, b \in \mathbb{K}^n$. Dann ist $F(A, b) = A^{-1}b$ "Lösungsoperator" der Gleichung $G(x) = Ax - b = 0$.

Betrachte die relative Kondition von F, zunächst nur bzgl. Änderungen in b:

$$\frac{\|F(A, b + \delta b) - F(A, b)\|}{\|F(A, b)\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta b\|}{\|b\|}$$

Definition 5.18 (Konditionszahl)

Die Zahl $\operatorname{cond}(A) = \|A\| \cdot \|A^{-1}\|$ heißt *Konditionszahl* von A (bzgl. einer verträglichen Matrixnorm $\|\cdot\|$).

$\rightsquigarrow$ Rechnung: Tafel

Störungstheorie

Als nächstes wollen wir Störungen in A selbst zulassen. Dabei stellt sich zunächst die Frage, für welche Störungen überhaupt mit einer (eindeutigen) Lösung zu rechnen ist. Wenn A regulär ist, wann ist dann $A + \delta A$ regulär?

Hilfssatz 5.19

Sei $B \in \mathbb{K}^{n\times n}$ mit zugeordneter Matrixnorm $\|B\| < 1$. Dann ist $I + B$ regulär, und es gilt

$$\|(I + B)^{-1}\| \leq (1 - \|B\|)^{-1}$$

$\rightsquigarrow$ Beweis: Tafel

Störungssatz

Satz 5.20 (Störungssatz)

Sei $A \in \mathbb{K}^{n\times n}$ regulär und $\|\delta A\| \leq \|A^{-1}\|^{-1}$. Dann ist $\widetilde{A} = A + \delta A$ ebenfalls regulär, und für den relativen Fehler des gestörten Systems

$$(A + \delta A) \cdot (x + \delta x) = b + \delta b \quad \text{(oder kurz: } \widetilde{A}\widetilde{x} = \widetilde{b}\text{)}$$

gilt:

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\operatorname{cond}(A)}{1 - \operatorname{cond}(A) \cdot \frac{\|\delta A\|}{\|A\|}} \cdot \left\{ \frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right\}$$

$\rightsquigarrow$ Beweis: Tafel

Beispiele

Beispiel 5.21

Es sei $\frac{\|\delta A\|}{\|A\|} \approx 10^{-k}$ und $\frac{\|\delta b\|}{\|b\|} \approx 10^{-k}$, sowie $\text{cond}(A) \approx 10^s$ (mit $k, s > 0$). Weiter nehmen wir an, dass $10^s \cdot 10^{-k} \ll 1$, also z.B. $s - k \leq -3$.
Dann gilt

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{10^s}{1 - 10^s \cdot 10^{-k}} \cdot 2 \cdot 10^{-k} \approx 10^{-k+s}$$

Man verliert also bis zu s Stellen an Genauigkeit!

Beispiele

Beispiel 5.22 (Kondition und Determinante)

- Betrachte Matrix $B = 10^{-10} \cdot I$:

$$B = \begin{bmatrix} 10^{-10} & 0 \\ 0 & 10^{-10} \end{bmatrix}, B^{-1} = \begin{bmatrix} 10^{10} & 0 \\ 0 & 10^{10} \end{bmatrix}$$

  Es gilt $\text{cond}_\infty(B) = \|B\|_\infty \cdot \|B^{-1}\|_\infty = 1$, aber $\det(B) = 10^{-20}$!
- Generell gilt $\text{cond}(\epsilon A) = \text{cond}(A)$ für $\epsilon \neq 0$, aber $\det(\epsilon A) = \epsilon^n \det(A)$

$\Longrightarrow$ Die Determinante einer Matrix sagt nichts über die Fehlerverstärkung beim Lösen des zugehörigen Gleichungssystems aus

Beispiele

Beispiel 5.23

Für $A = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}$ ergibt sich $\operatorname{cond}_\infty(A) = \max(d_1, d_2)/\min(d_1, d_2)$, also z.B. 10^{10} für die Wahl $d_1 := 10^{10}, d_2 := 1$. Andererseits sind alle Gleichungen in A unabhängig und $x_i = b_i/d_i$ jeweils gut konditioniert. Was passiert hier?

- Bei Fehlern nur in b: das Problem liegt in der Verwendung von Normen, es handelt sich nur um eine *Abschätzung*, und diese ist für die einzelnen Komponenten hier sehr schlecht.
- Bei Fehlern auch in A: kleine Schwankungen in den Nebendiagonaleinträgen führen dazu, dass das gestörte System voll gekoppelt ist. Der Störungssatz behandelt die Auswirkungen.

Abschnitt 6

6 Eliminationsverfahren zur Lösung linearer Gleichungssysteme

- Dreieckssysteme (gestaffelte Gleichungssysteme)
- Gauß-Elimination
- LR-Zerlegung
- Rundungsfehleranalyse der LR-Zerlegung
- Pivotisierung
- Spezielle Systeme
- Nichtreguläre Systeme

Dreieckssysteme

Sei $A \in \mathbb{K}^{n \times n}$ von oberer Dreiecksgestalt:

$$\begin{aligned} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \cdots + a_{1n} \cdot x_n &= b_1 \\ a_{22} \cdot x_2 + \cdots + a_{2n} \cdot x_n &= b_2 \\ \ddots \quad \vdots \quad &\vdots \\ a_{nn} \cdot x_n &= b_n \end{aligned}$$

Dieses System ist genau dann eindeutig lösbar, wenn $a_{ii} \neq 0, i = 1, \ldots, n$, gilt.

Aufgrund der einfachen Struktur lässt sich das System dann durch "Rückwärtseinsetzen" lösen.

Dreieckssysteme

Lösung durch rückwärts einsetzen:

$$
\begin{aligned}
x_n &= b_n / a_{nn} \\
x_{n-1} &= \left(b_{n-1} - a_{(n-1)n} \cdot x_n\right) / a_{(n-1)(n-1)} \\
&\vdots \\
x_i &= \left(b_i - \sum_{k=i+1}^{n} a_{ik} \cdot x_k\right) / a_{ii}
\end{aligned}
$$

Die Anzahl der benötigten Rechenoperationen ist:

$$
N_\Delta(n) = \sum_{i=0}^{n-1} (2i + 1) = n^2
$$

(Gilt für untere Dreieckssysteme natürlich analog.)

Gauß-Elimination

Sei nun $A \in \mathbb{K}^{n \times n}$ regulär, aber mit beliebiger Struktur.

Ziel: Bringe A durch äquivalente Umformungen auf (obere) Dreiecksgestalt, um die Lösung leicht berechnen zu können.

Dazu benutzt man:

- Vertauschen zweier Gleichungen
- Addition eines Vielfachen einer Gleichung zu einer anderen

(Wir beschränken uns der Einfachheit halber im folgenden auf den Fall $\mathbb{K} = \mathbb{R}$, für die komplexen Zahlen funktioniert es aber analog.)

Gauß-Elimination

Wir schreiben

$$[A, b] = \left[\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \end{array}\right]$$

(erweiterte Koeffizientenmatrix)

Das Gaußsche Eliminationsverfahren erzeugt dann eine *endliche Folge von Matrizen*

$$[A, b] = \left[A^{(0)}, b^{(0)}\right], \left[A^{(1)}, b^{(1)}\right], \ldots, \left[A^{(n-1)}, b^{(n-1)}\right],$$

so dass nach $n-1$ Schritten $A^{(n-1)}$ obere Dreiecksgestalt hat.

Erster Schritt: $\left[A^{(0)}, b^{(0)}\right] \rightsquigarrow \left[A^{(1)}, b^{(1)}\right]$

Erster Schritt, Teil (a): $\left[A^{(0)}, b^{(0)}\right] \rightsquigarrow \left[\widetilde{A}^{(0)}, \widetilde{b}^{(0)}\right]$

- Bestimme ein $r \in \{1, \ldots, n\}$ mit $a_{r1}^{(0)} \neq 0$
- Dieses *Pivotelement* $a_{r1}^{(0)}$ existiert, weil A regulär ist
- Vertausche Zeilen 1 und r, dadurch $\widetilde{a}_{11}^{(0)} \neq 0$

Beispiel:

$$\left[\begin{array}{c|cccc} 0 & 4 & -2 & -3 & 1 \\ \hline -2 & -6 & 3 & 1 & 2 \\ 3 & 7 & 4 & -1 & 3 \\ 1 & 3 & 0 & -2 & 2 \end{array}\right] \rightsquigarrow \left[\begin{array}{c|cccc} 1 & 3 & 0 & -2 & 2 \\ \hline -2 & -6 & 3 & 1 & 2 \\ 3 & 7 & 4 & -1 & 3 \\ 0 & 4 & -2 & -3 & 1 \end{array}\right]$$

Erster Schritt: $\left[A^{(0)}, b^{(0)}\right] \rightsquigarrow \left[A^{(1)}, b^{(1)}\right]$

Erster Schritt, Teil (b): $\left[\widetilde{A}^{(0)}, \widetilde{b}^{(0)}\right] \rightsquigarrow \left[A^{(1)}, b^{(1)}\right]$

- Berechne für $i \in \{2, \ldots, n\}$ den Faktor $q_{i1} = \widetilde{a}_{i1}^{(0)} / \widetilde{a}_{11}^{(0)}$
- Ziehe das q_{i1}-fache der Zeile 1 von Zeile i ab
- Dadurch werden alle Einträge der ersten Spalte Null, bis auf den ersten (das Pivotelement)

$$\forall j \in \{1, \ldots, n\}: a_{ij}^{(1)} = \widetilde{a}_{ij}^{(0)} - q_{i1} \widetilde{a}_{1j}^{(0)}$$
$$b_i^{(1)} = \widetilde{b}_i^{(0)} - q_{i1} \widetilde{b}_1^{(0)}$$

$$\left[\begin{array}{c|cccc} 1 & 3 & 0 & -2 & 2 \\ \hline -2 & -6 & 3 & 1 & 2 \\ 3 & 7 & 4 & -1 & 3 \\ 0 & 4 & -2 & -3 & 1 \end{array}\right] \rightsquigarrow \left[\begin{array}{c|cccc} 1 & 3 & 0 & -2 & 2 \\ \hline 0 & 0 & 3 & -3 & 6 \\ 0 & -2 & 4 & 5 & -3 \\ 0 & 4 & -2 & -3 & 1 \end{array}\right]$$

k-ter Schritt: $\left[A^{(k-1)}, b^{(k-1)}\right] \rightsquigarrow \left[A^{(k)}, b^{(k)}\right]$

k-ter Schritt, Teil (a): $\left[A^{(k-1)}, b^{(k-1)}\right] \rightsquigarrow \left[\widetilde{A}^{(k-1)}, \widetilde{b}^{(k-1)}\right]$

- Die ersten $k-1$ Zeilen und Spalten sind die einer oberen Dreiecksmatrix
- Finde ein neues $r \in \{k, \ldots, n\}$ mit Pivotelement $a_{rk}^{(k-1)} \neq 0$
- Tauschen der k-ten und r-ten Zeilen ändert die ersten $k-1$ Zeilen und Spalten nicht

Beispiel für $k = 2$:

$$\left[\begin{array}{cc|ccc} 1 & 3 & 0 & -2 & 2 \\ 0 & 0 & 3 & -3 & 6 \\ \hline 0 & -2 & 4 & 5 & -3 \\ 0 & 4 & -2 & -3 & 2 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|ccc} 1 & 3 & 0 & -2 & 2 \\ 0 & -2 & 4 & 5 & -3 \\ \hline 0 & 0 & 3 & -3 & 6 \\ 0 & 4 & -2 & -3 & 2 \end{array}\right]$$

k-ter Schritt: $\left[A^{(k-1)}, b^{(k-1)}\right] \rightsquigarrow \left[A^{(k)}, b^{(k)}\right]$

k-ter Schritt, Teil (b): $\left[\widetilde{A}^{(k-1)}, \widetilde{b}^{(k-1)}\right] \rightsquigarrow \left[A^{(k)}, b^{(k)}\right]$

- Berechne für $i \in \{k+1, \ldots, n\}$ den Faktor $q_{ik} = \widetilde{a}_{ik}^{(k-1)} / \widetilde{a}_{kk}^{(k-1)}$
- Ziehe das q_{ik}-fache der Zeile k von Zeile i ab
- Diese Subtraktion ändert die $k-1$ ersten Zeilen und Spalten nicht (für die Spalten: wegen $\widetilde{a}_{kj}^{(k-1)} = 0$ für $j < k$)

$$\forall j \in \{k, \ldots, n\}: a_{ij}^{(1)} = \widetilde{a}_{ij}^{(0)} - q_{ik} \widetilde{a}_{kj}^{(0)}$$
$$b_i^{(1)} = \widetilde{b}_i^{(0)} - q_{ik} \widetilde{b}_1^{(0)}$$

$$\left[\begin{array}{cc|ccc} 1 & 3 & 0 & -2 & 2 \\ 0 & -2 & 4 & 5 & -3 \\ \hline 0 & 0 & 3 & -3 & 6 \\ 0 & 4 & -2 & -3 & 2 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|ccc} 1 & 3 & 0 & -2 & 2 \\ 0 & -2 & 4 & 5 & -3 \\ \hline 0 & 0 & 3 & -3 & 6 \\ 0 & 0 & 6 & 7 & -4 \end{array}\right]$$

Aufwand der Gauß-Elimination

- In jedem Schritt entsteht eine Zeile (und Spalte), die die korrekte Form hat und danach nicht mehr verändert wird
  $\Longrightarrow$ das Verfahren bricht nach einer *endlichen* Zahl Schritte ab
- Da es immer mindestens ein geeignetes Pivotelement gibt, lassen sich die einzelnen Teilschritte stets durchführen
  $\Longrightarrow$ das Verfahren besteht aus *einzelnen, berechenbaren* Operationen

$\rightsquigarrow$ Formaler Algorithmus

Wie aufwändig ist die Gauß-Elimination?

Aufwand der Gauß-Elimination

Lemma 6.1

Der Aufwand zur Transformation von A auf obere Dreiecksgestalt durch Gauß-Elimination beträgt

$$N_{\text{Gauß}}(n) = \frac{2}{3}n^3 + \mathcal{O}(n^2)$$

Da der Aufwand des Rückwärtseinsetzens vernachlässigbar ist ($N_\Delta(n) = n^2$), ist dies auch der Aufwand für das Lösen eines linearen Gleichungssystems mit dem Gauß-Verfahren.

$\rightsquigarrow$ Beweis: Tafel

Vollständiger Algorithmus

Algorithmus 6.2 (Gauß-Elimination)

Eingabe : $A \in \mathbb{R}^{n \times n}$ (wird überschrieben)
$b \in \mathbb{R}^n$ (wird überschrieben)
Ausgabe : $x \in \mathbb{R}^n$

```
for ( k = 1; k < n; k = k+1 )
{
  // Schritt (a)
  // Finde ein geeignetes Pivotelement
  Finde r in {k,...,n}
      mit a_rk != 0 (sonst Fehler);
  if (r != k)
  { // Tausche Zeilen r und k
    for ( j = k; j <= n; j = j+1 )
      t = a_kj; a_kj = a_rj; a_rj = t;
    t = b_k; b_k = b_r; b_r = t;
  }

  // (... -->)
  // (...)

  // Schritt (b)
  // Subtrahiere Vielfache der Zeile k
  for ( i = k+1; i <= n; i = i+1 )
  {
    q_ik = a_ik / a_kk;
    for (j = k; j <= n; j = j+1 )
      a_ij = a_ij - q_ik * a_kj;
    b_i = b_i - q_ik * b_k;
  }
}
// A ist jetzt obere Dreiecksmatrix,
// löse daher durch Rückwärtseinsetzen
```

Vorsicht: In der Numerik zählen wir üblicherweise ab Eins, aber in C++ ist Null der kleinste Index!

Beispiel

Beispiel 6.3

Ein weiteres Beispiel, diesmal ohne Vertauschungen:

$$\left[\begin{array}{cccc|c} 2 & 4 & 6 & 8 & 40 \\ 16 & 33 & 50 & 67 & 330 \\ 4 & 15 & 31 & 44 & 167 \\ 10 & 29 & 63 & 97 & 350 \end{array}\right] \rightsquigarrow \left[\begin{array}{cccc|c} 2 & 4 & 6 & 8 & 40 \\ 0 & 1 & 2 & 3 & 10 \\ 0 & 7 & 19 & 28 & 87 \\ 0 & 9 & 33 & 57 & 150 \end{array}\right]$$

$$\rightsquigarrow \left[\begin{array}{cccc|c} 2 & 4 & 6 & 8 & 40 \\ 0 & 1 & 2 & 3 & 10 \\ 0 & 0 & 5 & 7 & 17 \\ 0 & 0 & 15 & 30 & 60 \end{array}\right] \rightsquigarrow \left[\begin{array}{cccc|c} 2 & 4 & 6 & 8 & 40 \\ 0 & 1 & 2 & 3 & 10 \\ 0 & 0 & 5 & 7 & 17 \\ 0 & 0 & 0 & 9 & 9 \end{array}\right]$$

Lösung durch Rückwärtseinsetzen: $(4, 3, 2, 1)^T$

Transpositionsmatrizen

Ziel: Finde möglichst kompakte Darstellung der Gauß-Elimination
$\rightsquigarrow$ abstrakte Formulierung mit Matrixmultiplikationen

Definition 6.4 (Transpositionsmatrizen)

Eine Matrix $P_{rs} \in \mathbb{R}^{n\times n}, 1 \leq r, s \leq n, r \neq s$, mit

$$[P_{rs}]_{ij} = \begin{cases} 1 & i = j \wedge i \neq r \wedge i \neq s \\ 1 & (i = r \wedge j = s) \vee (i = s \wedge j = r) \\ 0 & \text{sonst} \end{cases}$$

Beispiel:

$$P_{24} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

heißt *Transpositionsmatrix*.

Transpositionsmatrizen

Hilfssatz 6.5

Eine Transpositionsmatrix P_{rs} hat die folgenden Eigenschaften:

1. $\widetilde{A} = P_{rs}A$ ist A mit *Zeilen* r und s vertauscht
2. $\widetilde{A} = AP_{rs}$ ist A mit *Spalten* r und s vertauscht
3. $P_{rs} = P_{rs}^T$ (symmetrisch)
4. $P_{rs}^{-1} = P_{rs} \implies P_{rs}^{-1} = P_{rs}^T$ (orthogonal)

Endliche Produkte von Transpositionsmatrizen heißen *Permutationsmatrizen*. Diese sind orthogonal, aber *nicht* symmetrisch.

$\rightsquigarrow$ Beweis: Tafel

Frobeniusmatrizen

Die Transpositionsmatrizen werden für Teil (a) der Eliminationsschritte verwendet, für Teil (b) benötigen wir eine spezielle Form von Matrix.

Definition 6.6 (Frobeniusmatrizen)

Eine Matrix $G_k \in \mathbb{R}^{n \times n}$ heißt *Frobeniusmatrix*, wenn $G_k = I + G'_k$, wobei $G'_k \in \mathbb{R}^{n \times n}$ mit $[G'_k]_{ij} = 0$ für $j \neq k \vee i \leq k$.

Beispiel:

$$G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 4 & 0 & 1 \end{bmatrix}$$

Frobeniusmatrizen

Hilfssatz 6.7

Eine Frobeniusmatrix G_k hat die folgenden Eigenschaften:

1. Für $\widetilde{A} = G_k A$ gilt $\widetilde{a}_{ij} = \begin{cases} a_{ij} & i \leq k \\ a_{ij} + g_{ik} a_{kj} & i > k \end{cases}$
   (Das ist gerade der Eliminationsschritt, wenn $g_{ik} := -q_{ik}$)
2. $G'_k G'_k = 0$
3. $G_k^{-1} = (I - G'_k)$
4. $G_1 G_2 \cdots G_k = I + \sum_{j=1}^{k} G'_j$ und $G_1^{-1} G_2^{-1} \cdots G_k^{-1} = I - \sum_{j=1}^{k} G'_j$
5. $\forall\, k < r \leq s$: $P_{rs} G_k = \widetilde{G}_k P_{rs}$ mit $\widetilde{G}_k := I + P_{rs} G'_k$
   (Ändern der Reihenfolge führt zu neuer Frobeniusmatrix $\widetilde{G}_k$)

$\rightsquigarrow$ Beweis: Tafel

LR-Zerlegung

Für das Produkt von Matrizen $B_i, a \leq i \leq b$, definieren wir:

$$\prod_{i=a}^{b} B_i := B_b B_{b-1} \cdots B_{a+1} B_a$$

(Achtung: *von rechts nach links*, da Multiplikation nicht kommutativ)

Für Transpositionsmatrizen ist dieses Produkt eine Permutationsmatrix, für Frobeniusmatrizen nach dem eben gezeigten eine untere Dreiecksmatrix.

LR-Zerlegung

Satz 6.8 (LR-Zerlegung)

Sei $A \in \mathbb{R}^{n\times n}$ regulär, dann gibt es eine Zerlegung $PA = LR$, wobei

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{11} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{n(n-1)} & 1 \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \ddots & r_{2n} \\ \vdots & \ddots & \ddots & r_{(n-1)n} \\ 0 & \cdots & 0 & r_{nn} \end{bmatrix},$$

und $P = \prod_{\alpha=1}^{n-1} P_{\alpha,s_\alpha}, s_\alpha \geq \alpha$, eine Permutationsmatrix ist. Im Fall $P = I$ ist die Zerlegung eindeutig.

$\rightsquigarrow$ Beweis: Tafel

LR-Zerlegung

Aus dem Beweis ergibt sich, dass

$$L = \prod_{k=n-1}^{1} \left(I - \left(\prod_{\alpha=k+1}^{n-1} P_{\alpha,s_\alpha} \right) G'_k \right)$$
$$= I - \sum_{k=1}^{n-1} \left(\prod_{\alpha=k+1}^{n-1} P_{\alpha,s_\alpha} \right) G'_k \quad \text{mit} \quad [G'_k]_{ik} = -q_{ik}$$

- Die Nichtnullelemente $-q_{ik}$ in G'_k nehmen den selben Platz wie die eliminierten Einträge $\widetilde{a}_{ik}^{(k-1)}$ ein
- Kein zusätzlicher Speicher notwendig, verwende einfach die linke untere Hälfte von A (wird sowieso überschrieben)
- Die Einträge von G'_k nehmen ebenfalls an späteren Zeilenvertauschungen teil (tausche daher einfach ganze Zeile)

Wozu das Ganze?

Lösen eines Gleichungssystems $Ax = b$ mittels LR-Zerlegung:

$$
\begin{aligned}
& Ax = b \\
\iff & PAx = Pb \\
\iff & LRx = Pb, \quad y := Rx \\
\iff & \begin{cases} Ly = Pb & \text{(vorwärts einsetzen)} \\ Rx = y & \text{(rückwärts einsetzen)} \end{cases}
\end{aligned}
$$

Bei bekannter LR-Zerlegung lässt sich das Gleichungssystem also mit zwei sehr einfachen Schritten lösen.

Wozu das Ganze?

Einzelne Schritte:

1. Berechne LR-Zerlegung und Matrix P für gegebenes A
2. berechne $b' = Pb$ für gegebenes b
3. Löse Dreieckssystem $Ly = b'$
4. Löse Dreieckssystem $Rx = y$

Die LR-Zerlegung ist äquivalent zur Gauß-Elimination und hat daher den selben Aufwand $N_{LR}(n) = \frac{2}{3}n^3 + \mathcal{O}(n)$.

Wichtiger Unterschied:
Für eine neue rechte Seite $A\widetilde{x} = \widetilde{b}$ kann die LR-Zerlegung wiederverwendet werden, die Gauß-Elimination müsste erneut durchgeführt werden!

Algorithmus

Algorithmus 6.9 (LR-Zerlegung)

Eingabe : $A \in \mathbb{R}^{n\times n}$ (wird überschrieben)
Ausgabe : $L, R \in \mathbb{R}^{n\times n}$ (ohne Diagonale von L)
$p\colon \{1,\dots,n\} \to \{1,\dots,n\}$

```
for ( k = 1; k < n; k = k+1 )
{
  // Schritt (a)
  // Finde ein geeignetes Pivotelement
  Finde r in {k,...,n}
      mit a_rk != 0 (sonst Fehler);
  if (r != k)
  { // Tausche Zeilen r und k
    // Tausche jetzt auch alte Spalten!
    for ( j = 1; j <= n; j = j+1 )
      t = a_kj; a_kj = a_rj; a_rj = t;
  }

  p_k = r; // merke Permutation
  // (... -->)
  // (...)

  // Schritt (b)
  // Subtrahiere Vielfache der Zeile k
  for ( i = k+1; i <= n; i = i+1 )
  { // Verwende  jetzt a_ik statt q_ik
    a_ik = a_ik / a_kk;
    // Subtrahiere weiterhin ab Spalte k
    // (sonst würden q_ij verändert!)
    for (j = k; j <= n; j = j+1 )
      a_ij = a_ij - a_ik * a_kj;
  }
}
```

(Wir speichern nicht die kompletten Transpositionsmatrizen, sondern nur den Index, mit dem jeweils getauscht werden muss. Die Diagonale von L ist konstant Eins und wird daher nicht gespeichert.)

Beispiel I

Beispiel 6.10 (LR-Zerlegung)

Betrachte wieder das Beispiel bei der Gauß-Elimination.
Einträge von R sind rot markiert, Einträge von L blau,
Permutationsindizes grün.

Tausche erste und vierte Zeile, erste Zeile ist damit fertig.
Notiere, dass die erste Zeile mit der vierten getauscht wurde:

$$\left[\begin{array}{c|ccc} 0 & 4 & -2 & -3 \\ \hline -2 & -6 & 3 & 1 \\ 3 & 7 & 4 & -1 \\ 1 & 3 & 0 & -2 \end{array}\right] \rightsquigarrow \left[\begin{array}{c|ccc} 1 & 3 & 0 & -2 \\ \hline -2 & -6 & 3 & 1 \\ 3 & 7 & 4 & -1 \\ 0 & 4 & -2 & -3 \end{array}\right]$$

$$(0,0,0)^T \rightsquigarrow (4,0,0)^T$$

Beispiel II

Beispiel 6.10 (LR-Zerlegung)

Eliminiere die verbleibenden Einträge der ersten Spalte, trage stattdessen die Faktoren q_{i1} ein:

$$\left[\begin{array}{c|ccc} 1 & 3 & 0 & -2 \\ \hline -2 & -6 & 3 & 1 \\ 3 & 7 & 4 & -1 \\ 0 & 4 & -2 & -3 \end{array}\right] \rightsquigarrow \left[\begin{array}{c|ccc} 1 & 3 & 0 & -2 \\ \hline -2 & 0 & 3 & -3 \\ 3 & -2 & 4 & 5 \\ 0 & 4 & -2 & -3 \end{array}\right]$$

(da das Pivotelement Eins ist, bleiben hier die Einträge einfach erhalten!)

Beispiel III

Beispiel 6.10 (LR-Zerlegung)

Tausche die zweite Zeile mit der dritten:

$$\left[\begin{array}{cc|cc} 1 & 3 & 0 & -2 \\ -2 & 0 & 3 & -3 \\ \hline 3 & -2 & 4 & 5 \\ 0 & 4 & -2 & -3 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|cc} 1 & 3 & 0 & -2 \\ 3 & -2 & 4 & 5 \\ \hline -2 & 0 & 3 & -3 \\ 0 & 4 & -2 & -3 \end{array}\right]$$

$$(4,0,0)^T \rightsquigarrow (4,3,0)^T$$

(*wichtig:* dabei werden die Einträge von L mitgetauscht!)

Beispiel IV

Beispiel 6.10 (LR-Zerlegung)

Eliminiere die verbleibenden Einträge der zweiten Spalte und merke die Faktoren q_{i2}:

$$\left[\begin{array}{cc|cc} 1 & 3 & 0 & -2 \\ 3 & -2 & 4 & 5 \\ \hline -2 & 0 & 3 & -3 \\ 0 & 4 & -2 & -3 \end{array}\right] \rightsquigarrow \left[\begin{array}{cc|cc} 1 & 3 & 0 & -2 \\ 3 & -2 & 4 & 5 \\ \hline -2 & 0 & 3 & -3 \\ 0 & -2 & 6 & 7 \end{array}\right]$$

Beobachtung: jeder Tausch-Schritt ergibt eine Zeile von R, jeder Eliminationsschritt eine Spalte von L

Beispiel V

Beispiel 6.10 (LR-Zerlegung)

Im letzten Schritt ist kein Tauschen nötig. Notiere dies durch den Index der aktuellen Zeile. Eliminiere den letzen Eintrag und speichere dort stattdessen q_{43}:

$$\left[\begin{array}{ccc|c} 1 & 3 & 0 & -2 \\ 3 & -2 & 4 & 5 \\ -2 & 0 & 3 & -3 \\ \hline 0 & -2 & 6 & 7 \end{array}\right] \rightsquigarrow \left[\begin{array}{ccc|c} 1 & 3 & 0 & -2 \\ 3 & -2 & 4 & 5 \\ -2 & 0 & 3 & -3 \\ \hline 0 & -2 & 2 & 13 \end{array}\right]$$

$$(4,3,0)^T \rightsquigarrow (4,3,3)^T$$

Beispiel VI

Beispiel 6.10 (LR-Zerlegung)

Prüfe nach, dass $L \cdot R = P \cdot A$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ 0 & -2 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 0 & -2 \\ 0 & -2 & 4 & 5 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 13 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 4 & -2 & -3 \\ -2 & -6 & 3 & 1 \\ 3 & 7 & 4 & -1 \\ 1 & 3 & 0 & -2 \end{bmatrix}$$

Absolutwertnotation

Sei $A \in \mathbb{R}^{m \times n}$, dann definieren wir die *Matrix der Beträge* $B = |A|$ durch

$$b_{ij} := |a_{ij}|, \quad 1 \leq i \leq m, 1 \leq j \leq n,$$

und analog $|x|$ für $x \in \mathbb{R}^n$ (*Vektor der Beträge*).

Damit können wir die Abbildung rd auf $\mathbb{R}^{m \times n}$ bzw. $\mathbb{R}^n$ erweitern:

$$\mathrm{rd}(A) = A + A' \quad \text{mit} \quad |A'| \leq |A| \cdot \mathrm{eps}$$

(das sind $m \cdot n$ Ungleichungen), wegen

$$\mathrm{rd}(a_{ij}) = a_{ij}(1 + \epsilon_{ij}) = a_{ij} + \epsilon_{ij} a_{ij}, \quad |\epsilon_{ij} a_{ij}| \leq |a_{ij}| \cdot \mathrm{eps}$$

fl-Notation

Sei E eine Formel, dann bezeichne $\mathrm{fl}(E)$ eine Berechnung der Formel in Gleitkommaarithmetik. Die Reihenfolge wird dabei meist offensichtlich sein (von links nach rechts, Klammerung), sonst muss sie angegeben werden.

Somit ist zum Beispiel:

$$\mathrm{fl}(A+B) = (A+B) + H \quad \text{mit} \quad |H| \leq \mathrm{eps} \cdot (|A+B|)$$

wegen

$$\mathrm{fl}(a_{ij} + b_{ij}) = a_{ij} \oplus b_{ij} = (a_{ij} + b_{ij})(1+\epsilon_{ij}) = (a_{ij} + b_{ij}) + \epsilon_{ij}(a_{ij} + b_{ij})$$

Man kann z.B. auch $\mathrm{fl}(\sqrt{x})$ und $\mathrm{fl}(\sin(x))$ schreiben, letzteres setzt Angabe von Algorithmus voraus.

Rückwärtsanalyse

Bisher haben wir die "Vorwärtsanalyse" der Rundungsfehler betrieben, z.B. gilt für das Skalarprodukt $x^T y$:

$$|\,\mathrm{fl}(x^T y) - x^T y| \leq n \cdot \mathrm{eps} \cdot |x|^T |y| + \mathcal{O}(\mathrm{eps}^2) \quad \text{(absolut)}$$

oder

$$\frac{|\,\mathrm{fl}(x^T y) - x^T y|}{|x^T y|} \leq n \cdot \mathrm{eps} \cdot \frac{|x|^T |y|}{|x^T y|} + \mathcal{O}(\mathrm{eps}^2) \quad \text{(relativ)}$$

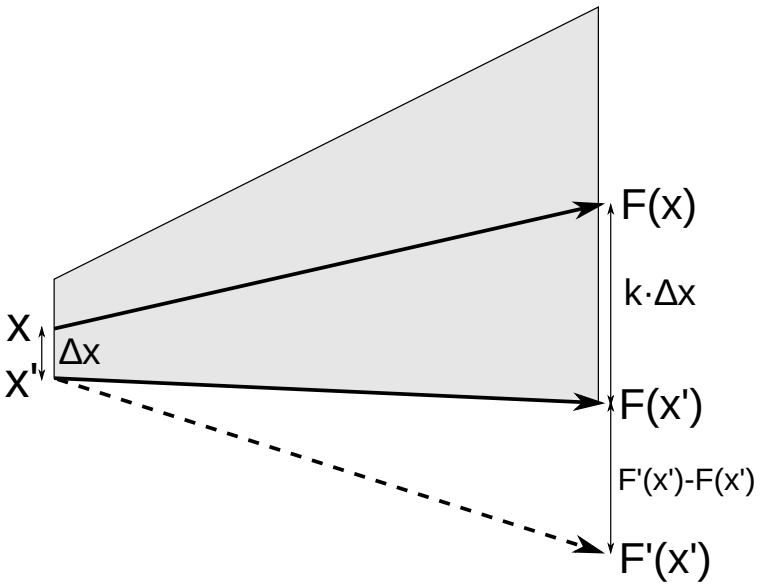
Eine Alternative ist die sog. "Rückwärtsanalyse". Dort versucht man, das Fließkommaergebnis als *exaktes Ergebnis* eines *modifizierten Ausdrucks* zu schreiben, z.B.:

$$\mathrm{fl}(x^T y) = (x + f)^T y \quad \text{mit} \quad |f| \leq n \cdot \mathrm{eps} \cdot |x| + \mathcal{O}(\mathrm{eps}^2)$$

Rückwärtsanalyse

Erinnerung: Ein Algorithmus ist *stabil*, wenn der akkumulierte Fehler der Rechnung den aufgrund der Problemstellung unvermeidbaren Fehler ($\rightarrow$ Kondition) nicht übersteigt (oder zumindest in der selben Größenordnung ist).

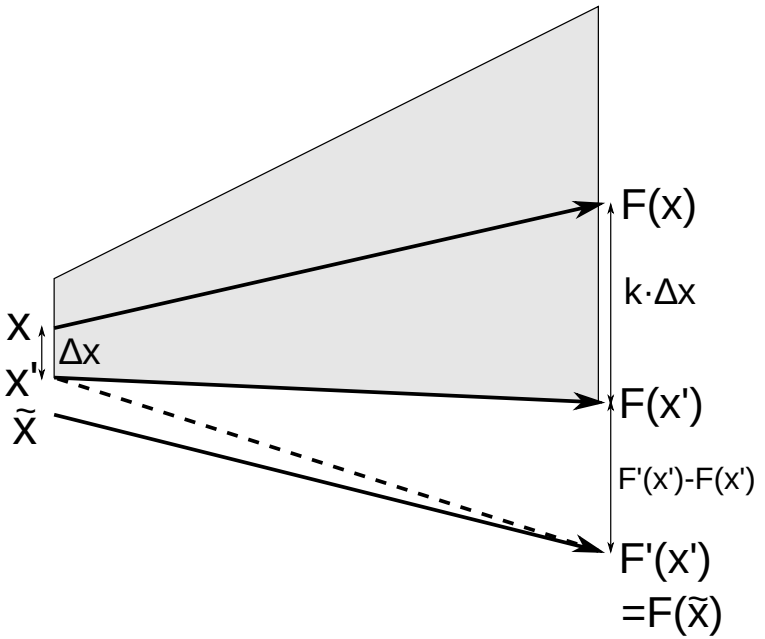
Der tatsächlich auftretende Diskretisierungsfehler hat dann die gleiche Größenordnung wie der kleinste für diese Aufgabe theoretisch mögliche Fehler.



Rückwärtsanalyse

Definition 6.11 (Rückwärtsstabilität)

Ein Algorithmus F' für eine Funktion F heißt *rückwärtsstabil*, wenn zu jeder Eingabe x eine modifizierte Eingabe $\widetilde{x}$ mit $F'(x) = F(\widetilde{x})$ existiert und $|x - \widetilde{x}| = \mathcal{O}(|x - \mathrm{rd}(x)|)$ gilt.



Interpretation: Der Algorithmus löst das gestellte Problem, aber für eine leicht gestörte Eingabe.

Man kann zeigen, dass rückwärtsstabile Algorithmen stets stabil sind.

Beispiel

Beispiel 6.12

Die Berechnungen $F'(x,y) = x \oplus y$ und $G'(x) = x \oplus 1$ sind jeweils stabile Algorithmen für die Addition bzw. das Inkrement. F' ist rückwärtsstabil:

$$x \oplus y = (x+y)\cdot(1+\epsilon) = x\cdot(1+\epsilon) + y\cdot(1+\epsilon) = \widetilde{x} + \widetilde{y}$$
$$\text{mit} \quad \widetilde{x} := x\cdot(1+\epsilon), \quad |x-\widetilde{x}| = |x|\cdot\epsilon \leq |x|\cdot\text{eps}, \quad \widetilde{y} \text{ analog}$$

Aber G' ist nicht rückwärtsstabil:

$$x \oplus 1 = (x+1)\cdot(1+\epsilon) = x\cdot(1+(1+x^{-1})\cdot\epsilon) + 1 = \widetilde{x} + 1$$
$$\text{mit} \quad \widetilde{x} := x\cdot(1+(1+x^{-1})\cdot\epsilon), \quad |x-\widetilde{x}| = |x|\cdot|1+x^{-1}|\cdot\epsilon$$

Für $x \ll 1$ wird der relative Fehler beliebig groß (vgl. $\epsilon \oplus 1 = 1$).

Skalarprodukt

Hilfssatz 6.13

Das Skalarprodukt $x^T y \in \mathbb{R}, x, y \in \mathbb{F}^n$, ist rückwärtsstabil:

$$\hat{s} = \mathrm{fl}(x^T y) = (x + f)^T y \quad \text{mit} \quad |f| \leq n \cdot \mathrm{eps} \cdot |x| + \mathcal{O}(\mathrm{eps}^2)$$

Bemerkung 6.14

Der Faktor $n \cdot \mathrm{eps}$ ist der schlechteste Fall und sehr pessimistisch. Rundungsfehler von Operationen hängen von den Argumenten ab und können sich auch wegheben (Vorzeichenwechsel!). Eine statistische Betrachtung würde auf einen deutlich geringeren *durchschnittlichen* Fehler führen.

$\rightsquigarrow$ Beweis: Tafel

Dyadisches Produkt

Beispiel 6.15

Das *dyadische Produkt* (äußere Produkt) $x \cdot y^T \in \mathbb{R}^{n \times n}$ hingegen ist zwar stabil, aber nicht rückwärtsstabil. *Warum?*

Ein dyadisches Produkt hat immer Rang Eins, $\mathrm{fl}(x \cdot y^T)$ wird jedoch durch die Störungen i.A. einen anderen Rang haben (und kann somit nicht das Ergebnis einer exakten Rechnung sein).

Lösen von Dreieckssystemen

Satz 6.16 (Lösen von Dreieckssystemen)

Es seien $\hat{x}$ und $\hat{y}$ die numerischen Lösungen des unteren bzw. oberen Dreieckssystems $Lx = b$ und $Ry = c$. Dann gilt

$$(L + F)\hat{x} = b \qquad |F| \leq n \cdot \text{eps} \cdot |L| + \mathcal{O}(\text{eps}^2)$$
$$(R + G)\hat{y} = c \qquad |G| \leq n \cdot \text{eps} \cdot |R| + \mathcal{O}(\text{eps}^2)$$

Das Lösen von Dreiecksystemen durch Vorwärts- / Rückwärtseinsetzen ist also rückwärtsstabil.

$\rightsquigarrow$ Beweis: Übung

LR-Zerlegung

Satz 6.17 (Rückwärtsanalyse der LR-Zerlegung)

Sei $A \in \mathbb{F}^{n\times n}$. Es werde die LR-Zerlegung von A *ohne Zeilentausch* berechnet (sofern möglich). Dann gilt für die numerisch berechneten Matrizen $\widehat{L}$ und $\widehat{R}$:

$$\widehat{L}\cdot\widehat{R} = A + H \quad \text{mit} \quad |H| \leq 3(n-1)\cdot \text{eps}\cdot\left(|A| + |\widehat{L}|\cdot|\widehat{R}|\right) + \mathcal{O}(\text{eps}^2)$$

$\rightsquigarrow$ Beweis: Tafel

LR-Zerlegung

Satz 6.18 (Lösen mittels LR-Zerlegung)

Seien $\widehat{L}$ und $\widehat{R}$ die numerisch berechnete LR-Zerlegung von $A \in \mathbb{F}^{n\times n}$ aus Satz 6.17. Sei weiter $\hat{y} \in \mathbb{F}^n$ die numerische Lösung von $\widehat{L}y = b$, und schließlich $\hat{x} \in \mathbb{F}^n$ die numerische Lösung von $\widehat{R}x = \hat{y}$. Dann gilt für $\hat{x}$ die Beziehung

$$(A + E)\hat{x} = b$$

mit

$$|E| \leq n \cdot \text{eps} \cdot (3|A| + 5|\widehat{L}| \cdot |\widehat{R}|) + \mathcal{O}(\text{eps}^2)$$

$\rightsquigarrow$ Beweis: Tafel

Folgerung

Nach Satz 6.18 ist $\hat{x}$ *exakte* Lösung des *modifizierten* Systems $(A+E)\hat{x} = b$. Mit dem Störungssatz gilt dann in $\|\cdot\|_\infty$-Norm

$$\frac{\|\hat{x}-x\|_\infty}{\|x\|_\infty} \leq \operatorname{cond}_\infty(A)\cdot\left\{3n\cdot\operatorname{eps} + 5n\cdot\operatorname{eps}\frac{\|\widehat{L}\|_\infty\cdot\|\widehat{R}\|_\infty}{\|A\|_\infty} + \mathcal{O}(\operatorname{eps}^2)\right\}$$

- Der erste Term ist mit dem aus der Konditionsanalyse vergleichbar und daher gutartig.
- Der zweite Term kann problematisch werden, da $\widehat{L}$ Einträge der Form $\widetilde{a}_{ij}/\widetilde{a}_{ii}$ enthält (und $\widetilde{a}_{ii}$ beliebig klein werden kann). Man kann zeigen, dass $\widehat{R}$ genauso problematisch sein kann.

$\Longrightarrow$ Die Gauß-Elimination (LR-Zerlegung) ist in dieser Form *nicht* numerisch stabil!

Beispiel

Beispiel 6.19

Betrachte die Matrix A gegeben durch

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & -\epsilon \end{bmatrix}$$

mit $0 < \epsilon \ll 1$. $\text{cond}_\infty(A) = (1+\epsilon)^2$, also ist A gut konditioniert, und Rückwärtseinsetzen bietet sich als stabiler Algorithmus an. Die (exakte) LR-Zerlegung ist aber in diesem Fall $A = LR$ mit

$$L = \begin{bmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{bmatrix}, \quad R = \begin{bmatrix} \epsilon & 1 \\ 0 & -\epsilon^{-1} \end{bmatrix}$$

$$\Longrightarrow \|\widehat{L}\|_\infty \approx \|L\|_\infty = \epsilon^{-1} + 1 \gg 1, \quad \|\widehat{R}\|_\infty \approx \|R\|_\infty = \epsilon^{-1} \gg 1$$

Pivotisierung

Die Rundungsfehleranalyse in Satz 6.18 führt auf den unvorteilhaften Term $|\widehat{L}| \cdot |\widehat{R}|$, und wie wir gesehen haben, können diese beiden Faktoren prinzipiell beliebig groß werden.

Wählt man im Gauß-Algorithmus (bzw. bei der LR-Zerlegung) den Index r so, dass

$$|a_{rk}^{(k)}| \geq |a_{ik}^{(k)}| \quad \forall\, k \leq i \leq n,$$

dann gilt

$$|\hat{l}_{ij}| \leq 1 \quad \text{und damit} \quad \|\widehat{L}\|_\infty \leq n$$

Diese Wahl des Pivotelements nennt man *Spaltenpivotisierung*.

Beispiel I

Beispiel 6.20

Das Gauß-Verfahren für das lineare Gleichungssystem

$$\begin{bmatrix} -10^{-5} & 1 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

führt in exakter Arithmetik auf

$$\begin{bmatrix} -10^{-5} & 1 \\ 0 & 1 + 2 \cdot 10^5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \cdot 10^5 \end{bmatrix}$$

mit der Lösung $(x_1, x_2)^T = (-0.4999975, 0.999995)^T \approx (-0.5, 1)^T$.

Beispiel II

Beispiel 6.20

In $\mathbb{F}(10, 4, 1)$ ergibt sich zwar ebenfalls ein Faktor $q_{21} = 2 \cdot 10^5 = 0.2 \cdot 10^6$, das dadurch entstehende System ist aber

$$\begin{bmatrix} -0.1 \cdot 10^{-4} & 1 \\ 0 & 0.2 \cdot 10^6 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.2 \cdot 10^6 \end{bmatrix},$$

der Eintrag $a_{22}^{(1)}$ ist also (als einziger!) leicht fehlerbehaftet.

Als Ergebnis erhält man das völlig inakzeptable $(x_1, x_2)^T = (0, 1)^T$.

$\rightsquigarrow$ Rechnung: Tafel

Beispiel III

Beispiel 6.20

Die berechnete Lösung hat also einen relativen Fehler von 50% in der Maximumsnorm, obwohl die Matrix sehr gut konditioniert ist.

Rückwärtsanalyse:
Tatsächlich wurde das modifizierte System

$$\begin{bmatrix} -10^{-5} & 1 \\ 2 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

exakt gelöst, welches aber eine völlig andere Lösung hat.

Beispiel IV

Beispiel 6.20

Spaltenpivotisierung (in $\mathbb{F}(10, 4, 1)$) führt in diesem Fall auf das System

$$\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

mit der vollkommen akzeptablen Antwort $(x_1, x_2)^T = (-0.5, 1)^T$.

Beachte aber:
Multiplikation der ersten Zeile des ursprünglichen Systems mit -10^6 führt auf ein mathematisch äquivalentes System, das auch bei Spaltenpivotisierung keine Vertauschungen braucht und daher trotzdem das ursprüngliche, unbrauchbare, Ergebnis liefert.

$\rightsquigarrow$ Rechnung: Tafel

Äquilibrierung

Solche Probleme kann man oft durch eine *Skalierung* (Äquilibrierung) des Gleichungssystems vermindern:

$$Ax = b \iff D^{-1}Ax = D^{-1}b \quad \text{mit} \quad d_{ii} = \sum_{j=1}^{n} |a_{ij}|$$

$$\iff \widetilde{A}x = \widetilde{b} \quad \text{mit} \quad \|\widetilde{A}\|_\infty = 1$$

(*Vorsicht:* keine Aussage über $\|\widetilde{A}^{-1}\|_\infty$, es kann durchaus auch $\text{cond}_\infty(\widetilde{A}) > \text{cond}_\infty(A)$ sein!)

Spezialfall der allgemeineren Idee der *Vorkonditionierung*:

Transformiere die ursprüngliche Aufgabenstellung in eine neue Formulierung, die besser konditioniert ist und sich stabil lösen lässt, und transformiere danach die Ergebnisse zurück (falls nötig)

Beispiel

Beispiel 6.21

Betrachte die folgende vom Parameter $c \neq 0$ abhängige Matrix A_c:

$$A_c = \begin{bmatrix} c & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 2c & -c \\ -1 & 2 \end{bmatrix}, \quad A_c^{-1} = \frac{1}{3} \cdot \begin{bmatrix} 2c^{-1} & 1 \\ c^{-1} & 2 \end{bmatrix}$$

Es gilt $\operatorname{cond}_\infty(A_c) = 1 + 2 \cdot \max\{|c|, |c|^{-1}\}$.

Optimal ist also die Wahl $c = 1$ mit $\operatorname{cond}_\infty(A_1) = 3$. Für $c \neq 1$ wird eine der Zeilen mehr betont als die andere, was im Extremfall zu einem sehr unausgeglichenen Gleichungssystem führt.

$\rightsquigarrow$ Rechnung: Tafel

Rundungsfehleranalyse bei Pivotisierung

Analog zu Satz 6.18 zeigt man, dass für die Lösung $\hat{x}$ bei Spaltenpivotisierung gilt:

$$(A + E)\hat{x} = b$$

mit

$$|E| \leq n \cdot \text{eps} \cdot (3|A| + 5P^T \cdot |\widehat{L}| \cdot |\widehat{R}|) + \mathcal{O}(\text{eps}^2)$$

Nach Konstruktion ist $\|\widehat{L}\|_\infty \leq n$, und mit der Definition

$$\rho := \|A\|_\infty^{-1} \cdot \max_{i,j,k} |\hat{a}_{ij}^{(k)}|$$

(sog. "Wachstumsfaktor") zeigt man

$$\|E\|_\infty \leq 8n^3 \|A\|_\infty \rho \cdot \text{eps} + \mathcal{O}(\text{eps}^2)$$

$\rightsquigarrow$ Beweis: [Golub, Van Loan: Matrix Computations]

Rundungsfehleranalyse bei Pivotisierung

Man kann Beispiele mit sehr schlechtem Verhalten für ρ konstruieren:
Betrachte $A \in \mathbb{R}^{n \times n}$ mit

$$a_{ij} := \begin{cases} 1 & j = i \vee j = n \\ -1 & j < i \\ 0 & \text{sonst} \end{cases}$$

Beispiel:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

Für diesen Extremfall ist $\rho = 2^{n-1}$.

In der Praxis wachsen die Einträge aber normalerweise nur um bis zu eine Größenordnung ($\rho \approx 10$, [Golub, Van Loan]), daher können Gauss-Verfahren und LR-Zerlegung *mit Spaltenpivotisierung* als stabil angesehen werden.

Totale Pivotisierung

Man kann auch $r, s \in \{k, \ldots, n\}$ wählen, so dass

$$|a_{rs}^{(k)}| \geq |a_{ij}^{(k)}| \quad \forall\, k \leq i, j \leq n$$

und erreicht dann durch *Zeilen- und Spaltenvertauschung*, dass

$$\widetilde{a}_{kk}^{(k)} = a_{rs}^{(k)}$$

Diese Wahl des Pivotelements nennt man *totale Pivotisierung*.

Man kann zeigen, dass damit

$$|a_{ij}^{(k)}| \leq k^{1/2}\left(2 \cdot 3^{1/2} \cdots k^{1/(k-1)}\right)^{1/2} \cdot \max |a_{ij}|,$$

also deutlich kleineres Wachstum der Einträge.

Totale Pivotisierung

In Matrixform ergibt sich (mit $P_{r_k} := P_{k,r_k}$ und $P_{s_k} := P_{k,s_k}$)

Schritt 1:
$$G_1 P_{r_1} A P_{s_1} \cdot P_{s_1} x = G_1 P_{r_1} b$$

Schritt 2:
$$G_2 P_{r_2} G_1 P_{r_1} A P_{s_1} P_{s_2} \cdot P_{s_2} P_{s_1} x = G_2 P_{r_2} G_1 P_{r_1} b$$

$\vdots$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\vdots$

Schritt $n-1$:
$$\underbrace{\left[\prod_{k=1}^{n-1}(G_k P_{r_k}) A \prod_{k=n-1}^{1} P_{s_k}\right]}_{=R \text{ (obere Dreiecksmatrix)}} \cdot \prod_{k=1}^{n-1} P_{s_k} x = \prod_{k=1}^{n-1}(G_k P_{r_k}) b$$

Analog zur Spaltenpivotisierung zeigt man, dass dann

$$PAQ^T \cdot Qx = Pb$$

mit der LR-Zerlegung $PAQ^T = LR$ (beachte $Q^T \cdot Q = I$).

Totale Pivotisierung

Setze dazu

$$P := \prod_{k=1}^{n-1} P_{r_k}, \quad Q := \prod_{k=1}^{n-1} P_{s_k},$$

$$L := P \cdot \left(\prod_{k=1}^{n-1} (G_k P_{r_k}) \right)^{-1}, \quad R := \prod_{k=1}^{n-1} (G_k P_{r_k}) \cdot A \cdot Q^T$$

R ist obere Dreiecksmatrix nach Konstruktion, und wie zuvor ist L untere Dreiecksmatrix (selbe Argumentation, da unabhängig von Q).

Außerdem kann L wieder unten links in A gespeichert werden (Spaltenvertauschungen ändern die bisher berechneten Einträge von L nicht, da $s_k \geq k$).

Lösen bei totaler Pivotisierung

Mit diesen Definitionen haben wir

$$PAQ^T \cdot Qx = LR \cdot Qx = Pb$$

Löse mit den folgenden Schritten nach x auf:

$$\begin{aligned} b' &= Pb && \text{(Permutation)} \\ Ly &= b' && \text{(Dreieckssystem)} \\ Rx' &= y && \text{(Dreieckssystem)} \\ x &= Q^T x' && \text{(Permutation)} \end{aligned}$$

Der letzte Schritt nutzt dabei, dass Q orthogonal ist ($Q^{-1} = Q^T$), um Q nicht explizit invertieren zu müssen.

Anmerkungen

Aufwand der Pivotisierung:

- $n^2/2$ Vergleiche bei Spaltenpivotisierung
- $n^3/3$ Vergleiche bei totaler Pivotisierung

$\Longrightarrow$ totale Pivotisierung hat gleiche Komplexität wie eigentliche Zerlegung, während Spaltenpivotisierung vernachlässigbar ist

Praktische Erfahrung zeigt trotz höheren Aufwands keine Vorteile für Rundungsfehler bei totaler Pivotisierung

Spaltenpivotisierung mit Zeilenskalierung ist *in der Praxis* effizient und numerisch stabil.

Symmetrisch positiv definite Matrizen

Satz 6.22

Eine symmetrisch positiv definite Matrix $A \in \mathbb{R}^{n \times n}$ ist stets *ohne* Pivotisierung stabil LR-zerlegbar. Für die Diagonalelemente der im Eliminationsprozess auftauchenden Matrizen gilt

$$a_{ii}^{(k)} \geq \lambda_{\min}(A), \quad k \leq i \leq n.$$

Die Symmetrie der Matrix kann man ausnutzen, um die Matrix A mit geringerem Aufwand zu zerlegen.

$\rightsquigarrow$ Beweis: Tafel, Stabilität: [Golub, Van Loan]

Cholesky-Zerlegung

Mit $D = \operatorname{diag}(R)$ gilt

$$A = LD(D^{-1}R) = LDU \quad \text{mit} \quad U := D^{-1}R,$$

und wegen der Symmetrie gilt $U = L^T$, also $A = LDL^T$. Da alle Diagonalelemente von D positiv sind, ist die Matrix $D^{1/2}$ mit

$$(D^{1/2})_{ii} = d_{ii}^{1/2}, \quad (D^{1/2})_{ij} = 0 \text{ für } i \neq j$$

wohldefiniert, und es gilt

$$A = LD^{1/2} \cdot D^{1/2}L^T = \widetilde{L}\widetilde{L}^T \quad \text{mit} \quad \widetilde{L} := LD^{1/2}$$

Diese spezielle Form der Zerlegung heißt *Cholesky-Zerlegung*.

Cholesky-Zerlegung

Ein einzelner Schritt der Cholesky-Zerlegung lässt sich über das äußere Produkt darstellen:

$$A = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix} = \begin{bmatrix} \beta & 0 \\ \beta^{-1} \cdot v & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & B - \alpha^{-1} \cdot vv^T \end{bmatrix} \cdot \begin{bmatrix} \beta & \beta^{-1} \cdot v^T \\ 0 & I_{n-1} \end{bmatrix}$$

mit $\beta := \alpha^{1/2}$. Dieser Prozess kann dann mit $B - \alpha^{-1} \cdot vv^T$ rekursiv fortgesetzt werden. Die dabei auftretenden Matrizen sind verallgemeinerte Frobenius-Matrizen (β als Diagonaleintrag).

Aufgrund der Symmetrie ist dabei der Aufwand halb so groß wie bei der allgemeinen LR-Zerlegung:

$$N_{\text{Chol}}(n) = \frac{1}{3}n^3 + \mathcal{O}(n^2)$$

Diagonaldominante Matrizen

Definition 6.23 (Diagonaldominanz)

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt *diagonaldominant*, falls

$$\sum_{j \neq i} |a_{ij}| \leq |a_{ii}| \quad i = 1, \ldots, n$$

und *strikt diagonaldominant*, falls die Summe echt kleiner als $|a_{ii}|$ ist.
(vgl. mit Kriterium der Gerschgorin-Kreise, was folgt dann für A?)

Satz 6.24

Diagonaldominante reguläre Matrizen erlauben eine LR-Zerlegung ohne Pivotisierung.

$\rightsquigarrow$ Beweis: Tafel

Rangbestimmung und Determinante

Für eine allgemeine, nicht unbedingt reguläre Matrix $A \in \mathbb{R}^{n \times n}$ lässt sich die LR-Zerlegung ebenfalls durchführen, sie bricht aber evtl. mangels Pivotelement frühzeitig ab.

- Wenn dies bei totaler Pivotisierung nach k Schritten passiert, dann hat A Rang k.
- Da die Einträge aufgrund von Fehlern nie exakt Null werden, muss mit einer Toleranz gearbeitet werden.

Falls A Vollrang haben sollte, kann man wegen

$$\det(A) = \det(P^T LR) = \det(P^T) \cdot \det(L) \cdot \det(R) = \pm \det(R)$$

über die LR-Zerlegung die Determinante von A berechnen (das Vorzeichen hängt davon ab, ob P eine gerade oder ungerade Permutation repräsentiert).

Inversenberechnung

Die Inverse einer Matrix $A \in \mathbb{R}^{n \times n}$ kann wie folgt berechnet werden:

- Berechne LR-Zerlegung (Aufwand $\frac{2}{3}n^3 + \mathcal{O}(n^2)$)
- Löse $Ax_i = e_i$, $i = 1, \ldots, n$, für die kanonische Basis e_i (Aufwand $2n \cdot n^2$: n Gleichungen, jeweils 2 Dreieckssysteme)
- Forme Matrix der Urbilder: $A^{-1} = [x_1, \ldots, x_n]$ (spaltenweise)

Der Gesamtaufwand einer solchen Matrixinversion ist daher

$$N_{\text{Inv}}(n) = \frac{8}{3}n^3 + \mathcal{O}(n^2)$$

Tridiagonalsysteme

Definition 6.25 (Tridiagonalmatrix)

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt *Tridiagonalmatrix*, falls

$$a_{ij} = 0 \quad \text{für} \quad |i - j| > 1,\ 1 \leq i \leq n$$

Eine Tridiagonalmatrix ist Spezialfall einer *Bandmatrix* mit

$$a_{ij} = 0 \quad \text{für} \quad j < i - m_l \text{ oder } j > i + m_r,\ 1 \leq i \leq n$$

Falls sich die LR-Zerlegung einer Tridiagonal- bzw. Bandmatrix ohne Pivotisierung durchführen lässt, sind L und R ebenfalls tridiagonal bzw. Bandmatrizen der selben Bandbreite (m_l, m_r).

$\rightsquigarrow$ Rechnung für Tridiagonalmatrizen: Tafel

Nichtreguläre Systeme

Sei nun $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, mit $\text{rang}(A)$ beliebig. Dann hat $Ax = b$ genau eine Lösung, gar keine Lösung, oder unendlich viele.

Einige Grundbegriffe aus der linearen Algebra:

$$\text{im}(A) = \{y \in \mathbb{R}^m \mid \exists\, x \in \mathbb{R}^n \colon y = Ax\}$$
$$\ker(A) = \{x \in \mathbb{R}^n \mid Ax = 0\}$$
$$\text{rang}(A) = \dim(\text{im}(A)) = \text{rang}(A^T) = \dim(\text{im}(A^T))$$

Nichtreguläre Systeme

Orthogonales Komplement:

$$\operatorname{im}(A)^{\perp} = \{y \in \mathbb{R}^m \mid \forall\, y' \in \operatorname{im}(A) \colon (y, y')_2 = 0\}$$

Es gilt

$$\operatorname{im}(A)^{\perp} = \ker(A^T)$$

und daher

$$\dim(\operatorname{im}(A)) + \dim(\ker(A^T)) = m$$
$$\dim(\operatorname{im}(A^T)) + \dim(\ker(A)) = n$$

Damit lässt sich der Lösungsbegriff für lineare Gleichungssysteme auf nichtreguläre Systeme erweitern.

$\rightsquigarrow$ Rechnung: Tafel

Least Squares

Satz 6.26 (Least Squares)

Für allgemeines $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, existiert ein $\bar{x} \in \mathbb{R}^n$, so dass

$$\|A\bar{x} - b\|_2 = \min_{x \in \mathbb{R}^n} \|Ax - b\|_2.$$

Diese Bedingung ist äquivalent dazu, dass $\bar{x}$ Lösung der folgenden "Normalengleichung" ist:

$$A^T A\bar{x} = A^T b, \quad A^T A \in \mathbb{R}^{n \times n}$$

Falls $\operatorname{rang}(A) = n$ (und damit zwingend $m \geq n$), ist $\bar{x}$ eindeutig, sonst hat jede weitere Lösung die Form $\bar{x} + y$ mit $y \in \ker(A)$.

$\rightsquigarrow$ Beweis: Tafel

Least Squares

$A^T A$ ist symmetrisch positiv semidefinit, also könnte man Least Squares prinzipiell über die Cholesky-Zerlegung lösen (mit symmetrischem Pivoting im nicht-definiten Fall).

Aber dieses Problem ist i.A. sehr schlecht konditioniert:

$$\text{cond}(A^T A) \approx \text{cond}(A)^2 \quad \text{für} \quad m = n$$

Daher verwendet man üblicherweise spezielle Zerlegungen, die ähnlich zur Cholesky-Zerlegung

$$A^T A = LL^T = R^T R$$

zur Verfügung stellen, aber ohne $A^T A$ explizit aufzustellen.

Householdermatrizen

Definition 6.27 (Householderreflektion)

Sei $v \neq 0 \in \mathbb{R}^n$ gegeben, dann heißt die Matrix

$$Q_v = I - 2\frac{vv^T}{v^T v}$$

Householdermatrix oder *Householderreflektion*.

vv^T ist das äußere Produkt, also eine Matrix mit Rang 1, und $v^T v$ das innere Produkt, also eine Zahl.

Householdermatrizen

Hilfssatz 6.28

Eine Householdermatrix Q_v hat die folgenden Eigenschaften:

- Q_v ist orthogonal und symmetrisch, d.h. $Q_v^T = Q_v^{-1} = Q_v$
- Es gilt für $x \in \mathbb{R}^n$:

$$Q_v x = x - 2\beta v, \quad \beta = \frac{v^T x}{v^T v} \in \mathbb{R}$$

- Speziell gilt $x = \alpha v \implies Q_v x = -x$
- $(x, v)_2 = 0 \implies Q_v x = x$
- Als Operator aufgefasst bewirkt Q_v eine Reflektion an der Ebene mit Normale v

$\rightsquigarrow$ Beweis: Tafel

Householdermatrizen

Sei $x \neq 0 \in \mathbb{R}^n$, und sei $v := x \pm \|x\|_2 e_1$, dann gilt

$$Q_v x = \mp 2 \|x\|_2 e_1$$

Wenn x die erste Spalte einer Matrix A ist, dann eliminiert also eine Multiplikation von A mit Q_v alle Einträge $a_{i1}, i > 1$.

Das heißt, dass Householdermatrizen Q_v genauso wie Frobeniusmatrizen G_k benutzt werden können, um eine Matrix auf Dreiecksgestalt zu bekommen. Für die Rekursion beschränkt man das obige Vorgehen auf die entsprechenden Unterräume.

$\rightsquigarrow$ Rechnung: Tafel

QR-Zerlegung

Satz 6.29 (QR-Zerlegung)

Zu jeder Matrix $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ und $\operatorname{rang}(A) = n$ existiert eine orthogonale Matrix $Q \in \mathbb{R}^{m \times m}$ und eine obere Dreiecksmatrix $R \in \mathbb{R}^{m \times n}$, so dass

$$A = QR.$$

Diese Zerlegung kann durch $n - 1$ Multiplikationen mit Householdermatrizen berechnet werden. Die ersten n Spalten von Q bilden eine orthonormale Basis von $\operatorname{im}(A)$.

$\rightsquigarrow$ Beweis: Tafel

Anmerkungen

Bemerkung 6.30

Anmerkungen zur QR-Zerlegung:

- Um Auslöschung zu vermeiden, wird das Vorzeichen in v meist so gewählt, dass sich das Vorzeichen des Diagonaleintrags nicht ändert. Falls dieser Eintrag Null ist, kann beliebig eines der beiden Vorzeichen gewählt werden.
- Analog zu P bei der LR-Zerlegung wird Q *nicht* explizit aufgestellt, sondern man speichert die einzelnen Normalenvektoren v_i in der linken unteren Hälfte von A (so wie zuvor die Einträge von L).
- Die QR-Zerlegung benötigt den doppelten Aufwand der LR-Zerlegung:

$$N_{QR} = \frac{4}{3}n^3 + \mathcal{O}(n^2)$$

Lösen nichtregulärer Systeme I

1. Fall $m > n$ (überbestimmt) und $\operatorname{rang}(A) = n$:

   Nach Satz 6.29 existiert QR-Zerlegung $A = QR$.

   $$A = QR \implies A^T A = (QR)^T QR = R^T Q^T QR = R^T R$$

   also

   $$A^T A\bar{x} = A^T b \iff R^T R\bar{x} = R^T Q^T b$$

   Seien $\widetilde{R}$ und $\widetilde{Q}$ die ersten n Zeilen von R, $R = \begin{bmatrix} \widetilde{R} \\ 0 \end{bmatrix}$, bzw. Spalten von Q, $Q = [\widetilde{Q} Q^*]$, dann gilt (Rückwärtseinsetzen)

   $$\widetilde{R}\bar{x} = \widetilde{Q}^T b \iff A^T A\bar{x} = A^T b$$

   $\rightsquigarrow$ Rechnung: Tafel

Lösen nichtregulärer Systeme II

2. Fall $m < n$ (unterbestimmt) und $\operatorname{rang}(A) = m$:

Nach Satz 6.29 existiert QR-Zerlegung $A^T = QR$.

$$A^T = QR \implies A^T A = QR(QR)^T = QRR^T Q^T$$

also

$$A^T Ax = A^T b \iff QRR^T Q^T x = QRb \iff RR^T Q^T x = Rb$$

Seien $\widetilde{R}$ und $\widetilde{Q}$ die ersten m Zeilen von R bzw. Spalten von Q, dann gilt (Rückwärtseinsetzen)

$$\widetilde{R}^T \widetilde{Q}^T \bar{x} = b \implies A^T A\bar{x} = A^T b$$

Die so bestimmte Lösung $\bar{x}$ ist jene mit minimaler Norm $\|\bar{x}\|_2$.

Lösen nichtregulärer Systeme III

3. Fall $m = n$ (quadratisch) und $\text{rang}(A) = n$:

   Nach Satz 6.29 existiert QR-Zerlegung $A = QR$. Da A Vollrang hat, gilt

   $$A^T A\bar{x} = A^T b \iff A\bar{x} = b \iff QRx = b$$

   Die Least-Squares-Lösung ist also die normale Lösung des Systems, und kann über die QR-Zerlegung berechnet werden.

   Überbestimmte, quadratische und unterbestimmte Systeme mit maximalem Rang haben also immer eine eindeutig bestimmte Least-Squares-Lösung $\bar{x}$, für die $\|\bar{x}\|_2$ minimal ist (in den ersten beiden Fällen, weil es nur eine solche Lösung gibt). Diese lässt sich jeweils mittels der QR-Zerlegung berechnen.

Lösen nichtregulärer Systeme IV

4. Fall $\operatorname{rang}(A) < \min\{m, n\}$ (rangdefizitär):

   Eine solche Lösung mit minimaler Norm gibt es immer, auch wenn der Rang von A nicht maximal ist, allerdings hängt sie nicht mehr stetig von der Problemstellung ab.

   Kleine Störungen in A oder b führen dann zu beliebig großen Änderungen in $\bar{x}$ (da die gestörte Matrix praktisch garantiert regulär ist). So etwas nennt man ein *schlecht gestelltes Problem.*

Moore-Penrose-Inverse

Das QR-Verfahren berechnet jeweils die Lösung der Gleichung $\bar{x} = A^+ b$ mit der *Moore-Penrose-Inversen* A^+ von A, gegeben durch

$$A^+ = \begin{cases} (A^T A)^{-1} A^T & \text{falls } m > n \text{ und } \operatorname{rang}(A) = n \\ A^T (A A^T)^{-1} & \text{falls } m < n \text{ und } \operatorname{rang}(A) = m \\ A^{-1} & \text{falls } m = n \text{ und } \operatorname{rang}(A) = n \end{cases}$$

Für die verbleibenden Fälle (rangdefizitär) ist die Pseudoinverse A^+ zwar ebenfalls eindeutig, es gibt aber keine einfache geschlossene Formel, und sie hängt nicht stetig von Änderungen in A ab.

Anwendung: Ausgleichsrechnung

Problemstellung der *Gaußschen Ausgleichsrechnung*:

Gegeben:

1. n Funktionen $u_1, \ldots, u_n \colon \mathbb{R} \to \mathbb{R}$
2. $m \geq n$ Datenpunkte $(x_i, y_i) \in \mathbb{R}^2$, $1 \leq i \leq m$

Gesucht: n Koeffizienten $c_1, \ldots, c_n$, so dass

$$y = u(x) = \sum_{j=1}^{n} c_j \cdot u_j(x) \quad \text{und} \quad \sum_{i=1}^{m} \left(u(x_i) - y_i\right)^2 \text{ minimal}$$

"Modell" für die Abhängigkeit der Größe y vom Parameter x, mit $\{u_1, \ldots, u_n\}$ als Ansatzraum.

Anwendung: Ausgleichsrechnung

Formuliere im Rahmen der Linearen Algebra:

$$c = (c_1, \ldots, c_n)^T, \quad y = (y_1, \ldots, y_m)^T, \quad A \text{ mit } a_{ij} = u_j(x_i)$$

Dann gilt

$$\sum_{i=1}^{m} (u(x_i) - y_i)^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_j \cdot u_j(x_i) - y_i \right)^2 = \|Ac - y\|_2^2$$

Optimale Lösung ist also durch Normalengleichung gegeben:

$$u(x) := \sum_{j=1}^{n} \bar{c}_j \cdot u_j(x) \quad \text{mit} \quad A^T A \bar{c} = A^T y$$

Singulärwertzerlegung (SVD)

Satz 6.31 (Singulärwertzerlegung)

Sei $A \in \mathbb{R}^{m \times n}$, dann gibt es orthogonale Matrizen $U \in \mathbb{R}^{m \times m}$ und $V \in \mathbb{R}^{n \times n}$ und eine Diagonalmatrix

$$D = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$$

mit $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ und

$$U^T A V = D \quad (\text{und daher } A = U D V^T)$$

Die Einträge von D heißen *Singulärwerte*, sie sind bis auf die Reihenfolge eindeutig. Die Spalten von U und V heißen *(linke und rechte) Singulärvektoren*.

$\rightsquigarrow$ Beweis: Tafel

Anmerkungen

Anmerkungen zur Singulärwertzerlegung (SVD):

- Die SVD kann als Verallgemeinerung der Diagonalisierung von symmetrisch positiv semidefiniten Matrizen betrachtet werden.
- Die SVD zerlegt eine beliebige Matrix A in eine sehr einfache und elegante Kette von drei Operationen:

  Drehung / Spiegelung — Streckung — Drehung / Spiegelung

- Trotz ihrer praktischen Relevanz kann auf die SVD hier nicht weiter eingegangen werden, da wir dazu Methoden zur numerischen Bestimmung von Eigenwerten benötigen würden.

Vergleich der Verfahren

Alle vorgestellten Verfahren lassen sich zur Lösung regulärer Systeme $Ax = b$ verwenden. Welches sollte man verwenden?

Hier eine Gegenüberstellung:

Methode	Einschränkung	Stabilität	Aufwand $T(n)$
LR-Zerlegung	A quadratisch	mit Pivot.	$2/3 \cdot n^3$
Cholesky	A s.p.d.	stabil	$1/3 \cdot n^3$
QR-Zerlegung	—	sehr stabil	$4/3 \cdot n^3$
SVD	—	sehr stabil	$12 \cdot n^3$

LR- und Cholesky-Zerlegung sind am effizientesten, aber die QR-Zerlegung kann sich aus Gründen der Stabilität anbieten. Die SVD wird bei speziellen Anwendungen mit hohem Anspruch an Präzision verwendet, wo der hohe Aufwand nötig ist.

Abschnitt 7

7 Interpolation und Approximation
- Einführung
- Polynominterpolation
- Anwendungen der Polynominterpolation
- Bernstein-Polynome und Kurvendarstellung
- Splines
- Trigonometrische Interpolation
- Approximation von Funktionen

Einführung

Ziel: Darstellung und Auswertung von Funktionen im Rechner.

Anwendungen:

- Rekonstruktion eines funktionalen Zusammenhangs aus "gemessenen Funktionswerten", Auswertung an Zwischenstellen
- Teuer auszuwertende Funktionen effizienter auswerten
- Darstellung von Fonts (2D), Körpern (3D) im Rechner
- Lösung von Differential- und Integralgleichungen
- Datenkompression

Einführung

Wir beschränken uns auf Funktionen in *einer* Variablen, z.B.

$$f \in C^r[a, b]$$

Dies ist ein *unendlichdimensionaler* Funktionenraum. Im Rechner betrachten wir Funktionenklassen, die durch *endlich viele* Parameter bestimmt sind (nicht unbedingt lineare Unterräume), z.B.:

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n \qquad \text{(Polynome)}$$

$$r(x) = \frac{a_0 + a_1 x + \cdots + a_n x^n}{b_0 + b_1 x + \cdots + b_m x^m} \qquad \text{(rationale Funktionen)}$$

$$t(x) = \frac{1}{2} a_0 + \sum_{k=1}^{n} (a_k \cos(kx) + b_k \sin(kx)) \qquad \text{(trigonom. Polynome)}$$

$$e(x) = \sum_{k=1}^{n} a_k \exp(b_k x) \qquad \text{(Exponentialsumme)}$$

Approximation

Grundaufgabe der *Approximation:*

Gegeben eine Menge von Funktionen P (Polynome, rationale Funktionen, ...) und eine Funktion f (z.B. $f \in C[a,b]$), finde $g \in P$, so dass der Fehler $f - g$ in geeigneter Weise minimiert wird.

Beispiele:

$$\left(\int_a^b (f-g)^2 \, dx \right)^{1/2} \to \min \qquad \text{(2-Norm)}$$

$$\max_{a \le x \le b} |f(x) - g(x)| \to \min \qquad (\infty\text{-Norm})$$

$$\max_{i \in \{0,\ldots,n\}} |f(x_i) - g(x_i)| \to \min \qquad \text{für } a \le x_i \le b,\ i = 0, \ldots, n$$

Interpolation

Man spricht von *Interpolation*, wenn g durch

$$g(x_i) = y_i := f(x_i) \quad i = 0, \ldots, n$$

festgelegt wird.

Dies ist ein Spezialfall der Approximationsaufgabe:

- Der Fehler $f - g$ wird nur in den *Stützstellen* $x_i, i = 0, \ldots, n$, betrachtet.
- In diesen endlich vielen Punkten muss die Abweichung nicht nur minimal, sondern Null sein.

Polynominterpolation

P_n sei die Menge der Polynome über $\mathbb{R}$ vom Grad kleiner gleich $n \in \mathbb{N}_0$:

$$P_n := \{p(x) = \sum_{i=0}^{n} a_i x^i \mid a_i \in \mathbb{R}\}$$

P_n ist ein $n+1$-dimensionaler Vektorraum.
Die *Monome* $1, x, x^2, \ldots, x^n$ bilden eine Basis von P_n.

Zu gegebenen (paarweise verschiedenen) $n+1$ Stützstellen $x_0, x_1, \ldots, x_n$ ist die Interpolationsaufgabe dann

$$\text{Finde } p \in P_n: \quad p(x_i) = y_i := f(x_i), \quad i = 0, \ldots, n$$

Polynominterpolation

Das ist äquivalent zum linearen Gleichungssystem

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{=:V[x_0,\ldots,x_n]} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Die Matrix $V[x_0, \ldots, x_n]$

- heißt *Vandermonde-Matrix*
- ist genau dann regulär, wenn alle x_i paarweise verschieden sind
- ist schlecht konditioniert: $\text{cond}_\infty(V) = \mathcal{O}(2^n)$ für x_i positiv!
- benötigt Rechenaufwand in $\mathcal{O}(n^3)$ zum Lösen des LGS

Polynominterpolation

Problem:
Aufstellen der Vandermonde-Matrix $V[x_0, \ldots, x_n]$ und anschließendes Lösen mit den vorgestellten Verfahren ist wegen der sehr schlechten Kondition nicht zu empfehlen.

Geht das auch besser und einfacher?

Ursache ist die Wahl der Monombasis $1, x, x^2, \ldots, x^n$, die auf eine sehr ungünstige Formulierung der Interpolationsaufgabe führt. Im folgenden werden wir uns mögliche Alternativen anschauen.

Langrange-Interpolation

Definition 7.1 (Lagrange-Polynome)

Zu den paarweise verschiedenen Stützstellen $x_i, i = 0, \ldots, n$, definiere die sog. *Lagrange-Polynome*:

$$L_i^{(n)}(x) := \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \ldots, n \quad (n+1 \text{ Stück})$$

Die $L_i^{(n)}$ haben Grad n, es gilt

$$L_i^{(n)}(x_k) = \delta_{ik} = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

und die $L_i^{(n)}$ bilden eine Basis von P_n.

$\rightsquigarrow$ Beweis: Tafel

Existenz und Eindeutigkeit

In der Lagrange-Basis sind die gesuchten Koeffizienten a_i gerade die Werte in den Stützstellen: $a_i = y_i$. Die Interpolationsaufgabe ist so also besonders einfach zu lösen.

Satz 7.2 (Eindeutigkeit der Polynominterpolation)

Zu gegebenen paarweise verschiedenen Stützstellen $x_0, \ldots, x_n$ gibt es genau ein Polynom p vom Grad n mit

$$p(x_i) = y_i \quad i = 0, \ldots, n,\ y_i \in \mathbb{R}$$

Die Interpolationsaufgabe ist also eindeutig lösbar.

$\rightsquigarrow$ Beweis: Tafel

Newton-Darstellung

Nachteil der Lagrange-Polynome:
Hinzufügen einer Stützstelle ändert alle bisherigen Basispolynome.
Eignet sich daher nicht für die "inkrementelle" Erstellung des Interpolationspolynoms.

Besser ist hier die *Newton-Darstellung* mit Basispolynomen

$$N_0(x) = 1; \quad i = 1, \ldots, n \colon N_i(x) = \prod_{j=0}^{i-1} (x - x_j)$$

N_i hat jeweils Grad i, es gilt

$$\forall k < i \colon N_i(x_k) = 0$$

und die $N_0, \ldots, N_n$ bilden eine Basis von P_n.

Gestaffelte Berechnung

In x_0 sind alle Newton-Basispolynome bis auf N_0 Null, in x_1 alle bis auf N_0 und N_1, usw.

Die Interpolationsaufgabe

$$p(x_k) = \sum_{i=0}^{n} a_i N_i(x_k) = \sum_{i=0}^{k} a_i N_i(x_k) = y_k \quad k = 0, \ldots, n$$

führt daher auf die folgende gestaffelte Berechnung:

$$a_0 = y_0; \quad k = 1, \ldots, n\colon\ a_k = \left[y_k - \sum_{i=0}^{k-1} a_i N_i(x_k) \right] / N_k(x_k)$$

Hintergrund

Die Polynominterpolation in der Sprache der Linearen Algebra:

- Die Monombasis $x^i, i = 0, \ldots, n$, führt auf eine Matrix $V[x_0, \ldots, x_n]$ (Vandermonde-Matrix), die dicht besetzt und sehr schlecht konditioniert ist.
- Die Lagrange-Basis $L_i^{(n)}, i = 0, \ldots, n$, führt stattdessen auf eine Einheitsmatrix, so dass die Lösung trivial ist. Eine Erweiterung der Stützstellen ändert aber alle Basisfunktionen.
- Die Newton-Basis $N_i, i = 0, \ldots, n$, wiederum führt auf eine untere Dreiecksmatrix. Die gestaffelte Berechnung entspricht daher gerade dem Vorwärtseinsetzen. Das Lösen ist aufwendiger als bei der Lagrange-Basis, weitere Stützstellen führen aber nur zu einer Erweiterung der unteren Dreiecksmatrix.

Dividierte Differenzen

Satz 7.3 (Dividierte Differenzen)

Man definiert rekursiv die sog. "dividierten Differenzen"

$$\forall i = 0, \ldots, n: \quad y[x_i] := y_i \quad \text{(Werte an den Stützstellen)}$$

$$\forall k = 1, \ldots, n - i:$$

$$y[x_i, \ldots, x_{i+k}] := \frac{y[x_{i+1}, \ldots, x_{i+k}] - y[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Dann gilt

$$p(x) = \sum_{i=0}^{n} y[x_0, \ldots, x_i] N_i(x)$$

$\rightsquigarrow$ Beweis: [Rannacher]

Dividierte Differenzen

Die dividierten Differenzen ordnet man in einem Tableau an:

$$
\begin{array}{llll}
y_0 = y[x_0] & \rightarrow & y[x_0, x_1] & \rightarrow \quad y[x_0, x_1, x_2] \quad \rightarrow \quad y[x_0, x_1, x_2, x_3] \\
 & \nearrow & & \nearrow \\
y_1 = y[x_1] & \rightarrow & y[x_1, x_2] & \rightarrow \quad y[x_1, x_2, x_3] \\
 & \nearrow & & \nearrow \\
y_2 = y[x_2] & \rightarrow & y[x_2, x_3] & \\
 & \nearrow & & \\
y_3 = y[x_3] & & &
\end{array}
$$

In der ersten Zeile finden sich dann die gesuchten Koeffizienten $a_i, i = 0, \ldots, n$, der Basispolynome.
Diese Form der Berechnung benötigt die Werte der $N_i(x_k)$ nicht und ist zudem stabiler.

Beispiel I

Beispiel 7.4

Gegeben seien die folgenden Paare von Stützstellen und Werten:

$$(x_0 = 0, y_0 = 1), \quad (x_1 = 1, y_1 = 4), \quad (x_2 = 2, y_2 = 3)$$

Die Monombasis führt auf das Gleichungssystem

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$$

mit der Lösung $[1, 5, -2]^T$, also

$$p(x) = 1 + 5 \cdot x - 2 \cdot x^2$$

Beispiel II

Beispiel 7.4

Die Lagrange-Basis führt bei diesen Werten auf

$$\begin{aligned} p(x) &= 1 \cdot L_0^{(2)}(x) + 4 \cdot L_1^{(2)}(x) + 3 \cdot L_2^{(2)}(x) \\ &= 1 \cdot \frac{x-1}{0-1} \cdot \frac{x-2}{0-2} + 4 \cdot \frac{x-0}{1-0} \cdot \frac{x-2}{1-2} + 3 \cdot \frac{x-0}{2-0} \cdot \frac{x-1}{2-1} \\ &= \frac{1}{2} \cdot (x-1) \cdot (x-2) - 4 \cdot x \cdot (x-2) + \frac{3}{2} \cdot x \cdot (x-1) \end{aligned}$$

Die einzelnen Basispolynome können für eine gegebene Menge von Stützstellen $x_0, \ldots, x_n$ einmal ausmultipliziert werden, danach sind weitere Interpolationsaufgaben leicht zu lösen.

Beispiel III

Beispiel 7.4

Für die Newton-Basis ergibt sich das Tableau

$$\begin{array}{lcccc} y_0 = a_0 = 1 & \to & a_1 = \frac{4-1}{1-0} = 3 & \to & a_2 = \frac{(-1)-3}{2-0} = -2 \\ & \nearrow & & \nearrow & \\ y_1 = 4 & \to & \frac{3-4}{2-1} = -1 & & \\ & \nearrow & & & \\ y_2 = 3 & & & & \end{array}$$

und daher

$$\begin{aligned} p(x) &= 1 \cdot N_0(x) + 3 \cdot N_1(x) - 2 \cdot N_2(x) \\ &= 1 + 3 \cdot x - 2 \cdot x \cdot (x-1) \end{aligned}$$

Eine weitere Stützstelle kann hier leicht hinzugefügt werden.

Auswertung von Polynomen

Um ein Polynom der Form

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_n x^n$$

an einer Stelle x auszuwerten, verwendet man aus Effizienz- und Stabilitätsgründen das sog. *Horner-Schema*:

$$b_n := a_n; \quad k = n-1, \ldots, 0 \colon b_k := a_k + x \cdot b_{k+1}; \quad p(x) = b_0$$

Für ein Polynom in Newton-Darstellung

$$p(x) = \sum_{i=0}^{n} a_i N_i(x) = a_0 + a_1 N_1(x) + \cdots + a_n N_n(x)$$

ergibt sich analog die Rekursion

$$b_n := a_n; \quad k = n-1, \ldots, 0 \colon b_k := a_k + (x - x_k) \cdot b_{k+1}; \quad p(x) = b_0$$

Interpolationsfehler

Sei $y_i = f(x_i), i = 0, \ldots, n$, die Auswertung einer Funktion f an $n+1$ paarweise verschiedenen Stützstellen, und $p(x)$ das Polynom vom Grad n, das die zugehörige Interpolationsaufgabe löst.

Die Differenz

$$e(x) := f(x) - p(x)$$

erfüllt nach Konstruktion die Bedingung

$$e(x_i) = 0 \quad \text{für} \quad i = 0, \ldots, n$$

Frage: Wie groß kann die Differenz an anderen Stellen werden?

Interpolationsfehler

Satz 7.5 (Interpolationsfehler)

Sei $f(x)$ $n+1$-mal stetig differenzierbar auf $[a,b]$ und

$$a \leq x_0 < x_1 < \cdots < x_n \leq b.$$

Dann gibt es zu jedem $x \in [a,b]$ ein $\xi_x \in \overline{(x_0, \ldots, x_n, x)}$ (kleinstes Intervall, das alle diese Punkte enthält), so dass

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{j=0}^{n} (x - x_j)$$

$\rightsquigarrow$ Beweis: Tafel

Anmerkungen

Für den Spezialfall *äquidistanter* Stützstellen, d.h.

$$x_{k+1} - x_k = h \quad \text{für} \quad k = 0, \ldots, n-1,$$

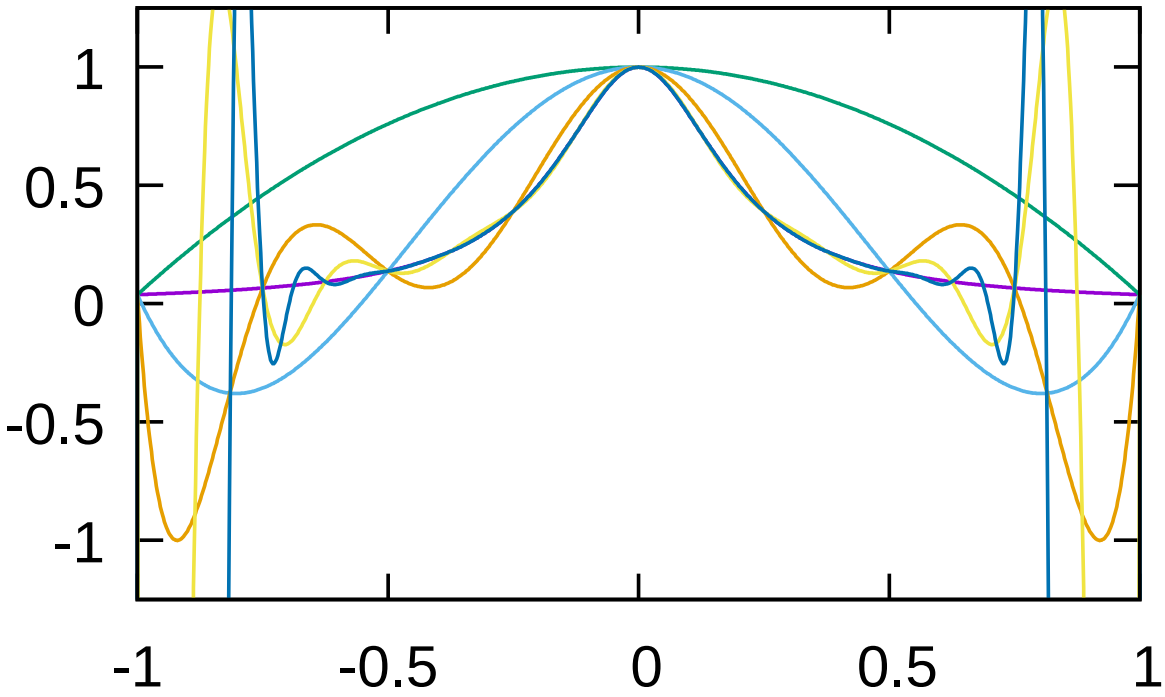
gilt damit

$$|f(x) - p(x)| \leq |f^{(n+1)}(\xi_x)| \cdot h^{n+1}$$

Für $|f^{(n+1)}|$ beschränkt und $n \to \infty$ gilt also $|f(x) - p(x)| \to 0$.

Allerdings sind die höheren Ableitungen auch einfacher Funktionen für $n \to \infty$ oft *nicht* beschränkt, sondern wachsen sehr schnell.

Runges Gegenbeispiel



Polynominterpolation von Runges Funktion $f(x) = (1 + 25x^2)^{-1}$ mit äquidistanten Stützstellen (3, 5, 9, 17 bzw. 33 Wertepaare).
Die Minima / Maxima der letzten beiden Polynome sind $-14.35/1.40$ bzw. $-5059/2.05$ (!).

Anmerkungen

Bemerkung 7.6

Laut *Approximationssatz von Weierstraß* lässt sich jede Funktion aus $C^0([a, b])$ gleichmäßig durch Polynome approximieren.
Die beobachteten Phänomene sind kein Widerspruch, denn:

- Die Approximation muss nicht durch Interpolation erfolgen (der Beweis nutzt Bernstein-Polynome).
- Mit nicht-äquidistanten Stützstellen erhält man bereits deutlich bessere Ergebnisse (wenn man weiß, wie sie zu wählen sind...).

Bemerkung 7.7

Allgemein gilt für "Methoden hoher (Polynom-) Ordnung", dass entsprechende Differnzierbarkeit vorliegen muss.

Konditionierung

Wenn $p(x; y)$ das Interpolationspolynom zu den Ordinatenwerten $(y_0, \ldots, y_n)^T$ und *festen* Abszissenwerten $(x_0, \ldots, x_n)^T$ ist, gilt

$$\begin{aligned} p(x; y + \Delta y) - p(x; y) &= \sum_{i=0}^{n} (y_i + \Delta y_i) L_i^{(n)}(x) - \sum_{i=0}^{n} y_i L_i^{(n)}(x) \\ &= \sum_{i=0}^{n} \Delta y_i L_i^{(n)}(x) \end{aligned}$$

Somit ist

$$\frac{p(x; y + \Delta y) - p(x; y)}{p(x; y)} = \sum_{i=0}^{n} \frac{L_i^{(n)}(x) y_i}{p(x; y)} \cdot \frac{\Delta y_i}{y_i}$$

Für große n kann $L_i^{(n)}$ sehr groß werden, die Interpolationsaufgabe ist dann schlecht konditioniert!

Numerische Differentiation

Problem:
Berechne die Ableitung (der Ordnung k) einer tabellarisch gegebenen Funktion oder einer im Rechner gegebenen Funktion (im Sinne der Informatik).

Idee:
Erstelle Interpolationspolynom zu bestimmten Stützstellen, leite dieses ab und werte aus.

Dabei sei zunächst Ableitungsordnung = Polynomgrad.

Numerische Differentiation

Die Lagrange-Polynome sind

$$L_i^{(n)}(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = \underbrace{\prod_{j \neq i} (x_i - x_j)^{-1}}_{=:\lambda_i \in \mathbb{R}} \cdot x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_0,$$

also liefert n-faches Differenzieren:

$$\frac{d^n}{dx^n} L_i^{(n)}(x) = n! \cdot \lambda_i$$

Somit gilt für die n-te Ableitung eines Interpolationspolynoms vom Grad n:

$$\frac{d^n}{dx^n} \left(\sum_{i=0}^{n} y_i L_i^{(n)}(x) \right) = n! \cdot \sum_{i=0}^{n} y_i \lambda_i \quad \text{(unabh. von } x\text{)}$$

Numerische Differentiation

Eine Aussage über den entstehenden Fehler liefert:

Satz 7.8

Sei $f \in C^n([a,b])$ und $a = x_0 < x_1 < \cdots < x_n = b$. Dann gibt es $\xi \in (a,b)$, so dass

$$f^{(n)}(\xi) = n! \cdot \sum_{i=0}^{n} y_i \lambda_i$$

Die über das Interpolationspolynom berechnete Ableitung stimmt also in mindestens einem Punkt mit der von f überein.

$\rightsquigarrow$ Beweis: Tafel

Numerische Differentiation

Für äquidistante Stützstellen, $x_i = x_0 + ih, 0 \leq i \leq n$, erhält man speziell

$$f^{(n)}(x) \approx h^{-n} \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} y_i, \quad \text{z.B.} \quad f^{(1)}(x) \approx \frac{y_1 - y_0}{h},$$

$$f^{(2)}(x) \approx \frac{y_2 - 2y_1 + y_0}{h^2}, \quad f^{(3)}(x) \approx \frac{y_3 - 3y_2 + 3y_1 - y_0}{h^3}$$

Mittels Taylorentwicklung zeigt man

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2) \qquad \text{für} \quad f \in C^3,$$

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2) \qquad \text{für} \quad f \in C^4$$

Extrapolation zum Limes I

Eine Größe $a(h)$ sei im Rechner für $h > 0$ berechenbar, nicht jedoch für $h = 0$. Man möchte

$$a(0) = \lim_{h \to 0} a(h)$$

mit guter Genauigkeit berechnen.

Beispiel 7.9

Mögliche Anwendungen:

1. Regel von de l'Hospital:

$$a(0) = \lim_{h \to 0} \frac{\cos(x) - 1}{\sin(x)} (= 0)$$

Extrapolation zum Limes II

Beispiel 7.9

2. Numerische Differentiation:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

(für kleine h tritt Auslöschung auf)

3. Numerische Integration:

$$\int_a^b f(x) = \lim_{N \to \infty} \sum_{i=1}^{n} N^{-1} f\left(a + (i - \frac{1}{2}) \frac{b-a}{N}\right)$$

(setze $h := N^{-1}$)

Extrapolation zum Limes III

Beispiel 7.9

4. Numerische Lösung des Anfangswertproblems

$$y'(t) = f(t, y(t)) \quad \text{auf} \quad [0, T]; \quad y(0) = y_0$$

Setze

$$h = N^{-1}; \quad y_n = y_{n-1} + h \cdot f(t, y_n - 1); \quad y(T) \approx y_N$$

Hier ist $h \to 0$ gleichbedeutend mit $N \to \infty$ und entsprechend ansteigendem Aufwand.

Grundlegende Idee

Idee der Extrapolation:

Zu $h_0 > h_1 > \cdots > h_n > 0$ bestimme Interpolationspolynom

$$p(h_i) = a(h_i) \quad i = 0, \ldots, n$$

und berechne

$$a(0) \approx p(0)$$

(Extrapolation statt Interpolation, da $0 \notin [h_n, \ldots, h_0]$)

Beispiel I

Beispiel 7.10

Für $a(h) = (\cos(h) - 1) \cdot (\sin(h))^{-1}$ erhält man für

$$\begin{aligned} h_0 = 1/8: &\quad a(h_0) = -6.258151 \cdot 10^{-2} \\ h_1 = 1/16: &\quad a(h_1) = -3.126018 \cdot 10^{-2} \\ h_2 = 1/32: &\quad a(h_2) = -1.562627 \cdot 10^{-2} \end{aligned}$$

(halbieren von h halbiert also $a(h)$), und bei Extrapolation mit einem Polynom p_2 vom Grad 2:

$$a(0) \approx p_2(0) = -1.02 \cdot 10^{-5}$$

also sehr viel besser als die ursprünglich berechneten Näherungen oder eine mögliche direkte Auswertung für $h \ll 1$ (Auslöschung)!

Beispiel II

Beispiel 7.10

Warum ist das so gut?

Sei $h_i = h \cdot r^i$ mit $r < 1$ (geometrische Verteilung), z.B. $r = 1/2$, und sei p das Interpolationspolynom von a zu den Stützstellen h_i. Dann gilt für den Fehler der Extrapolation

$$|p(0) - a(0)| \leq \|V^{-T}\|_\infty |a^{(n+1)}(\xi)| \frac{h^{n+1}}{(n+1)!}(1 + r^{n+1})$$

mit der Vandermonde-Matrix V und $\xi \in (0, h)$, sofern a hinreichend differenzierbar ist.

$\rightsquigarrow$ Beweis: Tafel

Extrapolation bei Ableitungen

Entscheidend ist die Taylor-Entwicklung von a in Null (Mclaurin-Entwicklung). Für den bekannten Differenzenquotienten der zweiten Ableitung erhält man

$$\begin{aligned} a(h) &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \\ &= f''(x) + \frac{h^2}{2 \cdot 4!} f^{(4)}(x) + \cdots + \frac{h^{2n}}{2 \cdot (2n+2)!} f^{(2n+2)}(x) \\ &\qquad + \frac{h^{2n+2}}{2 \cdot (2n+4)!} [f^{(2n+4)}(\xi_+) + f^{(2n+4)}(\xi_-)] \\ &= p_x(h^2) + \mathcal{O}(h^{2(n+1)}) \end{aligned}$$

Man kann also (falls f hinreichend differenzierbar ist) mit jeder Auswertung sogar zwei h-Potenzen gewinnen!

Bernstein-Polynome I

Wir gehen nun von der Interpolation zur Approximation über. Speziell für Kurven, d.h. Funktionen $u(t)\colon [a,b] \to \mathbb{R}^d$, haben sich die Bernstein-Polynome bewährt.

Definition 7.11 (Bernstein-Polynome)

Die Polynome

$$\beta_i^{(n)}(t) := \binom{n}{i}(1-t)^{n-i}t^i \quad i = 0,\dots,n$$

heißen *Bernstein-Polynome* auf dem Interval $[0,1]$.

Bernstein-Polynome II

Definition 7.11 (Bernstein-Polynome)

Mittels der affin-linearen Transformation

$$\phi\colon [a,b] \to [0,1], \quad p(t) = \frac{t-a}{b-a}$$

definiert man zusätzlich Bernstein-Polynome auf einem allgemeinen Intervall $[a,b]$:

$$\beta^{(n)}_{i,[a,b]}(t) := \beta^{(N)}_i(\phi(t)) = \binom{n}{i}(b-a)^{-n}(b-t)^{n-i}(t-a)^i$$

Eigenschaften I

Satz 7.12 (Eigenschaften der Bernstein-Polynome)

Die Bernstein-Polynome haben die folgenden Eigenschaften:

1. Zerlegung der Eins:
$$\sum_{i=0}^{n} \beta_i^{(n)}(t) = 1$$
2. $t = 0$ ist i-fache Nullstelle von $\beta_i^{(n)}$
3. $t = 1$ ist $(n - i)$-fache Nullstelle von $\beta_i^{(n)}$
4. Symmetrie:
$$\beta_i^{(n)}(t) = \beta_{n-i}^{(n)}(1 - t)$$

Eigenschaften II

Satz 7.12 (Eigenschaften der Bernstein-Polynome)

5. Positivität:

$$0 \leq \beta_i^{(n)}(t) \leq 1 \text{ für } t \in [0,1], \quad 0 < \beta_i^{(n)}(t) \text{ für } t \in (0,1)$$

6. $\beta_i^{(n)}, n \neq 0$, hat in $[0,1]$ genau ein Maximum in i/n.
7. Die $\beta_i^{(n)}$ sind lin. unabh. und bilden eine Basis von P_n.
8. Die Bernstein-Polynome erlauben eine rekursive Darstellung:

$$\beta_i^{(n)}(t) = \begin{cases} (1-t)\beta_0^{(n-1)}(t) & i = 0 \\ t\beta_{i-1}^{(n-1)}(t) + (1-t)\beta_i^{(n-1)}(t) & 0 < i < n \\ t\beta_{n-1}^{(n-1)}(t) & i = n \end{cases}$$

Eigenschaften III

Satz 7.12 (Eigenschaften der Bernstein-Polynome)

9. Für die erste Ableitung gilt die Rekursionsformel:

$$\frac{d}{dt}\beta_i^{(n)} = \begin{cases} -n\beta_0^{(n-1)}(t) & i = 0 \\ n[\beta_{i-1}^{(n-1)}(t) - \beta_i^{(n-1)}(t)] & 0 < i < n \\ n\beta_{n-1}^{(n-1)}(t) & i = n \end{cases}$$

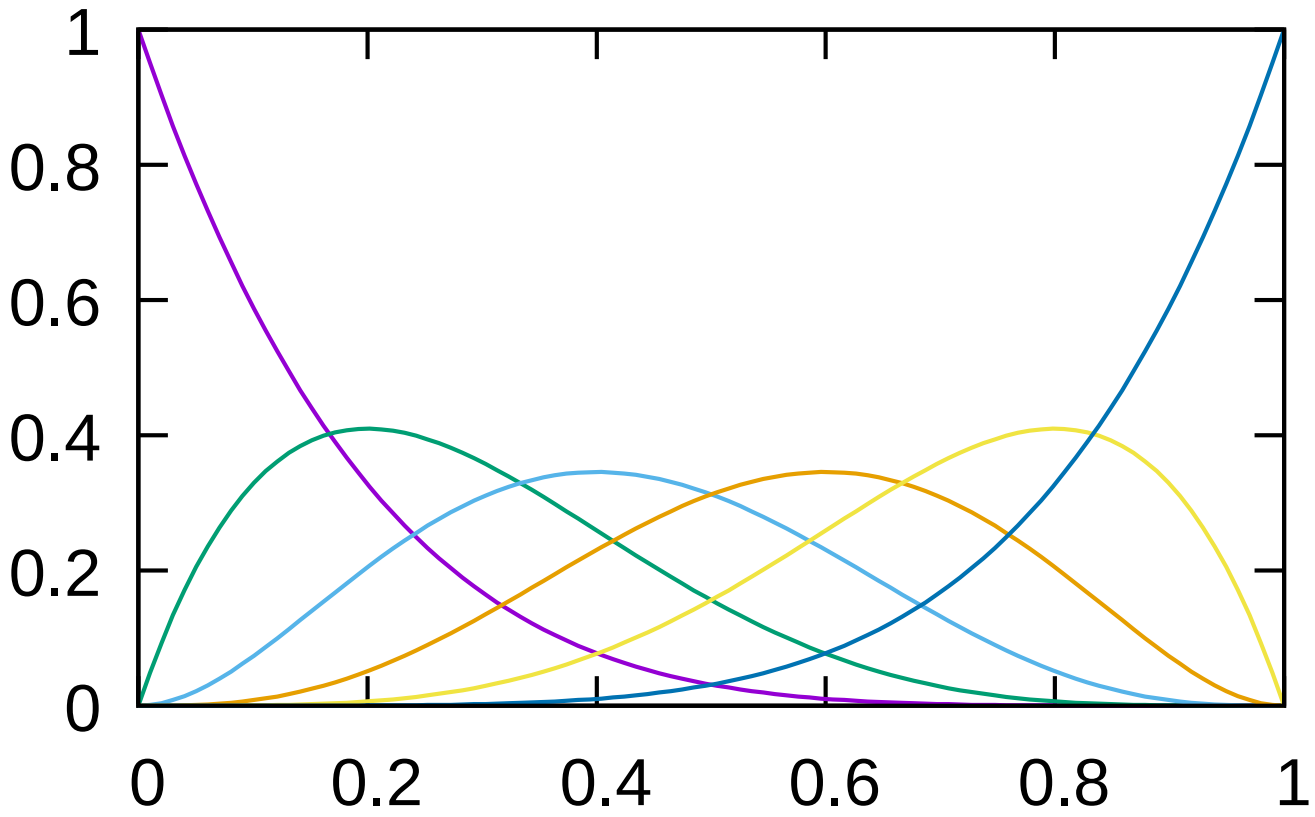
   Achtung: mit dem Superskript sind hier jeweils die Grade der Polynome gemeint, nicht höhere Ableitungen!

Zusammenfassend bilden die Bernstein-Polynome eine "Konvexkombination" (positive Zerlegung der Eins) mit zusätzlicher Symmetrieeigenschaft und rekursiver Darstellung.

$\rightsquigarrow$ Beweis: Tafel

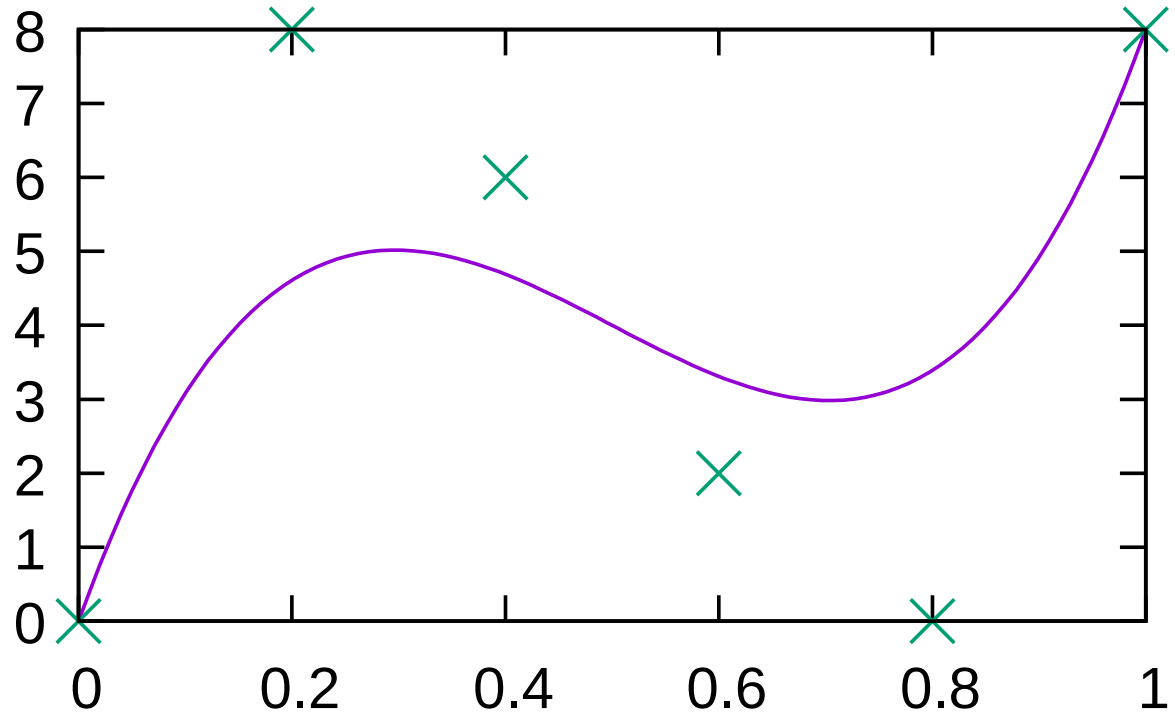
Graphische Darstellung

Bernstein-Polynome vom Grad 5:



Approximation mit Bernstein-Polynomen

Approximation des Datenvektors $y = (0, 8, 6, 2, 0, 8)^T$ mit Bernstein-Polynomen:



Die Daten werden nirgends im Intervall interpoliert, stattdessen hat jeder Datenpunkt einen "Bereich des Einflusses".

Bézier-Kurven

Kurvendarstellung mittels Bernstein-Polynomen:

Definition 7.13 (Bézier-Kurven)

Für gegebene Punkte $b_0, \ldots, b_n \in \mathbb{R}^d$ heißt das vektorwertige Polynom

$$B(t) := \sum_{i=0}^{n} b_i \beta_i^{(n)}(t)$$

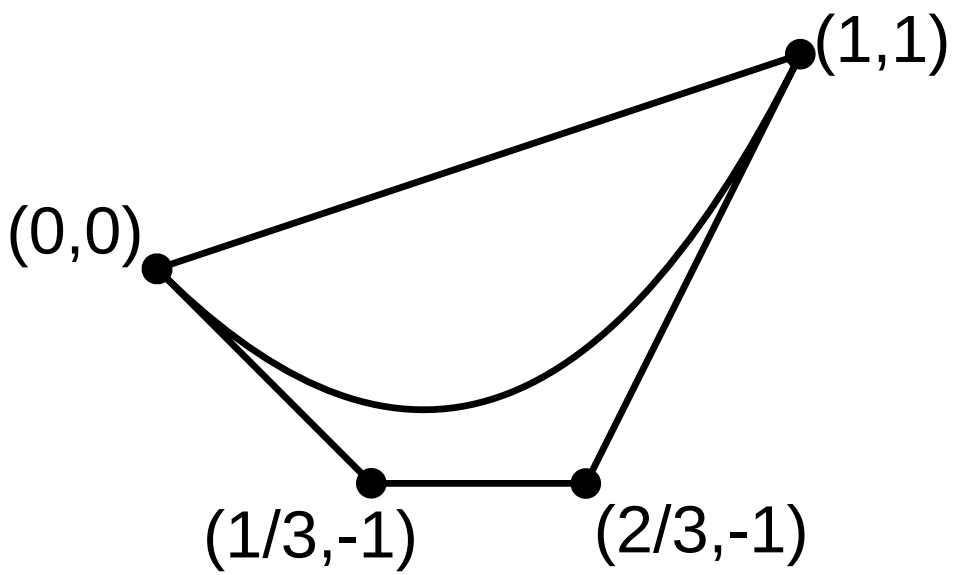
Bézier-Kurve. Bézier war Ingenieur bei Renault, und die nach ihm benannten Kurven sind die Grundlage von Vektorgraphiken.

Beispiel

Beispiel 7.14

Betrachte die Punkte $b_0 = (0,0)^T$, $b_1 = (1/3,-1)^T$, $b_2 = (2/3,-1)^T$ und $b_3 = (1,1)^T$, die zugehörige Bézier-Kurve ist:

$$B(t) = \sum_{i=0}^{3} b_i \beta_i^{(3)}(t) = \begin{pmatrix} t \\ t^3 + 3t^2 - 3t \end{pmatrix}$$



Eigenschaften

Es gelten folgende Eigenschaften:

1. Die Bézier-Kurve $B(t)$ liegt in der konvexen Hülle der Bézier-Punkte $b_0, \ldots, b_n$.
2. Es ist $B(0) = b_0$ und $B(1) = b_n$.
3. Die Ableitung (Tangente der Kurve) hat an den Endpunkten die Richtung $(b_1 - b_0)$ bzw. $(b_n - b_{n-1})$.

Dadurch sind also glatte, lokalisierte Kurven mit klar definierten Enden gegeben.

$\rightsquigarrow$ Rechnung: Tafel

Eigenschaften

Bernstein-Polynome erlauben eine gute Kontrolle über die *Werte* eines Polynoms durch dessen Koeffizienten.

Vergleich mit anderen Polynombasen:

- Monom- und Newton-Basis: keine einfache Kontrolle der Werte
- Langrange-Basis: exakte Kontrolle an den Stützstellen, dazwischen aber (sehr) große Abweichungen
- Bernstein-Basis: $\min\{b_i\} \leq \sum_{i=0}^{n} b_i \beta_i^{(n)} \leq \max\{b_i\}$

Das ist eine direkte Folge der Konvexkombination.

Algorithmus von de Casteljau

Für gegebene Bézier-Punkte $b_0^{(0)}, \ldots, b_n^{(0)}$ ergibt sich mittels der Rekursionsformel aus Satz 7.12:

$$B(t) = \sum_{i=0}^{n} b_i^{(0)} \beta_i^{(n)}(t) = \sum_{i=0}^{n-1} \underbrace{[b_i^{(0)}(1-t) + b_{i+1}^{(n-1)}(t)]}_{=: b_i^{(1)}} \beta_i^{(n-i)}(t)$$

Rekursiv setzt man daher die (von t abhängigen) "Koeffizienten"

$$b_i^{(k)} := b_i^{(k-1)}(1-t) + b_{i+1}^{(k-1)} t \quad 0 < k \leq n, \quad 0 \leq i \leq n-k,$$

um $B(t) = b_0^{(n)}$ zu berechnen.

$\rightsquigarrow$ Rechnung: Tafel

Algorithmus von de Casteljau

Dieser Algorithmus ist nach de Casteljau, Physiker und Mathematiker bei Citroën, benannt. Die Struktur der Auswertung ist mit den dividierten Differenzen vergleichbar:

$$
\begin{array}{ccccccc}
b_0 = b_0^{(0)} & \to & b_0^{(1)} & \to & b_0^{(2)} & \to & b_0^{(3)} = B(t) \\
 & \nearrow & & \nearrow & & \nearrow & \\
b_1 = b_1^{(0)} & \to & b_1^{(1)} & \to & b_1^{(2)} & & \\
 & \nearrow & & \nearrow & & & \\
b_2 = b_2^{(0)} & \to & b_2^{(1)} & & & & \\
 & \nearrow & & & & & \\
b_3 = b_3^{(0)} & & & & & &
\end{array}
$$

Algorithmus von de Casteljau

Bei fester Wahl von t ist damit auch ein Verfahren gegeben, um eine Bézier-Kurve in zwei Teilkurven zu zerlegen. Die neuen Bézier-Punkte sind

$$b_0^{(0)}, b_0^{(1)}, \ldots, b_0^{(n)} \quad \text{bzw.} \quad b_0^{(n)}, b_1^{(n-1)}, \ldots, b_n^{(0)}$$

(die ersten bzw. letzten Punkte der Rekursionsebenen).

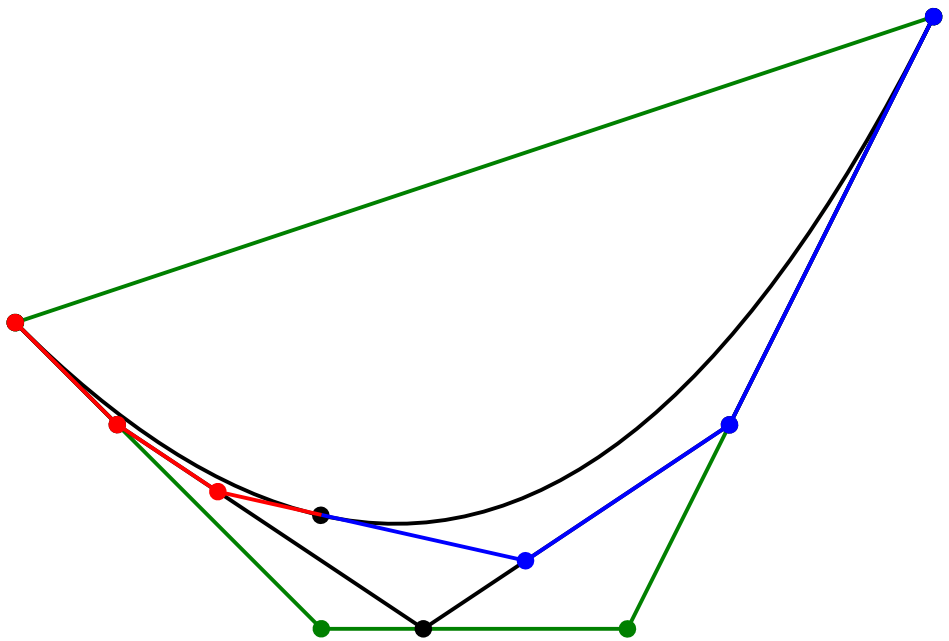
Offensichtlich liegen die Start- / Endpunkte $b_0^{(0)}$, $b_0^{(n)}$ und $b_n^{(0)}$ auf der Kurve. Dass die Kurven sich auch tatsächlich überlagern, ist eine Folge der Berechnung der $b_i^{(j)}$ und der rekursiven Darstellung der Bernstein-Polynome.

Beispiel

Am besten lässt sich das an der graphischen Darstellung erkennen:

Beispiel 7.15

Zerlegung der Kurve aus Beispiel 7.14 an der Stelle $t = 1/3$, bzw. Auswertung an dieser Stelle: rekursives Dritteln der Verbindungslinien, resultierende neue Punkte in rot und blau.



Splines

Wir kehren jetzt wieder zur Interpolation zurück.

Bis jetzt galt:

- Anzahl Stützstellen = Polynomgrad $+1$
- Viele Stützstellen $\Longrightarrow$ hoher Polynomgrad $\Longrightarrow$ evtl. starke Abweichung zwischen den Stützstellen

Idee: Benutze *stückweise* Polynome niedrigen Grades.

Splines

Definition 7.16 (Spline-Raum)

Sei $X = (x_0, x_1, \ldots, x_n)$ mit $a = x_0 < x_1 < \cdots < x_n = b$ eine Zerlegung des Intervalls $[a, b]$ und sei $m \in \mathbb{N}$. Die Menge

$$S^m(X) := \{s \in C^{m-1}([a,b]) : s|_{[x_i, x_{i+1}]} \in P_m,\ 0 \leq i < n\}$$

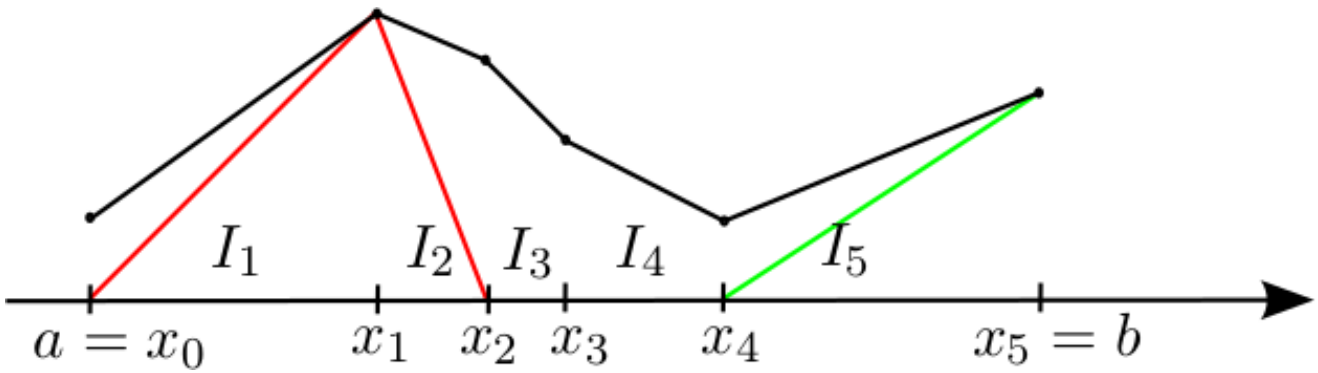
heißt *Spline-Raum* vom Grad m über der Zerlegung X.

Beispiel

Beispiel 7.17

$s \in S^1(X)$ bedeutet:

- s ist auf jedem Teilintervall $[x_i, x_{i+1}]$ ein Polynom vom Grad 1
- $S^1(X) \subset C^0([a,b])$, also ist s stetig



Dieser Vektorraum ist relevant beim Lösen partieller Differentialgleichungen, und wird dort als Finite-Elemente-Raum P_1 bezeichnet.

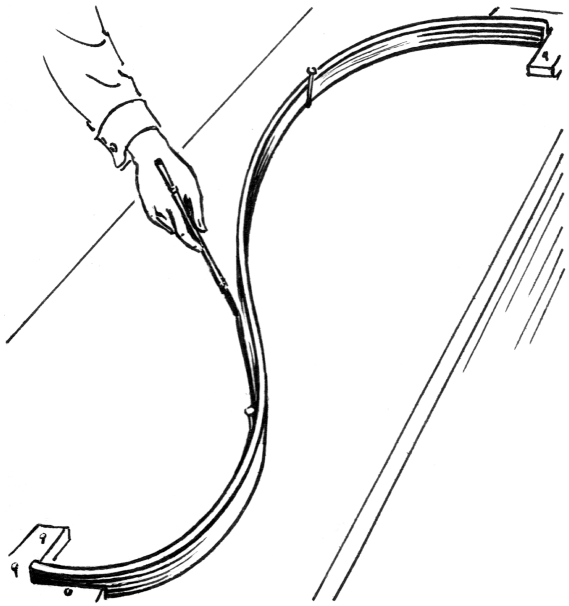
Kubische Splines

In der Praxis ist $S^3(X)$ sehr beliebt. $S^3(X)$ heißt *Raum der kubischen Splines*.

Geschichte: "Straklatte" zur Konstruktion glatter Kurven im Schiffs- und Flugzeugbau.

Ein dünnes Balsaholz biegt sich unter Energieminimierung

$$\int_a^b \frac{|y''(t)|^2}{1+|y'(t)|} = dt \approx \int_a^b |y''(t)|^2 \, dt \to \min$$



Quelle: Pearson Scott Foresman, gemeinfrei

Kubische Splines

Eine Funktion $s \in S^3(X)$ besteht stückweise aus n Polynomen:

$$s(x) = \begin{cases} p_i(x) & x \in [x_{i-1}, x_i), \quad i \in \{1, \dots, n\} \\ p_n(x_n) & x = x_n \end{cases}$$

Für die Polynome p_i gelten folgende Bedingungen:

- Interpolationsbedingung (Stetigkeit) in allen Punkten:

$$2n \text{ Bedingungen:} \quad i = 1, \dots, n: \begin{cases} p_i(x_{i-1}) = y_{i-1} \\ p_i(x_i) = y_i \end{cases}$$

- Stetigkeit von erster und zweiter Ableitung an *inneren* Punkten:

$$2(n-1) \text{ Bedingungen:} \quad i = 1, \dots, n-1: \begin{cases} p_i'(x_i) = p_{i+1}'(x_i) \\ p_i''(x_i) = p_{i+1}''(x_i) \end{cases}$$

$\Longrightarrow$ zusammen $4n - 2$ Bedingungen

Kubische Splines

Man hat $4n$ Freiheitsgrade, pro Polynom p_i vier (da Grad 3).

Die fehlenden 2 Bedingungen erhält man als *Randbedingungen* an den Stellen x_0 und x_n. Dabei gibt es verschiedene Varianten:

1. "Natürliche" Randbedingungen, "Krümmung Null":
$$p_1''(x_0) = p_n''(x_n) = 0$$
2. Hermite-Randbedingungen:
$$p_1'(x_0) = f'(x_0), \quad p_n'(x_n) = f'(x_n)$$
(f ist die zu interpolierende Funktion)
3. Periodische Randbedingungen:
$$p_1'(x_0) = p_n'(x_n), \quad p_1''(x_0) = p_n''(x_n)$$

Wir behandeln im folgenden nur die natürlichen Randbedingungen.

Berechnung kubischer Splines

Wir schreiben die Teilpolynome eines kubischen Splines in der Form

$$p_i(x) = a_0^{(i)} + a_1^{(i)}(x - x_i) + a_2^{(i)}(x - x_i)^2 + a_3^{(i)}(x - x_i)^3, \quad i = 1, \ldots, n$$

und nehmen natürliche Randbedingungen an, also

$$p_1''(x_0) = p_n''(x_n) = 0$$

Dadurch ist der Spline wie eben gezeigt eindeutig festgelegt, aber:
Wie bestimmt man die lokalen Koeffizienten $a_j^{(i)}$*?*

Berechnung kubischer Splines

Satz 7.18 (Berechnung kubischer Splines)

Definiere die lokale Maschenweite $h_i := x_i - x_{i-1}$ und $a_2^{(0)} = a_2^{(n)} = 0$. Die Koeffizienten $a_2^{(i)}$ der gewählten Darstellung des kubischen Splines sind dann die Lösung des linearen Gleichungssystems

$$h_i a_2^{(i-1)} + 2(h_i + h_{i+1}) a_2^{(i)} + h_{i+1} a_2^{(i+1)} = 3 \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right)$$

der Dimension $n - 1$. Die restlichen Koeffizienten ergeben sich zu:

$$a_0^{(i)} = y_i, \quad a_1^{(i)} = \frac{y_i - y_{i-1}}{h_i} + \frac{h_i}{3}(2a_2^{(i)} + a_2^{(i-1)}), \quad a_3^{(i)} = \frac{a_2^{(i)} - a_2^{(i-1)}}{3h_i}$$

$\rightsquigarrow$ Beweis: Tafel

Lösung des Tridiagonalsystems

Zur Lösung des entstehenden Tridiagonalsystems:

- Die Gaußelimination / LR-Zerlegung hat in diesem Fall Aufwand $\mathcal{O}(n)$ (Bandstruktur)
- Das Gleichungssystem ist symmetrisch und strikt diagonaldominant:

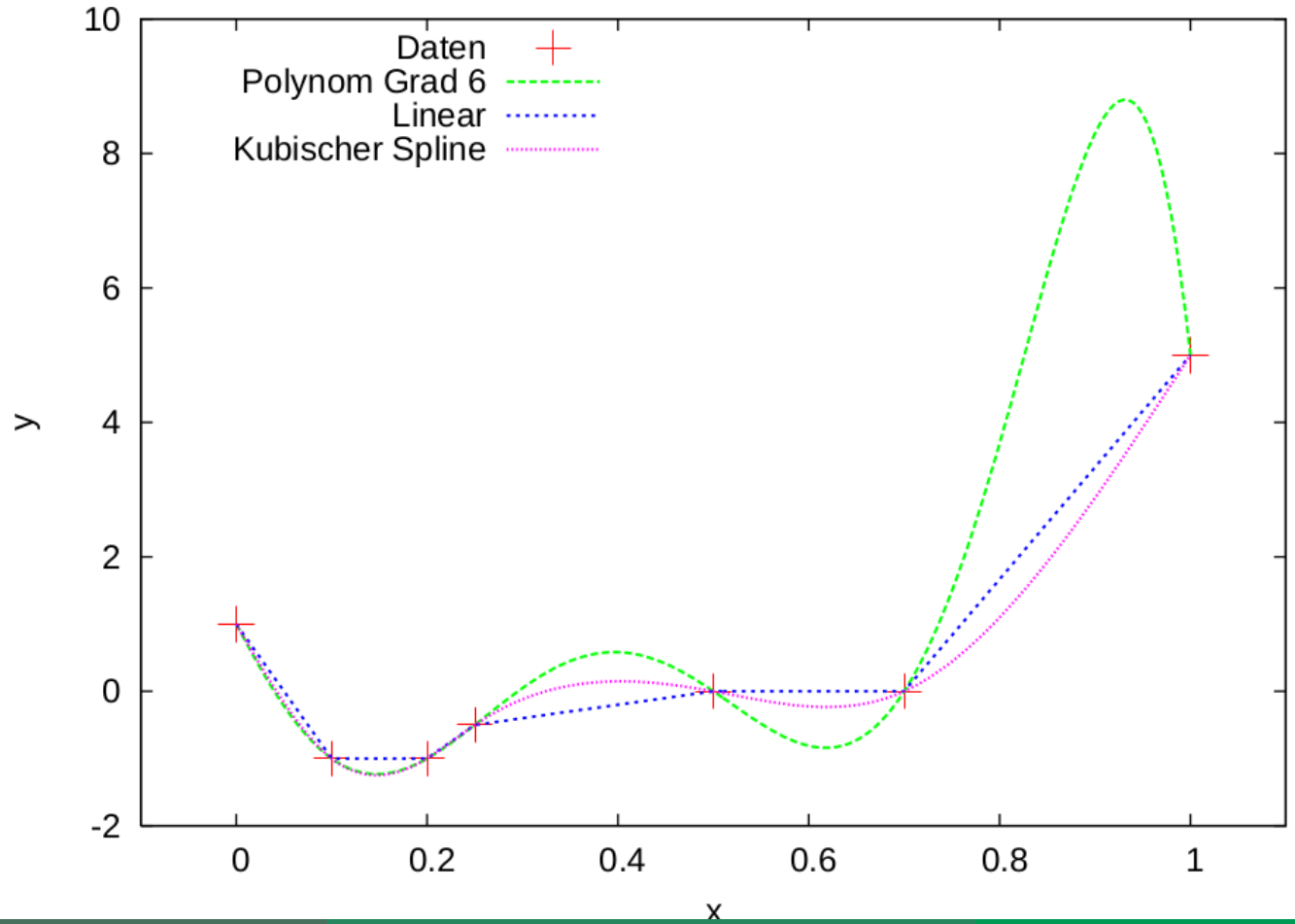
$$\sum_{j \neq i} |a_{ij}| < |a_{ii}|$$

Die Koeffizienten können also effizient und stabil ohne Pivotisierung berechnet werden.

Beispiel

Beispiel 7.19

Approximationseigenschaften der verschiedenen Ansätze:



Fehlerabschätzung

Satz 7.20 (Fehlerabschätzung für kubische Splines)

Sei $f \in C^4([a,b])$. Erfüllt der kubische Spline s zu f

$$s''(a) = f''(a) \quad \text{und} \quad s''(b) = f''(b)$$

(Hermitesche Randbedingungen), so gilt

$$\max_{a \le x \le b} |f(x) - s(x)| \le \frac{1}{2} h^4 \max_{a \le x \le b} |f^{(4)}(x)|$$

Selbst unter noch wesentlich schwächeren Voraussetzungen konvergiert die Spline-Interpolation gleichmäßig gegen f.

$\rightsquigarrow$ Beweis: siehe [Rannacher]

Beispiel

Beispiel 7.21

Interpolation der Funktion $\exp(-x^2)$ auf dem Intervall $[-10, 10]$ mit n äquidistanten Stützstellen, maximaler Punktfehler:

n	S^1	S^3
4	$6.0 \cdot 10^{-1}$	$7.4 \cdot 10^{-1}$
8	$3.0 \cdot 10^{-1}$	$3.9 \cdot 10^{-1}$
16	$1.1 \cdot 10^{-1}$	$2.8 \cdot 10^{-2}$
32	$6.9 \cdot 10^{-2}$	$7.1 \cdot 10^{-3}$
64	$2.2 \cdot 10^{-2}$	$3.3 \cdot 10^{-4}$
128	$6.0 \cdot 10^{-3}$	$1.9 \cdot 10^{-5}$
256	$1.5 \cdot 10^{-3}$	$1.2 \cdot 10^{-6}$
512	$3.8 \cdot 10^{-4}$	$7.3 \cdot 10^{-8}$

n	P_n
4	$8.0 \cdot 10^{-1}$
6	$1.0 \cdot 10^{0}$
8	$2.3 \cdot 10^{0}$
10	$5.9 \cdot 10^{0}$

Interpolation mit Splines konvergiert mit Ordnung h^2 für S^1 bzw. h^4 für S^3.

Trigonometrische Interpolation

Aufgabe:
Interpolation "periodischer" Funktionen, d.h. es gebe ein $\omega \in \mathbb{R}$, $\omega > 0$, so dass

$$f(x + \omega) = f(x) \quad \forall x \in \mathbb{R}$$

Es bietet sich die Interpolation mit "trigonometrischen Summen"

$$t_n(x) = \frac{a_0}{2} + \sum_{k=1}^{m} \left[a_k \cos(2\pi k \omega^{-1} x) + b_k \sin(2\pi k \omega^{-1} x) \right]$$

an, denn *jeder* der Summanden hat Periode ω.

Es gibt $2m + 1$ Parameter, also setze $n = 2m$ (konsistent mit Definition von n bei Polynominterpolation).

Trigonometrische Interpolation

Der Einfachheit halber betrachten wir nur $\omega = 2\pi$, also:

$$t_n(x) = \frac{a_0}{2} + \sum_{k=1}^{m} [a_k \cos(kx) + b_k \sin(kx)]$$

Die Interpolation führen wir dann mit den äquidistanten Stützstellen

$$x_j = \frac{2\pi j}{n+1} \quad j = 0, \ldots, n$$

durch. (Für den allgemeinen Fall transformiere man das Argument und die Stützstellen).

Trigonometrische Interpolation

Es stellt sich heraus, dass die Interpolationsaufgabe

$$t_n(x_j) = f(x_j) = y_j \quad j = 0, \ldots, n$$

zunächst im Körper $\mathbb{C}$ einfacher zu lösen ist, d.h. betrachte das *komplexe* trigonometrische Polynom

$$t_n^*(x) = \sum_{k=0}^{n} c_k e^{ikx}$$

mit $i = (-1)^{1/2}$ der imaginären Einheit und $c_k \in \mathbb{C}$ komplexen Koeffizienten. Für $\phi \in \mathbb{R}$ gilt die Eulersche Identität:

$$e^{i\phi} = \cos(\phi) + i \sin(\phi)$$

Komplexe Einheitswurzeln

Hilfssatz 7.22 (Komplexe Einheitswurzeln)

Die $(n+1)$-*ten komplexen Einheitswurzeln*

$$w_k := e^{ix_k} = e^{i\frac{2\pi k}{n+1}}$$

haben für alle $j, k \in \mathbb{Z}$ die folgenden Eigenschaften:

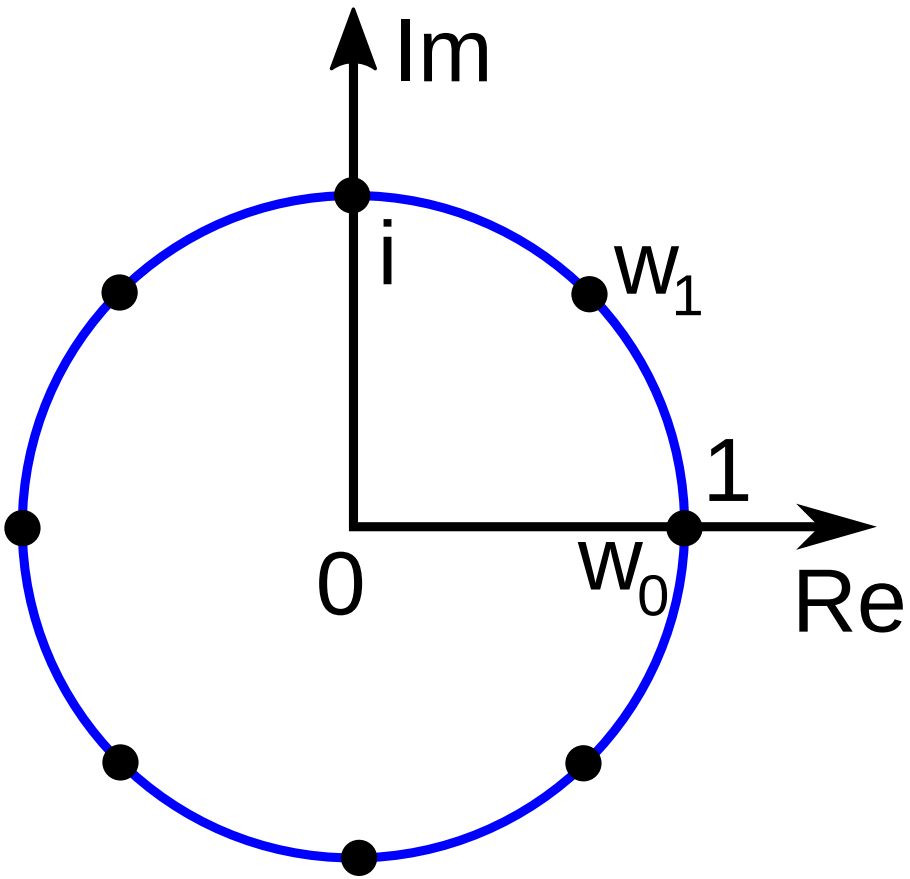
1. $w_k^{n+1} = 1$, $w_k^j = w_j^k$ und $w_k^{-j} = w_j^{-k}$
2. $w_k^j = w_k^{j \bmod (n+1)} = w_{k \bmod (n+1)}^j = w_{k \bmod (n+1)}^{j \bmod (n+1)}$
3. Es gilt

$$\sum_{j=0}^{n} w_k^j = \begin{cases} n+1 & k \bmod (n+1) = 0 \\ 0 & \text{sonst} \end{cases}$$

$\rightsquigarrow$ Beweis: Tafel

Komplexe Einheitswurzeln

Die komplexen Einheitswurzeln für $n = 7$ (also $n + 1 = 8$):



Komplexe Multiplikation ist eine *Drehstreckung*, und für Werte auf dem Einheitskreis entfällt die Streckung (der Faktor ist Eins).

Es handelt sich also um eine *Drehung*, und die Einheitswurzeln drehen jeweils um ein rationales Vielfaches von 360 Grad.

Komplexe Trigonometrische Interpolation

Satz 7.23 (Komplexe Trigonometrische Interpolation)

Zu gegebenen Zahlen $y_0, \ldots, y_n \in \mathbb{C}$ gibt es genau eine Funktion der Gestalt

$$t_n^*(x) = \sum_{k=0}^{n} c_k e^{ikx},$$

die den $n+1$ Interpolationsbedingungen $t_n^*(x_j) = y_j$ für $x_j = \frac{2\pi j}{n+1}$ entspricht. Die komplexen Koeffizienten sind bestimmt durch

$$c_k = (n+1)^{-1} \sum_{j=0}^{n} y_j e^{-ijx_k} = (n+1)^{-1} \sum_{j=0}^{n} y_j w_k^{-j} \quad k = 0, \ldots, n$$

$\rightsquigarrow$ Beweis: Tafel

Aufwand und reeller Fall

Mit der Formel aus Satz 7.23 ergibt sich ein Aufwand von $\mathcal{O}(n^2)$ für die Berechnung der Koeffizienten c_k.

Damit wurde auch die reelle Interpolationsaufgabe gelöst, da man in Satz 7.23 $y_j \in \mathbb{R}$ annehmen kann: aufgrund der Eulerschen Identität

$$e^{i\phi} = \cos(\phi) + i\sin(\phi)$$

ist dann auch eine passende reelle Funktion gegeben.

Wie berechnet man nun die reellen Koeffizienten a_k und b_k?

Diskrete Fourier-Analyse

Satz 7.24 (Diskrete Fourier-Analyse)

Für $n \in \mathbb{N}_0$ gibt es zu gegebenen reellen Zahlen $y_0, \ldots, y_n$ genau ein trigonometrisches Polynom der Form

$$t_n(x) = \frac{a_0}{2} + \sum_{k=1}^{m} [a_k \cos(kx) + b_k \sin(kx)] + \frac{\theta}{2} a_{m+1} \cos((m+1)x)$$

das die Interpolationsaufgabe löst, wobei $\theta = 0$ für n gerade, und $\theta = 1$ und für n ungerade. Die Koeffizienten sind

$$a_k = 2(n+1)^{-1} \sum_{j=0}^{n} y_j \cos(jx_k), \quad b_k = 2(n+1)^{-1} \sum_{j=0}^{n} y_j \sin(jx_k)$$

$\rightsquigarrow$ Beweisskizze: Tafel, genauer: [Rannacher]

Diskrete Fouriertransformation (DFT)

Die *schnelle Fouriertransformation* (FFT):
Einer der berühmtesten Algorithmen der angewandten Mathematik und Informatik! Entwickelt 1965 von James Cooley und John Tukey.

Ergebnis der bisherigen Überlegungen sind zwei Transformationen:

$$c_k = N^{-1} \sum_{j=0}^{N-1} y_j e^{-i\frac{2\pi jk}{N}} \quad \text{(Hintransformation)}$$

$$y_j = \sum_{k=0}^{N-1} c_k e^{i\frac{2\pi jk}{N}} \quad \text{(Rücktransformation)}$$

mit $N := n + 1$. Diese beiden Gleichungen bezeichnet man als *diskrete Fouriertransformation* (DFT).

Diskrete Fouriertransformation (DFT)

Setze

$$c := (c_0, \ldots, c_{N-1})^T, \quad y := (y_0, \ldots, y_{N-1})^T,$$

dann ist die diskrete Fouriertransformation äquivalent zu den Gleichungssystemen

$$c = N^{-1} W y, \quad y = U c \quad \text{mit}$$

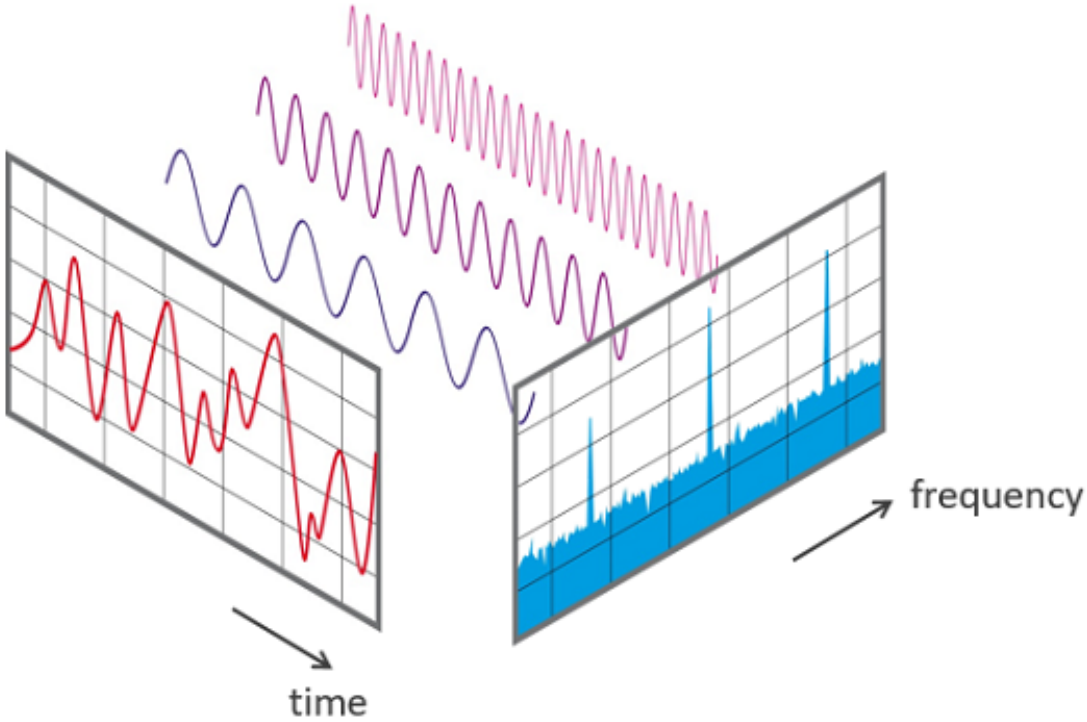
$$(W)_{k,j} = e^{-i\frac{2\pi jk}{N}} = w_k^{-j} \quad \text{und} \quad (U)_{j,k} = w_j^k$$

Es gilt hier also $W^{-1} = N^{-1} U$, so dass nur eine der Matrizen aufgestellt werden muss. Da sie beide dicht besetzt sind, ist der Aufwand für Hin- und Rücktransformation jeweils $\mathcal{O}(N^2)$.

$\rightsquigarrow$ Rechnung: Tafel

Diskrete Fouriertransformation (DFT)

Grundidee:



Quelle: Phonical, wikipedia.org, cc-by-sa

Die DFT zerlegt ein (periodisch fortgesetztes) diskretes Signal in sein Frequenzspektrum.

Schnelle Fouriertransformation (FFT)

Falls N gerade ist, kann man die DFT der Länge N durch zwei DFT der Länge $N/2$ berechnen (Teile-und-herrsche-Verfahren, *divide and conquer*).

Setze $\tilde{c}_k := Nc_k$ (d.h. ignoriere Vorfaktor), dann gilt

$$\tilde{c}_k = \begin{cases} \tilde{c}^g_k + e^{-i\frac{2\pi k}{N}} \tilde{c}^u_k & 0 \le k < N/2 \\ \tilde{c}^g_{k-N/2} + e^{-i\frac{2\pi k}{N}} \tilde{c}^u_{k-N/2} & N/2 \le k < N \end{cases}$$

mit den geraden und ungeraden Anteilen

$$\tilde{c}^g_k := \sum_{j=0}^{N/2-1} y_{2j} e^{-i\frac{2\pi jk}{N/2}}, \quad \tilde{c}^u_k := \sum_{j=0}^{N/2-1} y_{2j+1} e^{-i\frac{2\pi jk}{N/2}}$$

$\rightsquigarrow$ Rechnung: Tafel

Schnelle Fouriertransformation (FFT)

Wenn N eine Zweierpotenz ist, $N = 2^d$, lässt sich die DFT somit rekursiv auf den Fall der Länge 2 zurückführen. Dieser Fall lässt sich sehr einfach direkt lösen.

Im Gegensatz zum direkten Ansatz mittels Matrixmultiplikation ergibt sich damit der Aufwand

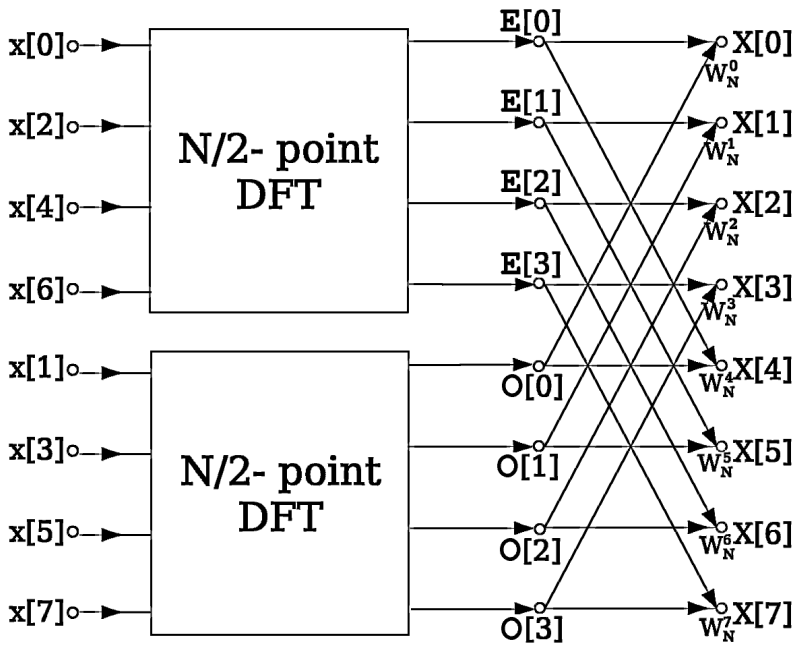
$$T_{\mathrm{FFT}}(N) = 2 \cdot T_{\mathrm{FFT}}(N/2) + T_{\mathrm{merge}}(N) = \mathcal{O}(N \cdot \log(N)),$$

also deutlich schneller für große N. Dies ist ein Spezialfall des *Master-Theorems* für rekursiv definierte Funktionen.

$\rightsquigarrow$ Rechnung: Tafel

Beispiel

Beispiel der rekursiven Zerlegung für $N = 8$:



Quelle: Virens, wikipedia.org, cc-by-sa

Berechnung der DCT für $N/2 = 4$ für die geraden bzw. ungeraden Indizes, danach Rekonstruktion der Ausgabe durch Zusammenführen von je zwei Zwischenergebnissen.

Anwendungen

Wichtige Anwendungen der FFT sind:

- Spektralanalyse (Zerlegung eines Signals in Frequenzen)
- Datenkompression (Vernachlässigung hochfrequenter Anteile)
- Lösen partieller Differentialgleichungen (Spektralmethoden)

Benutzt wird dies unter anderem für:

- Telekommunikation (z.B. WLAN, LTE-Mobilfunk)
- Multimedia (Datenformate wie JPEG, MPEG, MP3)
- Medizin (Kernspin- und Computertomographie)
- Forschung (z.B. digitale Oszilloskope)

Prähilberträume

Wir betrachten nun die Approximation von Funktionen in Prähilberträumen:

Definition 7.25 (Prähilbertraum)

Ein Vektorraum von Funktionen über $\mathbb{R}$ oder $\mathbb{C}$ mit Skalarprodukt heißt *Prähilbertraum*.

Im folgenden sei H ein Prähilbertraum und $S \subset H$ ein *endlichdimensionaler* Teilraum von H.

Beispiele I

Beispiel 7.26

Beispiele für Prähilberträume:

- Raum der stetigen Funktionen $C^0([a,b])$ mit Skalarprodukt

$$(f,g) = \int_a^b f(x)\overline{g(x)}\,dx$$

- Gegeben $\Psi = \{\psi_1, \ldots, \psi_N\}, \psi_i \in C^0([a,b])$, ist

$$S = \operatorname{span}(\Psi) = \{f \mid f = \sum_{i=1}^{N} c_i \psi_i\}$$

ein endlichdimensionaler Prähilbertraum. Analog für andere Prähilberträume statt C^0.

Beispiele II

Beispiel 7.26

- Raum der quadratintegrierbaren Funktionen

$$L^2((a,b)) = \{f \mid \int_a^b |f(t)|^2 \, dt < \infty\}$$

Dabei ist mit $\int_a^b$ das "Lebesgue-Integral" gemeint. Im Gegensatz zu $C^0([a,b])$ ist $L^2((a,b))$ vollständig bzgl. der Norm

$$\|f\| = (f,f)^{1/2}$$

und damit sogar ein Hilbertraum. $L^2((a,b))$ ist das Analogon des $\mathbb{R}^n$ für Funktionenräume.

Allgemeine Gauß-Approximation

Wir betrachten die folgende Aufgabe:
Zu $f \in H$ finde $g \in S$, so dass

$$\|f - g\| \to \min$$

wobei $\|f\| = (f, f)^{1/2}$ die durch das Skalarprodukt induzierte Norm bezeichne.

Satz 7.27 (Allgemeine Gauß-Approximation)

Die obige Aufgabe hat genau eine Lösung $g \in S$. Diese ist charakterisiert durch

$$(g, \phi) = (f, \phi) \quad \forall \phi \in S$$

$\rightsquigarrow$ Beweis: Tafel

Berechnung der Approximation

Wähle Basis $\Psi = \{\psi_1, \ldots, \psi_N\}$ des endlichdimensionalen Raums S, $N = \dim(S)$. Stelle damit die sogenannte *Massenmatrix (Gramsche Matrix)* A und rechte Seite b mit

$$[A]_{ij} = (\psi_i, \psi_j), \quad [b]_i = (f, \psi_i)$$

auf. A ist symmetrisch positiv definit bzw. hermitesch. Bestimme die gesuchten Koeffizienten α_i durch Lösen des linearen Gleichungssystems

$$A\alpha = b.$$

Approximation mit Orthonormalbasen

Besonders einfach wird die Lösung der Approximationsaufgabe, wenn Ψ eine Orthonormalbasis ist, d.h. $(\psi_i, \psi_j) = \delta_{ij}$. Dann gilt

$$\sum_{j=1}^{N} \alpha_j (\psi_j, \psi_i) = \alpha_i = (f, \psi_i) \quad \forall i = 1, \ldots, N$$

und somit

$$g = \sum_{j=1}^{N} \alpha_j \psi_j = \sum_{j=1}^{N} (f, \psi_j) \psi_j$$

Beispiel I

Beispiel 7.28 (Fourierreihe)

Für $N = 2m + d$, $m \in \mathbb{N}$ ist

$$\Psi_F = \pi^{-1/2} \cdot \{2^{-1/2}, \cos(x), \ldots, \cos(mx), \sin(x), \ldots, \sin(mx)\}$$

eine Orthonormalbasis von Funktionen auf dem Interval $[-\pi, \pi]$. Damit gilt dann

$$g(x) = \frac{a_0}{2} + \sum_{k=1}^{m} [a_k \cos(kx) + b_k \sin(kx)]$$

$$\text{mit} \quad a_k = \int_{-\pi}^{\pi} f(x) \cos(kx)\, dx, \quad b_k = \int_{-\pi}^{\pi} f(x) \sin(kx)\, dx$$

Beispiel II

Beispiel 7.28 (Fourierreihe)

Vorsicht: die a_k und b_k sind nicht die α_i von oben, sondern unterscheiden sich davon um einen Faktor $\pi^{1/2}$!

Für unendlich viele Glieder nennt man die entstehende Reihe, also

$$g(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [a_k \cos(kx) + b_k \sin(kx)],$$

Fourier-Reihe. Diese konvergiert gegen ein Element aus $L^2((-\pi, \pi))$.

Die DFT lässt sich als näherungsweie Auswertung der Integrale mittels "Trapezregel" verstehen.

Fehlerkontrolle

Bisher war $S \subset H$ fest gewählt, damit wurde das "optimale" g bestimmt. Der Fehler $\|f - g\|$ wurde einfach akzeptiert.

Daher nun Verfeinerung der Approximationsaufgabe:
Finde $S \subset H$ mit $\dim(S)$ möglichst klein, so dass

$$\|f - g\| < \mathrm{TOL},$$

wobei TOL eine vorgegebene Zahl sei.

Ein Spezialfall ist die "Kompression": hier ist H selbst bereits endlichdimensional. Man versucht, das "Signal" f möglichst platzsparend zu speichern.

Fehlerkontrolle

Mittels Orthonormalbasen lässt sich der Fehler recht einfach messen:

Sei $S_N \subset H$, $N \in \mathbb{N}$, eine Folge von Approximationsräumen mit $\dim(S_N) = N$ (oft gilt sogar $S_N \subset S_{N+1}$), und g_N die Bestapproximation von f in S_N. Sei zusätzlich Ψ_N eine Orthonormalbasis von S_N (oft mit $\Psi_N \subset \Psi_{N+1}$).

Dann gilt

$$0 \leq \|f - g_N\|^2 = (f, f) - \sum_{i=1}^{N} (f, \psi_i)^2$$

$\rightsquigarrow$ Rechnung: Tafel

Fehlerkontrolle

Wir möchten

$$\|f - g_N\|^2 = (f, f) - \sum_{i=1}^{N} (f, \psi_i)^2 \leq \text{TOL} \cdot (f, f),$$

und das ist äquivalent zu

$$\sum_{i=1}^{N} (f, \psi_i)^2 \geq (1 - \text{TOL}) \cdot (f, f) \tag{7.1}$$

(Frage: "Welche Anteile von f werden von S_N bereits repräsentiert?")

Achtung: hier wurde eine *relative* Toleranz TOL gewählt, diese lässt sich jedoch in eine absolute umrechnen, und umgekehrt.

Anmerkungen

Bemerkung 7.29

- Der Wert (f, f) wird als (zumindest hinreichend genau) berechenbar angenommen. Dafür kann man z.B. numerische Integration verwenden.
- Falls $\Psi_N \subset \Psi_{N+1}$, sind einfach weitere Basisfunktionen hinzuzufügen bis (7.1) erreicht ist. Das gilt zum Beispiel für die Fourierreihe, Bsp. 7.28.
- Bei $\Psi_N \subset \Psi_{N+1}$ folgt aus der Fehlerdarstellung unmittelbar

$$\|f - g_{N+1}\|^2 \leq \|f - g_N\|^2,$$

der Fehler ist also monoton fallend!

$\rightsquigarrow$ Rechnung: Tafel

Approximation mit Haar-Wavelets

Es bezeichne $\mathrm{tr}(f) = \{x \mid f(x) \neq 0\}$ den *Träger* einer Funktion. Bei der ONB aus Sinus- und Cosinus-Funktionen haben alle ψ_i im wesentlichen *globalen* Träger, d.h. $\overline{\mathrm{tr}(f)} = [a, b]$.

Oft variiert die zu approximierende Funktion aber nur *lokal* sehr stark. In diesem Fall möchte man ein Funktionensystem mit folgenden Eigenschaften:

- $(\psi_i, \psi_j) = \delta_{ij}$ (Orthogonalität)
- $\Psi_N \subset \Psi_{N+1}$ (Geschachteltheit)
- $\mathrm{diam}(\mathrm{tr}(\psi_i)) \to 0$ für $i \to \infty$

Diese Eigenschaften besitzten sogenannte *Waveletfunktionen*, von denen wir das sogenannte *Haar-Wavelet* untersuchen.

Approximation mit Haar-Wavelets

Definition 7.30

Wir definieren das "mother wavelet" $\psi(x)$ und die Abschneidefunktion $\chi(x)$ auf $(0, 1]$:

$$\psi(x) := \begin{cases} 1 & 0 < x \leq 1 \\ -1 & 1 < x \leq 2 \\ 0 & \text{sonst} \end{cases}, \qquad \chi(x) := \begin{cases} 1 & 0 < x \leq 1 \\ 0 & \text{sonst} \end{cases}$$

Für $l \in \mathbb{N}_0$ (Stufe) und $0 \leq i < 2^{l-1}$ (Index) ist das *Haar-Wavelet*

$$\psi_i^l := \max(2^{\frac{l-1}{2}}, 1) \cdot \psi(2^l x - 2i) \cdot \chi(x).$$

Die *Waveletbasis* der Stufe l ist $\Psi^l := \{\psi_0^0\} \cup \bigcup_{j=1}^{l} \bigcup_{i=0}^{2^{j-1}-1} \{\psi_i^j\}$.

Eigenschaften der Wavelets

Wir fassen einige Eigenschaften der Haar-Wavelets zusammen:

- Für $l > 0$ gilt

$$\psi_i^l = \begin{cases} 0 & x \leq \frac{i}{2^{l-1}} \vee x > \frac{i+1}{2^{l-1}} \\ 2^{\frac{l-1}{2}} & \frac{i}{2^{l-1}} < x \leq \frac{i+\frac{1}{2}}{2^{l-1}} \\ -2^{\frac{l-1}{2}} & \frac{i+\frac{1}{2}}{2^{l-1}} < x \leq \frac{i+1}{2^{l-1}} \end{cases}$$

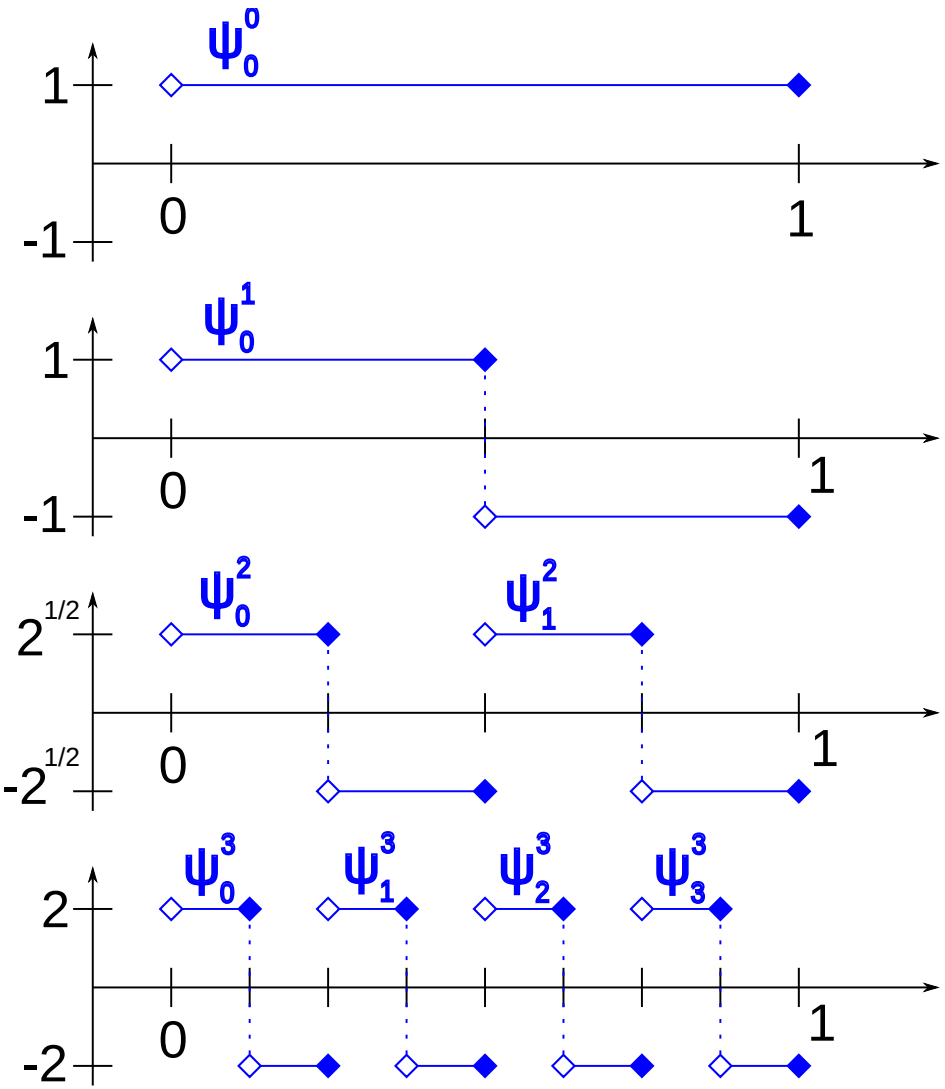
- Speziell für $l = 1$ gibt es ein einzelnes Wavelet

$$\psi_0^1 = \max(1,1) \cdot \psi(2x) \cdot \chi(x) = \psi(2x)$$

- Und speziell für $l = 0$ gibt es ein einzelnes Wavelet

$$\psi_0^0 = \max(2^{-1/2},1) \cdot \psi(x) \cdot \chi(x) = \chi(x)$$

Graphische Darstellung



Darstellung der Haar-Wavelets für $l = 0, \ldots, 3$.

Für $l = 0$ und $l = 1$ gibt es jeweils nur ein Wavelet (konstante Einsfunktion bzw. gestauchtes "mother wavelet"). Die Wavelets aller weiteren Level sind Kopien des Wavelets ψ_0^1, die in x-Richtung gestaucht und zur Kompensation in ihrem Wert gestreckt wurden.

Weitere Eigenschaften

Weiterhin gilt:

- Die Wavelets bilden eine Baumstruktur bzgl. ihrer Träger:

$$\mathrm{tr}(\psi_i^l) = \mathrm{tr}(\psi_{2i}^{l+1}) \cup \mathrm{tr}(\psi_{2i+1}^{l+1}), \quad \mathrm{tr}(\psi_i^l) \cap \mathrm{tr}(\psi_j^l) = \emptyset \text{ für } i \neq j$$

- Die Wavelets sind paarweise orthogonal:

$$(\psi_i^l, \psi_j^k) = \begin{cases} 1 & i = j \wedge l = k \\ 0 & \text{sonst} \end{cases}$$

- Offensichtlich erfüllt die Konstruktion $\Psi^l \subset \Psi^{l+1}$, da immer nur Basisfunktionen hinzugenommen werden.

Damit sind alle geforderten Eigenschaften erfüllt.

$\rightsquigarrow$ Beweis: Tafel

Aufgespannter Raum

Um die durch Haar-Wavelets darstellbaren Funktionen etwas anschaulicher zu machen, definieren wir die Funktionen

$$\phi_i^l(x) := \begin{cases} 1 & \frac{i}{2^l} < x \leq \frac{i+1}{2^l} \\ 0 & \text{sonst} \end{cases}$$

für $l \in \mathbb{N}_0$ und $0 \leq i < 2^l$, sowie $\Phi^l := \bigcup_{i=0}^{2^l-1} \{\phi_i^l\}$.

$S^l := \text{span}(\Phi^l)$ ist der Raum der stückweise konstanten Funktionen auf dem Intervall $(0, 1]$ bzgl. der äquidistanten Unterteilung in 2^l Teilintervalle. Es gilt

$$\text{span}(\Psi^l) = \text{span}(\Phi^l) = S^l.$$

$\rightsquigarrow$ Beweisskizze: Tafel

Darstellung von Funktionen

Da die Haar-Wavelets eine Orthonormalbasis bilden, ist die Approximationsaufgabe aus Satz 7.27 analog zur Fourier-Reihe durch Skalarprodukte der Form (f, ψ_i^j) berechenbar:

$$g(x) = \sum_{j=0}^{l} \sum_{i=0}^{2^j-1} (f, \psi_i^j) \cdot \psi_i^j$$

Im Unterschied zur Fourier-Reihe lassen sich die gesuchten Koeffizienten aus den Mittelwerten von f auf den Trägern der ψ_i^j berechnen, was eine stark lokalisierte und zugleich durch Rekursion besonders effiziente Berechnung ermöglicht.

Datenkompression mit Wavelets

Anwendung von (7.1), der Formel für die Fehlerkontrolle:

Gegeben: $f \in S^l$ in Basis Ψ^l
Gesucht: Teilraum $\widetilde{S} \subset S^l$, so dass $\|f - \tilde{f}\| \leq \text{TOL} \cdot (f, f)$ und $\widetilde{S}$ möglichst klein.

Algorithmus:

- $I \leftarrow \{\psi_0^0\}$
- $S \leftarrow (f, \psi_0^0)^2$
- while $(S < (1 - \text{TOL}) \cdot (f, f))$ do:
  - wähle Kind ψ_k^{l+1} eines $\psi_i^l \in I$ mit (f, ψ_k^{l+1}) maximal
  - $I \leftarrow I \cup \{\psi_k^{l+1}\}$
  - $S \leftarrow S + (f, \psi_k^{l+1})^2$

Ausblick

Eine leichte Abänderung der Gaußschen Approximationsaufgabe:

Finde $u \in S$ (endlich-dimensionaler Funktionenraum), so dass

$$\int_\Omega \nabla u \cdot \nabla \phi = \int_\Omega f\phi \quad \forall \phi \in S$$

Dies führt auf die (numerische) Lösung der "Poisson-Gleichung"

$$-\Delta u = -\sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} = f \quad \text{in } \Omega \subset \mathbb{R}^d$$

(vgl. das Einführungskapitel über lineare Gleichungssysteme)

Abschnitt 8

8 Numerische Integration
Einführung
Newton-Cotes-Formeln
Summierte Quadraturformeln
Quadraturen höherer Ordnung
Ausblick

Numerische Integration (Quadratur)

Wir behandeln die numerische Berechnung bestimmter Integrale in einer Raumdimension:

$$I_{[a,b]}(f) = \int_a^b f(x)\,dx$$

Alle von uns behandelten Verfahren führen auf die Form

$$I^{(n)}_{[a,b]}(f) = \sum_{i=0}^{n} w_i f(x_i) + \text{Fehler},$$

wobei $w_i \in \mathbb{R}$ die *Gewichte* und $x_i \in \mathbb{R}$ die *Stützstellen* der Integrationsformel sind.

Gründe für Numerische Integration

Es gibt mehrere Gründe, warum die Anwendung von numerischer Integration (numerischer Quadratur) sinnvoll bzw. notwendig sein kann:

- Oft ist eine Funktion f nur an endlich vielen Punkten bekannt, z.B. da sie aus Messreihen stammt. Häufig ist man neben der Darstellung der Funktion, z.B. mittels Least Squares oder Splines, auch an ihrer Ableitung oder ihrem Integral interessiert.
- Es gibt Funktionen, z.B. $f(x) = \exp(-x^2)$, deren Integral wichtig ist, aber es steht keine geschlossene Formel für dessen Berechnung zur Verfügung.

Newton-Cotes-Formeln

Die *Newton-Cotes-Formeln* sind sog. interpolatorische Quadraturformeln.

Idee: Stelle das Interpolationspolynom p zu gewissen Stützstellen auf und berechne das Integral von p exakt.

Formal: Stützstellen und Werte $(x_i, f(x_i)), i = 0, \ldots, n$, Lagrange-Darstellung:

$$p_n(x) = \sum_{i=0}^{n} f(x_i) L_i^{(n)}(x), \quad L_i^{(n)}(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

und somit

$$I_{[a,b]}(f) \approx I_{[a,b]}^{(n)}(f) = \int_a^b p_n(x)\, dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_i^{(n)}(x)\, dx$$

Ordnung einer Quadraturformel

Definition 8.1 (Ordnung einer Quadraturformel)

Eine Quadraturformel $I^{(n)}(f)$ hat mindestens die Ordnung m, wenn sie Polynome vom Grad $m-1$ exakt integriert.

Eine Formel zweiter Ordnung integriert beispielsweise lineare Funktionen exakt.

Die Newton-Cotes-Formeln benutzen Polynominterpolation und haben daher für $n+1$ Stützstellen mindestens Ordnung $n+1$. Wir werden aber später sehen, dass das noch besser geht.

Newton-Cotes-Formeln

Die Newton-Cotes-Formeln nutzen *äquidistante* Stützstellen. Es gibt davon zwei Varianten:

Abgeschlossene Formeln:
Die Intervallgrenzen a und b sind Stützstellen, es gilt

$$x_i = a + iH, \quad i = 0, \ldots, n, \quad \text{mit } H = \frac{b-a}{n}$$

Offene Formeln:
Die Grenzen a und b sind keine Stützstellen, und es gilt

$$x_i = a + (i+1)H, \quad i = 0, \ldots, n, \quad \text{mit } H = \frac{b-a}{n+2}$$

Newton-Cotes-Formeln

Generell muss für die Summe der Gewichte einer Quadraturformel

$$\sum_{i=0}^{n} w_i = \sum_{i=0}^{n} w_i \cdot 1 = \int_a^b 1 \, dx = b - a$$

gelten. Um Gewichte zu erhalten, die unabhängig vom gewählten Intervall sind, wird dieser Term ausgeklammert, und man erhält für die geschlossenen Newton-Cotes-Formeln

$$I_{[a,b]}^{(n)}(f) = (b-a) \cdot \sum_{i=0}^{n} \tilde{w}_i f(x_i)$$

mit

$$\tilde{w}_i = (b-a)^{-1} \int_a^b L_i^{(n)} \, dx = n^{-1} \int_0^n \prod_{j \neq i} \frac{s-j}{i-j} \, ds$$

$\rightsquigarrow$ Beweis: Tafel

Beispiele

Abgeschlossene Formeln für $n = 1, 2, 3$ und $H = (b - a)/n$ sind:

Die Trapezregel bzw. Sehnen-Trapezregel:

$$I^{(1)}(f) = \frac{b-a}{2} \cdot [f(a) + f(b)]$$

Die Simpsonregel bzw. Keplersche Fassregel:

$$I^{(2)}(f) = \frac{b-a}{6} \cdot \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Die 3/8-Regel bzw. Pulcherrima:

$$I^{(3)}(f) = \frac{b-a}{8} \cdot [f(a) + 3f(a+H) + 3f(b-H) + f(b)]$$

Beispiele

Offene Formeln für $n = 0, 1, 2$ und $H = (b - a)/(n + 2)$ sind:

Die Mittelpunktregel, Tangenten-Trapezregel bzw. Rechteckregel:

$$I^{(0)}(f) = (b - a) \cdot f\left(\frac{a + b}{2}\right)$$

Die zweite offene Regel (kein besonderer Name):

$$I^{(1)}(f) = \frac{b - a}{2} \cdot [f(a + H) + f(b - H)]$$

Die dritte offene Regel (ebenfalls kein Name):

$$I^{(2)}(f) = \frac{b - a}{3} \cdot [2f(a) - f\left(\frac{a + b}{2}\right) + 2f(b - H)]$$

Anmerkungen

Bemerkung 8.2

Ab $n = 7$ für die abgeschlossenen bzw. $n = 2$ für die offenen Formeln treten negative Gewichte auf. Das ist ungünstig, denn:

- Für strikt nicht-negative Funktionen f ist kann $I^{(n)}(f) < 0$ sein (Stoffkonzentration, Masseerhaltung...).
- Es herrscht erhöhte Gefahr der Auslöschung.
- Die Kondition kann schlechter werden, während sie für rein positive Gewichte beschränkt ist.

$\rightsquigarrow$ Rechnung: Tafel

Restgliedabschätzung I

Satz 8.3 (Restglieder)

Den begangenen Fehler kann man folgendermaßen abschätzen:

1. Trapezregel: $n = 1$, Ordnung 2, Es gilt

$$I(f) - \frac{b-a}{2} \cdot [f(a) + f(b)] = -\frac{(b-a)^3}{12} f''(\xi), \quad \xi \in [a, b]$$

für $f \in C^2([a, b])$. Es werden also Polynome bis zum Grad 1 exakt integriert, denn für diese ist $f''(x) = 0$ auf $[a, b]$.

Allgemein gilt: die Ordnung der ungeraden Formeln entspricht der Anzahl an Stützstellen, während die Ordnung der geraden Formeln um Eins höher ist.

Restgliedabschätzung II

Satz 8.3 (Restglieder)

2. Simpsonregel: n = 2, Ordnung 4, für $f \in C^4([a,b])$ gilt

$$I(f) - \frac{b-a}{6} \cdot \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] = -\frac{(b-a)^5}{2880} f^{(4)}(\xi)$$

3. Mittelpunktregel: n = 0, Ordnung 2, für $f \in C^2([a,b])$ gilt

$$I(f) - (b-a) \cdot f\left(\frac{a+b}{2}\right) = \frac{(b-a)^3}{24} f''(\xi)$$

also "halber Fehler der Trapezregel" bei nur einer Auswertung!

$\rightsquigarrow$ Beweis: Tafel

Summierte Quadraturformeln

Das Erhöhen des Polynomgrads ist wenig sinnvoll, da

- früh negative Gewichte auftreten
- die Lagrange-Interpolation mit äquidistanten Stützstellen nicht punktweise konvergiert
- man entsprechende Differenzierbarkeit von f für die Abschätzung benötigt

Idee der *summierten Quadraturformeln*:

- Unterteile das Interval $[a, b]$ in N Teilintervalle

$$[x_i, x_{i+1}], \quad x_i = a + ih, \ i = 0, \ldots, N-1, \ h = \frac{b-a}{N}$$

- Wende eine der obigen Formeln auf jedem Teilintervall an und summiere die Ergebnisse.

Beispiele

Beispielsweise ergeben sich bei N Teilintervallen der Schrittweite h:

Die summierte Trapezregel:

$$I_h^{(1)}(f) = \sum_{i=0}^{N-1} \frac{h}{2}[f(x_i) + f(x_{i+1})] = h \cdot \left[\frac{f(a)}{2} + \sum_{i=1}^{N-1} f(x_i) + \frac{f(b)}{2} \right]$$

Die summierte Simpsonregel:

$$I_h^{(2)}(f) = \frac{h}{3} \cdot \left[\frac{f(a)}{2} + \sum_{i=1}^{N-1} f(x_i) + 2 \sum_{i=1}^{N-1} f\left(\frac{x_i + x_{i+1}}{2} \right) + \frac{f(b)}{2} \right]$$

Die summierte Mittelpunktregel:

$$I_h^{(0)}(f) = h \cdot \sum_{i=0}^{N-1} f\left(\frac{x_i + x_{i+1}}{2} \right)$$

Restgliedabschätzung

Satz 8.4 (Restglieder für summierte Quadraturen)

Für die je Teilintervall verwendete Quadraturformel gelte

$$I_{[x_i,x_{i+1}]}(f) - I^{(n)}_{[x_i,x_{i+1}]}(f) = \alpha_n h^{m+1} f^{(m)}(\xi_i), \quad \xi_i \in [x_i, x_{i+1}].$$

Dann gilt für die summierte Quadraturformel

$$I_{[a,b]}(f) - I^{(n)}_h(f) = \alpha_n (b-a) h^m f^{(m)}(\xi), \quad \xi \in [a, b].$$

$\rightsquigarrow$ Beweis: Tafel

Restgliedabschätzung

Wie wir sehen, bleibt bei der Summation die Ordnung der Formel erhalten (da in der Abschätzung die gleiche Ableitung auftritt).

Konkret bedeutet das:

- Die Trapezregel und Mittelpunktregel haben Ordnung 2, also haben es die summierten Regeln ebenfalls. Es gilt

$$|I_{[a,b]}(f) - I_h^{(1 \text{ bzw. } 0)}(f)| = |\frac{b-a}{\alpha} \max_{[a,b]} f^{(2)}| \cdot h^2 \in \mathcal{O}(h^2),$$

  mit $\alpha = 12$ bzw. 24, falls die zweite Ableitung beschränkt ist.
- Die Simpsonregel hat Ordnung 4, also gilt das auch für die summierte Simpsonregel, und

$$|I_{[a,b]}(f) - I_h^{(2)}(f)| = |\frac{b-a}{2880} \max_{[a,b]} f^{(4)}| \cdot h^4 \in \mathcal{O}(h^4),$$

  falls die vierte Ableitung beschränkt ist.

Fehlerkontrolle

Es gelten

$$I_h^{(2)}(f) = \frac{1}{3} I_h^{(1)}(f) + \frac{2}{3} I_h^{(0)}$$

sowie für die halbierte Schrittweite $h/2$

$$I_{h/2}^{(1)}(f) = \frac{1}{2} I_h^{(1)}(f) + \frac{1}{2} I_h^{(0)}(f).$$

Diese Zusammenhänge kann man zur Fehlerkontrolle gemäß der Form

$$|I(f) - I_h(f)| \leq \mathrm{TOL}$$

verwenden.

Quadraturen höherer Ordnung

Wie wir gesehen haben, sind die Newton-Cotes-Formeln nur bis $n = 7$ bzw. $n = 2$ brauchbar.

Die summierten Quadraturformeln erlauben zwar beliebig feine Berechnungen, haben aber immer die selbe Ordnung wie die zugrundeliegenden Formeln.

Wie erreicht man eine höhere Ordnung?

Zwei Ansätze:

- Extrapolation auf summierte Formeln anwenden
- Nicht-äquidistante Stützstellen wählen (Gauß-Quadratur)

Romberg-Integration

Bei den summierten Formeln ist man an $\lim_{h\to 0} I_h^{(n)}(f)$ interessiert. Dafür ist Extrapolation zum Limes anwendbar.

Herzstück des Ansatzes ist die Euler-Maclaurinsche Summenformel:

$$I(f) - I_h^{(1)}(f) = \sum_{k=1}^{m-1} h^{2k} \frac{B_{2k}}{(2k)!}(f^{(2k-1)}(b) - f^{(2k-1)}(a)) + h^{2m} \frac{B_{2m}}{(2m)!}(b-a) f^{(2m)}(\xi), \quad \xi \in [a,b]$$

Speziell die Trapezsumme hat also eine Fehlerentwicklung in geraden Potenzen, was Extrapolation besonders effizient macht. Die in der Formel auftretenden Konstanten B_i sind die sogenannten Bernoulli-Zahlen.

Gauß-Integration

Frage: *Lässt sich durch nicht-äquidistante Stützstellen die Genauigkeit von Quadraturformeln verbessern?*

Idee: Wähle x_i und w_i so, dass Polynome von möglichst hohem Grad exakt integriert werden (Ordnungsmaximierung).

Satz 8.5

Die maximal erreichbare Ordnung einer Quadraturformel mit $n+1$ Stützstellen ist $2n+2$ (d.h. Polynome vom Grad $2n+1$ werden exakt integriert).

$\rightsquigarrow$ Beweis: Tafel

Gauß-Integration

Satz 8.6

Es gibt genau eine interpolatorische Quadraturformel zu $n+1$ paarweise versch. Stützstellen in $[-1,1]$ mit der Ordnung $2n+2$. Ihre Stützstellen sind die Nullstellen $\lambda_0, \ldots, \lambda_n \in (-1,1)$ des $(n+1)$-ten Legendre-Polynoms L_{n+1}, wobei

$$L_0(x) = 1, \quad L_1(x) = x, \quad L_{k+1}(x) = \frac{2k+1}{k+1} L_k(x) - \frac{k}{k+1} L_{k-1}(x)$$

Die Gewichte erhält man mittels

$$w_i = \int_{-1}^{1} \prod_{j \neq i} \left(\frac{x - \lambda_j}{\lambda_i - \lambda_j} \right)^2 > 0, \quad 0, \ldots, n$$

$\rightsquigarrow$ Beweis: [Rannacher]

Beispiele

Durch Auswerten der Nullstellen der Legendre-Polynome ergeben sich mit

$$h = \frac{b-a}{2}, \quad c = \frac{a+b}{2}$$

zum Beispiel die folgenden Gauß-Quadraturen:

Eine Stützstelle, Ordnung 2:

$$I^{(0)} = (b-a) \cdot f(c) \quad \text{(Mittelpunktsregel)}$$

Zwei Stützstellen, Ordnung 4:

$$I^{(1)} = \frac{b-a}{2} \cdot [f(c - (1/3)^{1/2} h) + f(c + (1/3)^{1/2} h)]$$

Beispiele

Drei Stützstellen, Ordnung 6:

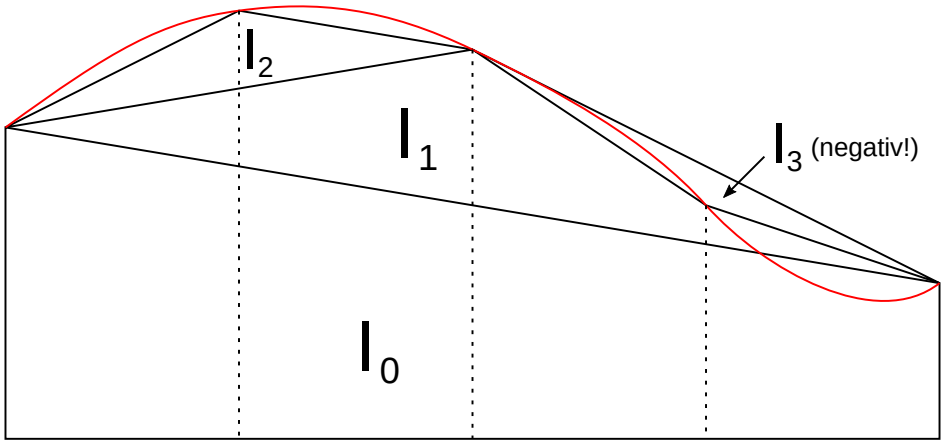
$$I^{(2)} = \frac{b-a}{18} \cdot [5f(c-(3/5)^{1/2}h) + 8f(c) + 5f(c+(3/5)^{1/2}h)]$$

Natürlich lassen sich mithilfe der Gauß-Quadraturen wieder summierte Formeln definieren, indem man das Intervall unterteilt und die Ergebnisse summiert.

Adaptive Quadratur

Oftmals ist eine äquidistante Aufteilung des Intervals wenig sinnvoll, da die zu integrierende Funktion lokal unterschiedliches Verhalten zeigt.

Idee der *adaptiven* Quadratur:



- Berechne sukzessive Updates für die Näherung durch feinere Unterteilung (Prinzip des Archimedes)
- Breche Verfeinerung ab, wenn lokales Update $|I_j|$ klein genug

Mehrdimensionale Quadratur

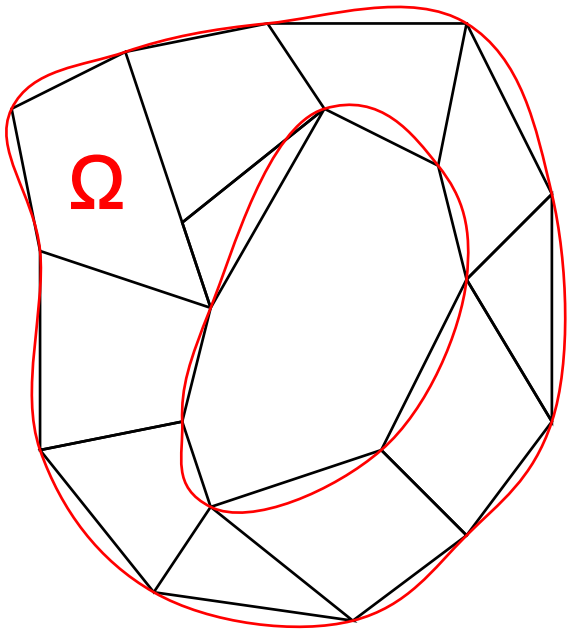
Im Eindimensionalen gilt:

$$\begin{aligned}\text{Intervall} &= \text{zusammenhängende Menge}\\ &= \text{wegzusammenhängende Menge}\\ &= \text{einfach zusammenhängende Menge}\end{aligned}$$

Daher ist es einfach, auf beliebigen "Gebieten" $\Omega \subset \mathbb{R}$ zu integrieren. Im Mehrdimensionalen werden die Intervalle zu (Hyper-) Quadern, die oben genannten Begriffe sind aber nicht mehr äquivalent:

$$\begin{aligned}\text{Quader} &\neq \text{zusammenhängende Menge}\\ &\neq \text{wegzusammenhängende Menge}\\ &\neq \text{einfach zusammenhängende Menge}\end{aligned}$$

Mehrdimensionale Quadratur



- Die vorgestellten Formeln lassen sich leicht auf Quader verallgemeinern.
- Im allgemeinen muss man diese Quader und die Integrale darauf transformieren (Transformationssatz, Verallgemeinerung der Substitutionsregel).
- Man kann auch Dreiecke für die Unterteilung verwenden.
- Bei komplizierten Geometrien entsteht zusätzlich ein Geometriefehler durch Approximation der Geometrie.

Fluch der Dimension

Wenn die Dimension d sehr groß ist, sind die hier behandelten Methoden nicht brauchbar.

Betrachte $\Omega = [0,1]^d$. Zerlegt man $[0,1]$ in zwei Teilintervalle je Dimension, so hat man den d-dimensionalen Würfel in 2^d Teilwürfel zerlegt. Der Aufwand steigt somit exponentiell in d an. Dies bezeichnet man als "Fluch der Dimension".

Eine Möglichkeit ist dann die Monte-Carlo-Integration

$$I(f) \approx \frac{C}{N} \sum_{i=1}^{N} f(\xi_i)$$

mit zufälligen Punkten $\xi_i \in \Omega$.

Abschnitt 9

9 Iterative Lösung von Gleichungssystemen
- Einführung
- Newton-Verfahren
- Sukzessive Approximation (Fixpunktiteration)
- Iterationsverfahren zur Lösung linearer Gleichungssysteme

Einführung

In diesem Abschnitt betrachten wir die Lösung von algebraischen Gleichungen

$$f(x) = 0 \quad \text{mit} \quad f : \mathbb{R}^n \to \mathbb{R}^n.$$

Dabei beschränken wir uns zunächst auf den Fall $n = 1$ (skalar).

Idee der *Intervallschachtelung*:
Angenommen man kennt ein Teilintervall $[a_0, b_0]$, so dass $f(a_0) \cdot f(b_0) < 0$ (Vorzeichenwechsel), und f sei stetig. Dann hat f nach dem Zwischenwertsatz mindestens eine Nullstelle in $[a_0, b_0]$.
Bilde Folge von Teilintervallen durch Bisektion und Prüfen der Bedingung.

Algorithmus: Intervallschachtelung

Gegeben: $I_0 = [a_0, b_0]$ mit $f(a_0) \cdot f(b_0) < 0$ und $\epsilon > 0$

Algorithmus:

$t = 0;$
while $(b_t - a_t > \epsilon)$:
    $x_t = (a_t + b_t)/2;$
    if $(f(x_t) == 0)$:
        $a_{t+1} = x_t - \epsilon;\ b_{t+1} = x_t;$
    else if $(f(a_t) \cdot f(x_t) < 0)$:
        $a_{t+1} = a_t;\ b_{t+1} = x_t;$
    else :
        $a_{t+1} = x_t;\ b_{t+1} = b_t;$
    $t = t + 1;$

Idee:

- Halbiere das aktuelle Intervall
- Falls zufällig in der Mitte eine Nullstelle liegt, beende das Verfahren
- Falls die Bedingung für das linke Intervall erfüllt ist, wähle es als nächstes Intervall
- Andernfalls muss die Nullstelle im rechten Intervall sein, setze die Suche dort fort

Analyse

Es gilt

$$a_t \leq a_{t+1} < b_{t+1} \leq b_t$$

und (sofern nicht $f(x_t) = 0$)

$$|b_{t+1} - a_{t+1}| \leq \frac{1}{2}|b_t - a_t| = \left(\frac{1}{2}\right)^{t+1} |b_0 - a_0|$$

Anmerkungen:

- Die Konvergenz ist linear mit Rate $1/2$
- Gut für monotone Funktionen geeignet (globale Konvergenz)
- Nur für reelle Funktionen im $\mathbb{R}^1$ geeignet

Newton-Verfahren

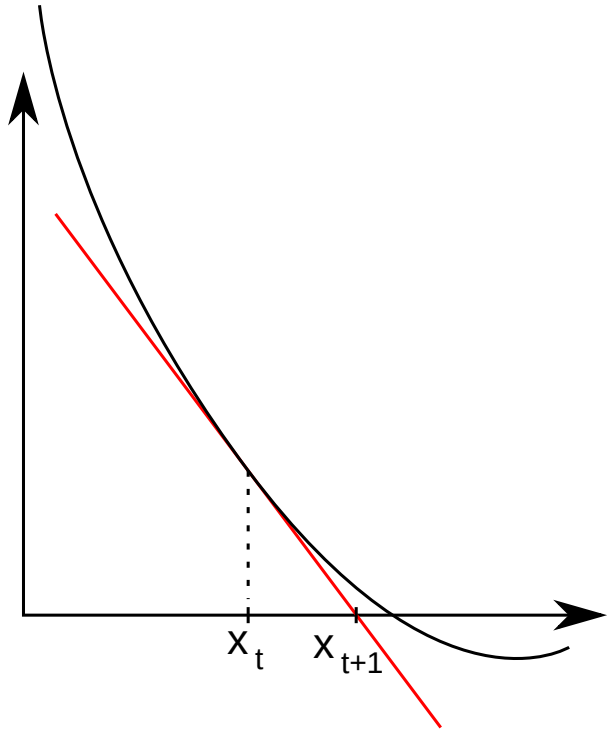
Die Funktion f sei (mindestens) einmal stetig differenzierbar.

Für gegebenes x_t gibt es die "Tangente"

$$T_t(x) = f'(x_t)(x - x_t) + f(x_t)$$

mit der Nullstelle

$$T_t(x) = 0 \iff x = x_t - \frac{f(x_t)}{f'(x_t)}.$$



Das führt auf die Iterationsvorschrift

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}.$$

Offensichtlich ist $|f'(x_t)| > 0$ erforderlich, d.h. wir setzen voraus, dass die Nullstelle *einfach* ist.

Newton-Verfahren im Mehrdimensionalen

Das Newton-Verfahren lässt sich auf Systeme $f : \mathbb{R}^n \to \mathbb{R}^n$ erweitern:

Es existiere die Taylorentwicklung von f:

$$f_i(x) = f_i(x_t+\Delta x) = f_i(x_t)+\sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x_t)\Delta x_i+R_i(x_t,\Delta x) \quad i = 1,\ldots,n$$

oder in vektorieller Schreibweise

$$f(x_t + \Delta x) = f(x_t) + J(x_t)\Delta x + R(x_t, \Delta x)$$

mit der "Jacobimatrix"

$$[J(x_t)]_{ij} = \frac{\partial f_i}{\partial x_j}(x_t)$$

Newton-Verfahren im Mehrdimensionalen

Das Ignorieren des Restglieds entspricht einer "Linearisierung von f".
Finde näherungsweise eine Nullstelle von f:

$$f(x) \approx f(x_t) + J(x_t)\Delta x = 0 \iff \Delta x = -J^{-1}(x_t)f(x_t)$$

Dies führt auf die Iteration

$$x_{t+1} = x_t - J^{-1}(x_t)f(x_t)$$

Dabei erfordert jeder Schritt die Lösung eines linearen Gleichungssystems mit der Jacobimatrix!

Konvergenz des Newton-Verfahrens I

Satz 9.1 (Newton-Verfahren)

Die Funktion $f \in C^2([a,b])$ habe in (a,b) (Inneres!) eine Nullstelle z, und es sei

$$m := \min_{a \le x \le b} |f'(x)| > 0, \quad M := \max_{a \le x \le b} |f''(x)|.$$

Es sei $\rho > 0$ so gewählt, dass

$$q := \frac{M}{2m}\rho < 1, \quad K_\rho(z) := \{x \in \mathbb{R} \mid |x - z| \le \rho\} \subset [a,b]$$

Dann sind für jeden Startwert $x_0 \in K_\rho(z)$ die Newton-Iterierten $x_t \in K_\rho(z)$ definiert und konvergieren gegen die Nullstelle z.

Konvergenz des Newton-Verfahrens II

Satz 9.1 (Newton-Verfahren)

Dabei gilt die a-priori Fehlerabschätzung

$$|x_t - z| \leq \frac{2m}{M} q^{(2^t)}, \quad t \in \mathbb{N}$$

und die a-posteriori Fehlerabschätzung

$$|x_t - z| \leq m^{-1}|f(x_t)| \leq \frac{M}{2m}|x_t - x_{t-1}|^2, \quad t \in \mathbb{N}.$$

(a-priori: nutzt nur die Voraussetzungen, a-posteriori: nutzt auch die bereits berechneten Iterierten)

$\rightsquigarrow$ Beweis: Tafel

Beispiel: Wurzelberechnung

Beispiel 9.2 (Wurzelberechnung mit Newton-Verfahren)

Sei $a > 0$ und $n \geq 1$. Löse $x^n = a$, d.h.

$$f(x) = x^n - a = 0, \quad f'(x) = n \cdot x^{n-1}.$$

Das führt auf die Iterationsvorschrift

$$x_{t+1} = n^{-1} \cdot [(n-1) \cdot x_t + a \cdot x_t^{1-n}].$$

Nach Satz 9.1 konvergiert die Iteration, falls x_0 nahe genug an $a^{1/n}$ ist. Hier konvergiert sie jedoch *global*, d.h. für alle $x_0 > 0$ (aber nicht unbedingt von Anfang an quadratisch).

Anmerkungen I

Bemerkung 9.3

- Das Newton-Verfahren konvergiert nur *lokal*, d.h. wenn $|x_0 - z| \leq \rho$ ("Einzugsbereich"). Dabei ist ρ i.d.R. unbekannt und möglicherweise sehr klein.
- Das Newton-Verfahren konvergiert quadratisch,

$$|x_t - z| \leq c \cdot |x_{t-1} - z|^2,$$

im Gegensatz zur Intervallschachtelung, die nur linear konvergiert.

Anmerkungen II

Bemerkung 9.3

- Gedämpftes Newton-Verfahren:
  Verbesserung der Konvergenz *außerhalb* des Einzugsbereichs:

$$x_{t+1} = x_t - \lambda_t \frac{f(x_t)}{f'(x_t)}$$

  mit der Wahl einer Folge $\lambda_t \in (0, 1]$ als "Dämpfungsstrategie".

Anmerkungen III

Bemerkung 9.3

- Mehrfache Nullstellen:
  Falls z eine p-fache Nullstelle ist, mit $p > 1$, konvergiert das Newton-Verfahren dennoch, aber nur noch linear. Man kann zeigen, dass die alternative Vorschrift

$$x_{t+1} = x_t - p \cdot \frac{f(x_t)}{f'(x_t)}$$

  in diesem Fall quadratisch konvergiert.

Sekanten-Methode

Unter Umständen ist die Berechnung der Ableitungen sehr teuer.

Idee der *Sekanten-Methode*:
Ersetze die Tangente durch eine Sekante

$$s(x) = f(x_t) + (x - x_t)\frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}},$$

dies führt auf die Iteration

$$x_{t+1} = x_t - f(x_t)\frac{x_t - x_{t-1}}{f(x_t) - f(x_{t-1})}.$$

Sekanten-Methode

Die Sekanten-Methode konvergiert lokal mit

$$|x_t - z| \leq \frac{2m}{M} q^{\gamma_t}, \quad t \in \mathbb{N},$$

wobei γ_t die *Fibonacci-Zahlen* bezeichnet:

$$\gamma_0 = \gamma_1 = 1, \quad \gamma_{t+1} = \gamma_t + \gamma_{t-1}$$

Es gilt $\gamma_t \approx 0.723 \cdot (1.618)^t$ ("goldener Schnitt"), also liegt die Konvergenzordnung zwischen 1 und 2. Problematisch ist die bei der Auswertung auftretende Auslöschung.

Fixpunktiteration

Mit $g(x) = x - \frac{f(x)}{f'(x)}$ hat das Newton-Verfahren die Form $x_{t+1} = g(x_t)$. Da die Nullstelle z wegen

$$f(z) = 0 \implies g(z) = z$$

ein Fixpunkt der Iteration $x_{t+1} = g(x_t)$ ist, nennt man das auch Fixpunktiteration. Hier untersuchen wir allgemeine Iterationen dieser Art.

Falls die Berechnung von $f'(x)$ sehr teuer ist, könnte man z.B. f' nur einmal "in der Nähe von z" auswerten, und wie folgt iterieren:

$$x_{t+1} = x_t - \frac{f(x)}{f'(c)}$$

Frage: Wann konvergiert so eine Iteration? Insbesondere wollen wir dabei auch $f : \mathbb{R}^n \to \mathbb{R}^n$, $n > 1$, zulassen.

Kontraktionen

Antwort gibt der sogenannte "Banachsche Fixpunktsatz".

Dafür zunächst eine Definition:

Definition 9.4 (Kontraktion)

Sei $G \subset \mathbb{R}^n$ eine nichtleere, *abgeschlossene* Punktmenge und $g\colon G \to G$ Lipschitz-stetig mit Konstante $q < 1$, d.h.

$$\|g(x) - g(y)\| \leq q \cdot \|x - y\|$$

mit einer Vektornorm $\|\cdot\|$ im $\mathbb{R}^n$. Ein solches g nennt man eine *Kontraktion* auf G.

Sukzessive Approximation

Satz 9.5 (Sukzessive Approximation)

Sei g eine Kontraktion auf $G \subset \mathbb{R}^n$. Dann existiert genau ein Fixpunkt $z \in G$ von g und für jeden Startpunkt $x^{(0)} \in G$ konvergiert die Folge der Iterierten $x^{(t+1)} = g(x^{(t)})$ gegen z.

Es gelten die a-priori und a-posteriori Fehlerabschätzungen

$$\|x^{(t)} - z\| \leq \frac{q}{1-q}\|x^{(t)} - x^{(t-1)}\| \leq \frac{q^t}{1-q}\|x^{(1)} - x^{(0)}\|.$$

(wir schreiben den Index oben in Klammern, um bei Vektoren unten Platz für den Komponentenindex zu haben.)

$\rightsquigarrow$ Beweis: Tafel

Iterationsverfahren zur Lösung von LGS

Wir kehren zurück zur Lösung von linearen Gleichungssystemen

$$Ax = b, \quad A \in \mathbb{R}^{n \times n} \text{ regulär}, \quad b \in \mathbb{R}^n.$$

Definition 9.6 (dünn besetzte Matrizen)

Eine Folge von Matrizen $\{A^{(n)} \mid n \in \mathbb{N}\}$ heißt *dünn besetzt*, falls

$$|\{a_{ij}^{(n)} \mid a_{ij}^{(n)} \neq 0\}| =: \operatorname{nnz}(A^{(n)}) = \mathcal{O}(n)$$

(nnz = "number of non-zeros").

Aufgrund von "fill in" ist die Gauß-Elimination für dünn besetzte Matrizen oft schlecht geeignet. Außerdem: für große n nicht mehr tragbarer Aufwand in $\mathcal{O}(n^3)$.

Iterationsverfahren zur Lösung von LGS

Es gilt

$$\text{Lösen von } Ax = b \iff \text{“Nullstellensuche” } f(x) = b - Ax = 0.$$

Definiere mit Matrix C die Iteration

$$\begin{aligned} x^{(t+1)} &= g(x^{(t)}) = x^{(t)} + C^{-1} f(x^{(t)}) \\ &= x^{(t)} + C^{-1}(b - Ax^{(t)}) \\ &= \underbrace{(I - C^{-1}A)}_{=:B} x^{(t)} + C^{-1} b \end{aligned}$$

mit “Iterationsmatrix” B.

Iterationsverfahren zur Lösung von LGS

Für $x := A^{-1}b$ gilt

$$g(x) = (I - C^{-1}A)x + C^{-1}b = x - (C^{-1}A) \cdot (A^{-1}b) + C^{-1}b = x,$$

also ist x Fixpunkt von g.

Für die Lipschitz-Konstante der Funktion g gilt

$$\|g(x) - g(y)\| = \|B(x - y)\| \leq \|B\| \cdot \|x - y\|,$$

d.h. falls $\|B\| < 1$ (für verträgliche Matrixnorm $\|\cdot\|$) ist, ist g Kontraktion auf $\mathbb{R}^n$.

Beispiele für Iterationsverfahren

Zerlege $A = L + D + U$ mit L strikte untere Dreiecksmatrix, D Diagonalmatrix und U strikte obere Dreiecksmatrix.

Jacobi-Verfahren:
Setze $C = D$, also

$$x^{(t+1)} = x^{(t)} + D^{-1}(b - Ax^{(t)})$$

Gauß-Seidel-Verfahren:
Setze $C = L + D$, also (Vorwärtseinsetzen)

$$x^{(t+1)} = x^{(t)} + (L + D)^{-1}(b - Ax^{(t)})$$

Solche Iterationsverfahren konvergieren normalerweise nur für bestimmte Klassen von Matrizen (da $\|B\| < 1$ sein muss).

Konvergenz des Jacobi-Verfahrens

Erinnerung:
Eine Matrix heißt *strikt diagonaldominant*, falls

$$\sum_{j \neq i} |a_{ij}| < |a_{ii}| \quad \forall\, i = 1, \ldots, n.$$

Beispiele: LGS für kubische Splines, Radiosity-Methode.

Satz 9.7

Das Jacobi-Verfahren konvergiert für strikt diagonaldominante Matrizen.

$\rightsquigarrow$ Beweis: Tafel

Aufwand

Es gibt viele weitere solche Aussagen für symmetrisch positiv definite Matrizen, schwach diagonaldominante Matrizen, sogenannte M-Matrizen, ...

Der Aufwand für eine Iteration sei $\alpha(n)$, typischerweise mit $\alpha(n) = \mathcal{O}(n)$. Wegen

$$\|x^{(t)} - x\| \leq \|B\|^t \|x^{(0)} - x\|$$

sind für eine Reduktion des Fehlers um den Faktor $\epsilon \ll 1$ insgesamt $t \geq \frac{\log(\epsilon)}{\log(\|B\|)}$ Iterationen nötig, also ist der Gesamtaufwand

$$T_{\text{fix}}(n) = \frac{\log(\epsilon)}{\log(\|B\|)} \alpha(n).$$

Problem: hohe Kosten wenn $\|B\|$ nahe Eins, $\|B\|$ ist problemabhängig und wächst häufig mit n.

Konjugierte Gradienten

Für symmetrisch positiv definite Matrizen A kann man stattdessen das Verfahren der konjugierten Gradienten (CG-Verfahren) verwenden. Es berechnet mit einem Startwert x_0, $r_0 := b - Ax_0$ und $d_0 := r_0$ die Iterierten

$$\begin{aligned}
\alpha_t &= \frac{r_t^T r_t}{d_t^T A d_t} \\
x_{t+1} &= x_t + \alpha_t d_t \\
r_{t+1} &= r_t - \alpha_t A d_t \\
\beta_t &= \frac{r_{t+1}^T r_{t+1}}{r_t^T r_t} \\
d_{t+1} &= r_{t+1} + \beta_t d_t
\end{aligned}$$

- Das CG-Verfahren konvergiert in n Schritten als exaktes Verfahren.
- Für $n \gg 1$ kann man es als iteratives Verfahren verwenden, und erhält oft nach wenigen Schritten bereits gute Konvergenz.

Abschnitt 10

⑩ Schlussbemerkungen

Top 10 der wichtigsten Algorithmen

Die 10 Algorithmen mit "dem größten Einfluss auf Wissenschaft und Technik" im 20. Jahrhundert, laut *Computing in Science and Engineering* (2000):

- Metropolis algorithm for Monte Carlo
- Simplex method for linear programming
- Krylov subspace iteration methods
- The decompositional approach to matrix computations
- The Fortran optimizing compiler
- QR algorithm for computing eigenvalues
- Quicksort algorithm for sorting
- Fast Fourier transform
- Integer relation detection
- Fast multipole method

In rot: wurde im Rahmen der Vorlesung behandelt / angesprochen.

Top 10 der wichtigsten Algorithmen

Die Top 10 laut *Princeton Companion to Applied Mathematics* (2015, größerer Fokus auf angewandte Mathematik):

- Newton and quasi-Newton methods
- Matrix factorizations (LU, Cholesky, QR)
- Singular value decomposition, QR and QZ algorithms
- Monte-Carlo methods
- Fast Fourier transform
- Krylov subspace methods (conjugate gradients, Lanczos, GMRES, minres)
- JPEG
- PageRank
- Simplex algorithm
- Kalman filter

In rot: wurde im Rahmen der Vorlesung behandelt / angesprochen.

Weiterführende Veranstaltungen

Auswahl an weiterführenden Veranstaltungen in den Bereichen Numerik und Wissenschaftliches Rechnen:

- Numerik 1 (MD 1, Sommersemester)
- Numerische Lineare Algebra (MH5)
- Numerik gewöhnlicher Differentialgleichungen (MH6)
- Numerik partieller Differentialgleichungen (MH7)
- Object-Oriented Prog. for Scientific Computing (IOPWR)
- Paralleles Höchstleistungsrechnen (IPHR)
- Parallele Lösung großer Gleichungssysteme (IPLGG)

Falls Sie Interesse an einem Praktikum haben, das thematisch an die Vorlesung anschliesst, können Sie sich gerne bei mir melden.

Vielen Dank für die
Aufmerksamkeit
und gutes Gelingen für die Klausur!