Numerische Mathematik für die Fachrichtungen Informatik und Ingenieurwesen

Nicolas Neuß
Institut für Angewandte und Numerische Mathematik
Universität Karlsruhe
Englerstraße 2, 76128 Karlsruhe
neuss@math.uni-karlsruhe.de

Erstellt: 18. Juli 2008

URL für die Vorlesung:
http://www.mathematik.uni-karlsruhe.de/ianm3/lehre/numinf2008s/

Inhaltsverzeichnis

1 Einführung

1.1 Was ist „Numerische Mathematik“?

Definition: Numerische Mathematik beschäftigt sich mit der Entwicklung und Analyse von Methoden zur Lösung mathematischer Problemstellungen.

Schema:

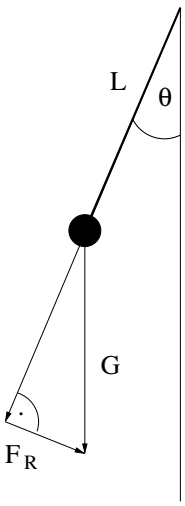
Problem des praktischen Lebens
$\Rightarrow$
Mathematisches Modell & Messung von Parametern
$\Rightarrow$
. . . mit einem guten numerischen Verfahren . . .
Brauchbare Lösung

1.2 Beispiel

Wir wollen die Schwingung eines Stabpendels berechnen.

1.2.1 Problemstellung

- L: Länge des Pendelstabs
- m: Masse des Pendels
- g: Erdbeschleunigung
- $G = mg$: Gewichtskraft
- θ: Auslenkungswinkel
- $F_R = G \sin \theta$: Rückstellkraft



1.2.2 Modellierung

Wissen: Aus der Mechanik wissen wir

- Auslenkungswinkel θ ist eine Funktion $\theta(t)$ der Zeit t
- Winkelgeschwindigkeit $\omega(t) = \dot{\theta}(t)$, wobei $\dot{\theta}(t) := \frac{d}{dt}\theta(t)$.
- Geschwindigkeit des Pendelgewichts ist $v(t) = L\omega(t)$
- Beschleunigung des Pendelgewichts $a(t) = \dot{v}(t) = L\dot{\omega}(t) = L\ddot{\theta}(t)$
- Rückstellkraft $F_R(t) = -mg\sin\theta(t)$ (Vorzeichen!)
- Andererseits: $F_R(t) = ma(t) = mL\ddot{\theta}(t)$ (Newtonsches Kraftgesetz)

1.2.3 Mathematisches Modell

Zusammensetzen liefert die Differentialgleichung

$$\ddot{\theta}(t) = -\frac{g}{L}\sin\theta(t)$$

zu deren Lösung man noch Anfangsbedingungen braucht:

$$\theta(0) = \theta_0 \quad \dot{\theta}(0) = \omega(0) = \omega_0\,.$$

Bemerkung: Für diese Differentialgleichung kann man die Lösung $\theta(t)$ nicht mittels elementarer Funktionen ausdrücken (was für viele Differentialgleichungen der Fall ist). Man muss sie daher numerisch lösen.

1.2.4 Numerisches Verfahren

- *Approximiere* $\dot{\theta}(t)$ und $\dot{\omega}(t)$ durch die Differenzenquotienten (mit kleinem $h > 0$)

$$\dot{\theta}(t) \approx \frac{\theta(t+h) - \theta(t)}{h}\,, \quad \dot{\omega}(t) \approx \frac{\omega(t+h) - \omega(t)}{h}\,.$$

- Andererseits wissen wir: $\dot{\theta}(t) = \omega(t)\,, \quad \dot{\omega}(t) = -\frac{g}{L}\sin\theta(t)\,.$
- Man kann nun aus $(\theta(t), \omega(t))$ eine Approximation zu $(\theta(t+h), \omega(t+h))$ berechnen:

$$\begin{aligned}\theta(t+h) &\approx \theta(t) + h\omega(t)\\ \omega(t+h) &\approx \omega(t) + h(-\frac{g}{L}\sin\theta(t))\end{aligned}$$

1.2.5 Theorie

Satz: (Analysis) Die Differentialgleichung

$$\ddot{\theta}(t) = -\frac{g}{L}\sin\theta(t)\,, \quad \theta(0) = \theta_0,\ \dot{\theta}(0) = \omega_0$$

besitzt eine eindeutige Lösung $\theta : \mathbb{R} \to \mathbb{R}$ für alle Anfangsbedingungen (θ_0, ω_0).

Satz: (Numerik) Das numerische Verfahren (es heißt explizites Euler-Verfahren)

$$\begin{pmatrix}\theta_{h,0}\\ \omega_{h,0}\end{pmatrix} = \begin{pmatrix}\theta_0\\ \omega_0\end{pmatrix}\,, \quad \begin{pmatrix}\theta_{h,k+1}\\ \omega_{h,k+1}\end{pmatrix} = \begin{pmatrix}\theta_{h,k}\\ \omega_{h,k}\end{pmatrix} + h\begin{pmatrix}\omega_{h,k}\\ -\frac{g}{L}\sin\theta_{h,k}\end{pmatrix} \text{ für } k \geq 0$$

konvergiert, d.h. für jedes $t > 0$ gilt

$$\lim_{N\to\infty,\, h=\frac{t}{N}\to 0} \theta_{h,N} \to \theta(t)\,, \qquad \lim_{N\to\infty,\, h=\frac{t}{N}\to 0} \omega_{h,N} \to \omega(t) = \dot{\theta}(t)\,.$$

1.2.6 Implementation

Eine für die Numerik sehr gut geeignete Programmiersprache ist das kommerzielle Matlab, oder aber freie Derivate davon, insbesondere Octave und Scilab.

Vorteile:

- Starke Sprache zur Matrizenmanipulation
- Interaktives Arbeiten
- Die Sprache ist recht konventionell, d.h. die meisten Sprachelemente kennt man aus anderen Computersprachen in sehr ähnlicher Form.
- Viele Bibliotheken

Bemerkung: Wegen der freien Verfügbarkeit werden wir in diesem Kurs vorerst Scilab benutzen.

1.2.7 Programm

Programm: (`pendel.sci`)

```
g = 9.81; L = 1.0;

function theta = pendel(theta0, omega0, T, N)
  h = T/N;
  time(1) = 0.0;
  theta(1) = theta0; omega(1) = omega0;
```

```
  for i=1:N
    time(i+1) = time(i)+h;
    theta(i+1) = theta(i)+h*omega(i);
    omega(i+1) = omega(i)-h*g/L*sin(theta(i));
  end
endfunction

theta0 = 0.0; omega0 = 1.0;
T = 1.0; N = 1000;

for i=1:10
  N=2^i;
  theta = pendel(theta0, omega0, T, N);
  //disp(theta(N+1));
  if i>2
    disp(theta_i-theta(N+1));
  end
  theta_i=theta(N+1);
end

//plot(time,theta)  // or: plot(time,[theta,omega])
// ... test also N=10000 and values of omega0 around 6.26
```

1.2.8 Beobachtungen

- Die Lösungskurve approximiert die korrekte Pendelbewegung für $N \to \infty$.
- Die Approximation ist leider nicht sehr schnell (Fehler $O(h)$ siehe Übung).
- Wenn man auch ω plottet, sieht man den Austausch zwischen potentieller und kinetischer Energie.
- Das Verfahren ist nicht exakt energieerhaltend, sondern die Energie nimmt im Verlauf der Iteration zu.

1.2.9 Sinnvolle Problemstellungen

Problem: Gegeben sei $L = 1\mathrm{m}$, $\theta_0 = \pi$, $\omega_0 = 0\frac{\mathrm{m}}{\mathrm{s}}$. Zu dieser instabilen Lage betrachten wir drei Fragestellungen:

(I) Fällt das Pendel rechts herum, links herum, oder bleibt es oben stehen?

(II) Wie ist die Lage des Pendels nach 2 Sekunden?

(III) Wie ist die Lage des Pendels nach einer Minute?

Frage: Welche der obigen Fragestellungen ist für die Praxis brauchbar?

Antwort:

- Frage (I) ist in der Praxis unbrauchbar, weil die Antwort „Das Pendel bleibt für immer bei $\theta = \pi$ stehen“ durch beliebig kleine Störung der Anfangsdaten (welche in der Praxis unvermeidbar sind, z.B. durch Luftzug oder Erschütterungen) falsch wird.
- Frage (II) ist sinnvoll, weil die Antwort „Das Pendel befindet sich nach 2 Sekunden immer noch bei $\theta = \pi$“ auch bei kleinen Störungen der Anfangsdaten noch (halbwegs) richtig ist.
- Auch Frage (III) ist in der Praxis unbrauchbar. Eine genauere Betrachtung zeigt nämlich, dass ein Anfangsfehler ε mit der Zeit t um den Faktor $e^{t\sqrt{g/L}}$ wächst. Für $L = 1\mathrm{m}$ und $t = 60\mathrm{s}$ ist dieser Faktor etwa $4 \cdot 10^{80}$.

Bemerkung: Falls man experimentell dennoch ein Stehenbleiben bei $\theta = \pi$ beobachten sollte, liegt das wahrscheinlich an Haftreibungseffekten in der Aufhängung, die im mathematischen Modell der Pendelgleichung nicht berücksichtigt werden.

Definition: Man nennt Probleme/Fragestellungen, bei denen sich die Antwort stark ändert, obwohl die Eingangsdaten nur wenig variiert werden, schlecht gestellt oder schlecht konditioniert.

Bemerkung: In der Praxis sind leider viele Probleme recht schlecht konditioniert. Manchmal ist eine geänderte Fragestellung die beste Abhilfe, ansonsten sollte aber der Numeriker danach trachten, den Fehler nicht auch noch durch ungeeignete Verfahren zu vergrößern.

1.2.10 Etwas Scilab

- *zeros(m,n)* — Nullmatrix
- *eye(m,n)* — Einheitsmatrix
- *[1,2;3,4]* — Matrix aus Elementen
- *A(i,j)* — Zugriff auf Matrixelement
- *i:j* — Integerbereich
- *i:d:j* — Bereich mit Schrittweite d
- *A*B* — Matrizenprodukt
- *A\b* — Lösung LGS
- *A'* — Transposition
- *det(A)* — Determinante
- *spec(A)* — Eigenwerte/Spektrum
- *dot(x,y)* — Skalarprodukt
- *if condition . . . (Codezeilen). . . end*
- *if condition . . . else . . . end*
- *for i=1:n . . . end* — Schleife
- *function y = name (x) . . . endfunction* — Funktionsdefinition

1.3 Philosophie dieses Kurses

- Wir lernen die *Arbeitsweise*
- von *numerischen Verfahren* kennen,
- die für viele *wissenschaftliche Anwendungen* wichtig sind.
- Dieses Hintergrundwissen ist *notwendig*,
- um *Standardsoftware*
- *korrekt und sicher* anwenden zu können.

1.4 Beabsichtigter Inhalt

- Gleitkommazahlen, Kondition, Stabilität, Vektor- und Matrixnormen
- Direkte und iterative Lösung linearer Gleichungssysteme
- Lineare Ausgleichsprobleme
- Lineare Eigenwertprobleme
- Lösung nichtlinearer Probleme (Fixpunktsatz, Newton-Verfahren)
- Polynominterpolation; Fouriertransformation
- Numerische Quadratur
- Lösung gewöhnlicher Differentialgleichungen

1.5 Technische Hinweise

- Die Vorlesung am Mittwoch fängt in Zukunft um *8:15 Uhr* an, die Übung am Freitag um *10:00 Uhr*. Am *18.4.* ist noch *keine Übung*!
- Wer einen *Schein* erhalten will, muss die *Klausur* bestehen.
- Wer die *Klausur* mitschreiben will, muss bei den *Übungen* mindestens 50% der Punkte erreichen.
- Wer an den Übungen teilnehmen will, *muss sich in der Vorlesungsverwaltung anmelden* (Link auf der Homepage). Bitte tun Sie das *so bald wie möglich*!
- Beachten Sie, dass die *Klausur direkt nach der Vorlesung* stattfindet! Eine *kontinuierliche Mitarbeit* in Vorlesung und Übungen ist *unbedingt notwendig*!

1.6 Zum Skript

- Das Skript besteht aus den in der Vorlesung verwendeten Folien, die in anderem Format ausgedruckt wurden. Abgesehen von Korrekturen sollte der Inhalt identisch sein.
- Man beachte aber, dass verschiedene Sachverhalte in der Vorlesung zusätzlich zur Information auf den Folien auch noch an der Tafel veranschaulicht werden.
- Insbesondere enthalten die Folien kaum Bilder. Diese werden in der Vorlesung teils an der Tafel gemalt, teils aber auch durch Demonstrationen am Computer erzeugt.

2 Grundlagen

2.1 Motivation

2.1.1 Beispiel

Aufgabe: Man berechne $e^x = \exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$ auf eine Genauigkeit $\varepsilon > 0$.

Algorithmus: Berechne $\sum_{k=0}^{N} \frac{x^k}{k!}$ wobei N so gewählt ist, dass $|\frac{x}{N}| \leq \frac{1}{2}$ und $|\frac{x^N}{N!}| \leq \varepsilon$.

Satz: Bei exakter Arithmetik berechnet dieser Algorithmus $\exp(x)$ auf einen Fehler $\varepsilon > 0$ genau.

Beweis:

$$|\sum_{k=N+1}^{\infty} \frac{x^k}{k!}| \leq |\frac{x^N}{N!}| \, |\sum_{l=1}^{\infty} \frac{x^l}{(N+1)\cdots(N+l)}| \leq \varepsilon \sum_{l=1}^{\infty} \left(\frac{1}{2}\right)^l = \varepsilon$$

Programm: (`exp_series.sce`)

```
eps = 1.0e-15;

function s = exp_series (x)
  s = 1;
  k = 1;

  summand = x/k;
  while (abs(summand)>eps) | (abs(x/k)>0.5) do
    s = s + summand;
    k = k+1;
    summand = summand*x/k;
  end
endfunction

for x=40:-10:-40
  disp(x)
  disp([exp(x),exp_series(x)]);
end
```

Beobachtung: Wir erhalten folgende Tabelle:

x	$\exp(x)$	`exp_series(x)`
40	2.354E+17	2.354E+17
30	1.069E+13	1.069E+13
20	4.852E+08	4.852E+08
10	22026.466	22026.466
0	1.0000000	1.0000000
-10	0.0000454	0.0000454
-20	2.061E-09	5.622E-09
-30	9.358E-14	-0.0000307
-40	4.248E-18	-3.1657319

Offenbar versagt unser Algorithmus für $x \ll 0$.

Abhilfe: Berechne für $x < 0$ einfach $\dfrac{1}{\texttt{exp\_series}(-x)}$.

2.2 Gleitkomma-Arithmetik

2.2.1 Ganzzahl-Arithmetik

Ein Computer kann aufgrund seines begrenzten Speichers nicht einmal alle ganzen Zahlen $k \in \mathbb{Z}$ darstellen. Viele Computersprachen beschränken sich sogar auf einen bestimmten Bereich, etwa $-2^{31}, \ldots 2^{31} - 1$ (kann durch 32 Bit dargestellt werden) oder $-2^{63}, \ldots 2^{63} - 1$ (kann durch 64 Bit dargestellt werden). Die Darstellung geschieht oft vorzeichenlos als $\tilde{k} = k + 2^{32}$ bzw. $\tilde{k} = k + 2^{64}$.
Die Arithmetik ist die übliche, solange das Ergebnis nicht überläuft, d.h. wenn es innerhalb des erlaubten Bereichs bleibt. Ein Überlauf führt in manchen Computersprachen zu einem Laufzeitfehler, in anderen wird eine Modulo-Arithmetik durchgeführt, wieder andere gehen zu einer anderen Darstellung über (z.B. Gleitkommazahlen oder Zahlen beliebiger Länge).

2.2.2 Wissenschaftliche Darstellung

Auch kann ein Computer nicht mit reellen Zahlen beliebiger Genauigkeit rechnen. Man muss daher Approximationen verwenden. Außerdem spielen in der Praxis sehr unterschiedliche Größenordnungen eine Rolle (z.B. Atomdurchmesser $1.0 \cdot 10^{-10} m$, Entfernung Erde-Sonne $1.5 \cdot 10^{11} m$). Daher ist es sinnvoll, Approximationen zu verwenden, die einen kleinen relativen Fehler aufweisen.

Idee: Approximiere reelle Zahlen durch eine wissenschaftliche Darstellung der Form

$$\pm 0, d_1 d_2 \ldots d_m \cdot B^e$$

Hierbei ist B die Basis, m die Mantissenlänge, $d_i \in \{0, \ldots, B-1\}$ die Ziffern, $e \in \mathbb{Z}$ der Exponent und $\pm$ ein Vorzeichen.

Bemerkung: Die Darstellung wird eindeutig, wenn man 0 durch $d_1 = \ldots = d_m = 0$, $e = 0$ und das Vorzeichen $+$ darstellt, aber ansonsten $d_1 \neq 0$ fordert.

2.2.3 IEEE-Gleitkommazahlen

Auf üblichen Computern wird meist die Basis $B = 2$ verwendet. Die bekannte doppeltgenauen IEEE-Gleitkommazahlen verwenden zudem noch $m = 52$. Außerdem wird der Exponent e noch auf den Bereich $-1022, \ldots, 1023$ beschränkt.

Bemerkungen:

- Das Format braucht 1 Bit (Vorzeichen)+52 Bit (Mantisse) + 11 Bit (Exponent), also insgesamt 64 Bit=8 Bytes.
- Die kleinste darstellbare positive Zahl ist somit $a_{\text{posmin}} = 2^{-1022} \approx 2.225 \cdot 10^{-308}$, die größte ist $a_{\text{posmax}} = 2^{1024} - 2^{1972} \approx 1.8 \cdot 10^{308}$.
- Der Exponent e wird vorzeichenfrei als $C = e + 1023$ dargestellt, wobei man C die Charakteristik nennt. $C = 0$ ist für 0 und $C = 2047$ für `NaN` (Not-a-Number) und `inf` (unendlich) reserviert.

Rundung Jede reelle Zahl x in $[-a_{\text{posmax}}, -a_{\text{posmin}}] \cup [a_{\text{posmin}}, a_{\text{posmax}}]$ kann nun durch diese Darstellung approximiert werden. Die entsprechende Approximation heißt Rundung. Die am häufigsten gewählte Rundung ist folgende: Wenn x die wissenschaftliche Darstellung

$$0, d_1 \cdots d_m d_{m+1} \cdots \cdot 2^e$$

hat, so wählen wir

$$\tilde{x} = \begin{cases} 0, d_1 \cdots d_m \cdot 2^e & d_{m+1} = 0 \\ 0, d_1 \cdots d_m \cdot 2^e + 0, 0_1 \cdots 0_{m-1} 1_m \cdot 2^e & d_{m+1} = 1 \end{cases}$$

Bezeichnung: Mit $\tilde{x} = \text{rd}(x)$ bezeichnen wir die auf eine Gleitkommazahl $\tilde{x}$ gerundete Zahl x.

Satz: Die Rundung approximiert eine Zahl x im zulässigen Bereich bis auf einen relativen Fehler $eps = 2^{-m-1}$ genau (was für $m = 52$ den Wert $eps \approx 10^{-16}$ liefert).

Beweis: Weil $|x| \geq 2^e$ und $|x - \tilde{x}| \leq 2^{-(m+1)} 2^e$ gilt dann für den relativen Fehler

$$\frac{|x - \tilde{x}|}{|x|} \leq 2^{-(m+1)}$$

Bemerkung: Endliche Gleitkommazahlen im Dezimalsystem (z.B. 0.1) haben normalerweise eine unendliche Dualdarstellung und müssen daher gerundet werden!

2.2.4 Arithmetik

Definition: Arithmetische Operationen auf Gleitkommazahlen sind so definiert, dass sie die entsprechende Operation in $\mathbb{R}$ durchführen und dann runden.

Beispiel: Sei $B = 2$ und $m = 3$ sowie $a = 0.111 \cdot 2^0$, $b = 0.101 \cdot 2^{-1}$. Dann erhalten wir

$$0.111 \cdot 2^0 \oplus 0.101 \cdot 2^{-1} = \mathrm{rd}(0.111 \cdot 2^0 + 0.101 \cdot 2^{-1}) = \mathrm{rd}(0,10011 \cdot 2^1) = 0.101 \cdot 2^1$$

und

$$0.111 \cdot 2^0 \odot 0.101 \cdot 2^{-1} = \mathrm{rd}(111 \cdot 2^{-3} \cdot 101 \cdot 2^{-4}) = \mathrm{rd}(100011 \cdot 2^{-7}) = 0.100 \cdot 2^{-1}$$

Bemerkung:

- Die Gleitkomma-Operationen *erfüllen nicht* das Assoziativitätsgesetz oder Distributivgesetz (abgesehen davon, dass auch ein Überlauf geschehen kann). Sie bilden also (im Gegensatz zu $\mathbb{R}$) *keine Gruppe* bezüglich $\oplus$ und *keinen Körper*!
- Die Analyse von Gleitkomma-Arithmetik geschieht meist, indem man die Operationen als Störungen der jeweiligen Operationen in $\mathbb{R}$ betrachtet, deren Ergebnis mit einem relativen Fehler der Ordnung eps behaftet ist.
- Längere Rechnungen entstehen durch Hintereinanderausführen von elementaren Gleitkommaoperationen. Die Fehlerfortpflanzung der dabei entstehenden Rundungsfehler ist ein Spezialfall des folgenden Abschnitts über die Kondition.
- Das Experiment vom Anfang dieses Kapitels erklärt sich dann so, dass für $x \ll 0$ die Rundungsfehler der Zwischenergebnisse viel größer sind als das sehr kleine Endergebnis.

2.3 Kondition und Stabilität

2.3.1 Qualitative Definition der Kondition

Definition: Eine numerische Aufgabe heißt gut konditioniert, wenn

1. zu den Eingabeparametern eine eindeutige Lösung existiert,
2. kleine Änderungen der Eingabeparameter nur zu kleinen Änderungen dieser Lösung führen.

Frage: Was heißt „klein“?

Antwort: Die Wahl *geeigneter* Normen, um den Ausdruck „klein“ zu *quantifizieren*, ist *problemabhängig*!

2.3.2 Absolute Kondition in 1D

Voraussetzung: Die Lösung einer von einem Parameter $x \in \mathbb{R}$ abhängigen numerischen Aufgabe sei gegeben als Funktion $f : \mathbb{R} \to \mathbb{R}, x \mapsto f(x)$ (somit ist die Lösung auf jeden Fall eindeutig). f sei außerdem stetig differenzierbar, d.h. es gibt eine stetige Funktion $f' : \mathbb{R} \to \mathbb{R}$, so dass für $x \in \mathbb{R}$ gilt

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + o(\Delta x) \quad \text{wobei} \quad \frac{o(\delta)}{\delta} \to 0 \ (\delta \to 0) \, .$$

Satz: Die absolute Kondition des durch f gelösten Problems an der Stelle x mit $f(x) = y$ definieren wir als $\kappa_{\text{abs}}(f, x) = |f'(x)|$. Es gilt

$$|f(x + \Delta x) - f(x)| = \kappa_{\text{abs}}(f, x)|\Delta x| + o(\Delta x) \, .$$

Hierbei ist $o : \mathbb{R} \to \mathbb{R}$ eine Funktion mit $\frac{o(\delta)}{\delta} \to 0$ für $\delta \to 0$.

Beweis: Dies folgt sofort aus der Definition von Differenzierbarkeit!

Bezeichnung: Für die obige Gleichung mit der o-Notation schreiben wir kurz

$$|f(x + \Delta x) - f(x)| \doteq \kappa_{\text{abs}}(f, x)|\Delta x| \, .$$

Das Zeichen $\doteq$ steht dabei für „ist in führender Ordnung gleich mit“.

Beispiel: Sei

$$f : \mathbb{R} \setminus \{0\} \to \mathbb{R} \setminus \{0\} \, , \quad x \mapsto \frac{1}{x} \, .$$

Dann ist die absolute Kondition an der Stelle x gegeben als $\kappa_{\text{abs}}(f, x) = \frac{1}{x^2}$.

Bemerkung: Eine große Konditionszahl ist daher ein Hinweis auf eine große Verstärkung von Fehlern und somit eine schlechte Konditionierung des Problems.

Aber: Die absolute Kondition misst die Variation im Funktionswert für (kleine) Variationen in x unabhängig von der Größe von x oder dem Funktionswert $f(x)$. Wie im nächsten Abschnitt gezeigt wird, ist dies nicht immer die günstigste Art, um die Gutgestelltheit eines Problems zu quantifizieren.

2.3.3 Relative Kondition in 1D

Beispiele:

- Eine einfache Umskalierung $x \mapsto sx$ hat die absolute Kondition $|s|$. Man hätte aber oft lieber ein Maß für die Güte eines Problems, das *gegen Skalierungen invariant ist*.
- Eine (positive) Gleitkommazahl $\tilde{x}$ repräsentiert eigentlich ein ganzes *Intervall von reellen Zahlen* $[\tilde{x}(1 - \varepsilon), \tilde{x}(1 + \varepsilon)[$. Der interessante Fehler ist also *relativ* zur Größe von $\tilde{x}$!

Idee: Wir untersuchen, wie sich der relative Fehler fortpflanzt, d.h. wir suchen eine Funktion $\kappa_{\text{rel}}(f,x)$, so dass

$$\frac{|f(x)-f(x+\Delta x)|}{|f(x)|} \doteq \kappa_{\text{rel}}(f,x)\frac{|\Delta x|}{|x|}\,.$$

Beobachtung: Eine solche Funktion kann man offenbar über die relative Kondition

$$\kappa_{\text{rel}}(f,x) = \kappa_{\text{abs}}(f,x)\frac{|x|}{|f(x)|}$$

erhalten.

Beispiel: Sei $f : \mathbb{R}\setminus\{0\} \to \mathbb{R}\setminus\{0\}$, $x \mapsto \frac{1}{x}$. Dann ist die relative Kondition an der Stelle x

$$\kappa_{\text{rel}}(f,x) = |x|\Big(\frac{1}{|x|}\Big)^{-1}\kappa_{\text{abs}}(f,x) = \frac{|x|^2}{|x|^2} = 1\,.$$

Der *relative Fehler* wird also (in führender Ordnung) nicht verstärkt, und die Inversion ist somit für beliebige $x \neq 0$ gut konditioniert (für $x = 0$ ist sie nicht definiert).

2.3.4 Wiederholung: Mehrdimensionale Differentiation

Definition: Die stetige Funktion $f : \mathbb{R}^n \to \mathbb{R}^m$ heißt differenzierbar im Punkt x, wenn eine Matrix $A = A(x) \in \mathbb{R}^{m\times n}$ existiert mit

$$f(x+\Delta x) = f(x) + A\Delta x + o(\Delta x)$$

wobei für die Funktion $o(y)$ gilt $\lim\limits_{y\to 0}\dfrac{\|o(y)\|_{\mathbb{R}^m}}{\|y\|_{\mathbb{R}^n}} \to 0\,.$
f heißt stetig differenzierbar, wenn die Funktion $Df : x \mapsto A(x)$ stetig ist.

Satz: Die obige Definition ist unabhängig von der Wahl der Normen $\|\cdot\|_{\mathbb{R}^n}$ in $\mathbb{R}^n$ und $\|\cdot\|_{\mathbb{R}^m}$ in $\mathbb{R}^m$.
(Dies folgt, weil in *endlich-dimensionalen* Räumen beliebige Normen gegeneinander mit multiplikativen Konstanten abschätzbar sind, siehe Übung 2.2 für Spezialfälle.)

Bezeichnung: Den Vektorraum der stetigen Funktionen von $\mathbb{R}^n$ nach $\mathbb{R}^m$ bezeichnet man mit $C^0(\mathbb{R}^n,\mathbb{R}^m)$, den Vektorraum der stetig differenzierbaren Funktionen von $\mathbb{R}^n$ nach $\mathbb{R}^m$ mit $C^1(\mathbb{R}^n,\mathbb{R}^m)$.

Satz: Für $f \in C^1(\mathbb{R}^n,\mathbb{R}^m)$ kann $Df(x)$ durch die partiellen Ableitungen der Komponentenfunktionen wie folgt ausgedrückt werden:

$$Df(x) = \Big(\frac{\partial f_i}{\partial x_j}(x)\Big)_{ij} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \cdots & \frac{\partial f_m}{\partial x_n}(x) \end{pmatrix}$$

Beispiel: Sei

$$f : \mathbb{R}^2 \to \mathbb{R}^3, \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + x_2 \\ x_1 \cdot x_2 \\ \frac{x_1}{x_2} \end{pmatrix}$$

Dann ist die Ableitungsmatrix an der Stelle $x = (x_1, x_2)^t$ gegeben als

$$Df(x) = \begin{pmatrix} 1 & 1 \\ x_2 & x_1 \\ \frac{1}{x_2} & \frac{-x_1}{x_2^2} \end{pmatrix}.$$

2.3.5 Absolute Kondition allgemeiner Funktionen

Satz: Sei $f \in C^1(\mathbb{R}^n, \mathbb{R}^m)$. Dann kann man den absoluten Fehler in der i-ten Komponente in führender Ordnung wie folgt abschätzen:

$$|f_i(x + \Delta x) - f_i(x)| \stackrel{\cdot}{\leq} \sum_{j=1}^{n} \underbrace{|\frac{\partial f_i}{\partial x_j}(x)|}_{=:\kappa_{\text{abs}}^{ij}(f,x)} |\Delta x_j|$$

Das Zeichen $\stackrel{\cdot}{\leq}$ bedeutet wieder, dass die Ungleichung „in führender Ordnung“ gilt, d.h. bis auf einen Fehler der Größe $o(\Delta x)$ mit $\frac{\|o(\Delta x)\|}{\|\Delta x\|} \to 0$.

Bezeichnung: Die Zahlen $\kappa_{\text{abs}}^{ij}(f, x)$ heißen absolute komponentenweise Konditionszahlen. Wenn klar ist, welche Funktion f gemeint ist, werden wir auch einfach $\kappa_{\text{abs}}^{ij}(x)$ schreiben.

Beispiele:

- Für die Addition $f : (x_1, x_2) \mapsto x_1 + x_2$ gilt

  $$Df(x) = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

  Somit ist $\kappa_{\text{abs}}^{11}(x) = \kappa_{\text{abs}}^{12}(x) = 1$.

- Für die Multiplikation $f : (x_1, x_2) \mapsto x_1 \cdot x_2$ ist

  $$Df(x) = \begin{pmatrix} x_2 & x_1 \end{pmatrix}$$

  Somit ist $\kappa_{\text{abs}}^{11}(x) = |x_2|$ und $\kappa_{\text{abs}}^{12}(x) = |x_1|$.

2.3.6 Relative Kondition allgemeiner Funktionen

Beobachtung: Wieder ist in vielen Fällen die relative Kondition interessanter.

Satz: Für $f \in C^1(\mathbb{R}^n, \mathbb{R}^m)$ kann man den relativen Fehler in f_i abschätzen als

$$\frac{|f_i(x+\Delta x) - f_i(x)|}{f_i(x)} \lesssim \sum_{j=1}^{n} \underbrace{|\frac{\partial f_i}{\partial x_j}(x)| \frac{|x_j|}{|f_i(x)|}}_{=:\kappa_{\text{rel}}^{ij}(f,x)} \frac{|\Delta x_j|}{|x_j|} .$$

Für die relativen komponentenweisen Konditionszahlen $\kappa_{\text{rel}}^{ij}(f,x)$ gilt also

$$\kappa_{\text{rel}}^{ij}(f,x) = \kappa_{\text{abs}}^{ij}(f,x) \frac{|x_j|}{|f_i(x)|} .$$

Beispiele:

- Für die Addition $(x_1, x_2) \mapsto x_1 + x_2$ gilt daher

$$\kappa_{\text{rel}}^{11}(x) = \frac{|x_1|}{|x_1 + x_2|} , \quad \kappa_{\text{rel}}^{12}(x) = \frac{|x_2|}{|x_1 + x_2|} .$$

- Für die Multiplikation $(x_1, x_2) \mapsto x_1 x_2$ ist

$$\kappa_{\text{rel}}^{11}(x) = |x_2| \frac{|x_1|}{|x_1 x_2|} = 1 , \quad \kappa_{\text{rel}}^{12}(x) = |x_1| \frac{|x_2|}{|x_1 x_2|} = 1 .$$

Beobachtung: Die Multiplikation vergrößert relative Fehler in ihren beiden Argumenten nicht. Sie ist immer gut konditioniert. Die Addition dagegen kann den relativen Fehler im Fall $|x_1 + x_2| \ll |x_i|$ für $i = 1, 2$ erheblich vergrößern. Diesen Effekt nennt man Auslöschung (von Genauigkeitsstellen). Er tritt bei der Addition von zwei Zahlen mit ähnlichem Betrag aber unterschiedlichen Vorzeichen auf.

Beispiel: Sei $x = \sqrt{2}$ und $\tilde{x} = \text{rd}(x)$ in IEEE doppelt genauer Gleitkomma-Arithmetik. Dann ist der *relative* Fehler von $\tilde{x}$ in der Größenordnung $\varepsilon = 10^{-16}$. Nun setzen wir $\tilde{y} = 10^8 \oplus \tilde{x}$. Der *relative* Fehler in $\tilde{y}$ ist dann immer noch in der Größenordnung $\varepsilon = 10^{-16}$. Nun ziehen wir aber wieder ab $\tilde{z} = \tilde{y} \ominus 10^8$. An dieser Stelle führt die schlechte relative Kondition der Addition zur Auslöschung von gültigen Stellen. Wie man leicht nachprüft, besitzt $\tilde{z}$ nur noch eine Genauigkeit von etwa 8 Dezimalstellen gegenüber dem mit exakter Arithmetik berechneten Wert.

Bemerkung: Die Auslöschung erklärt auch das Versagen der einfachen Reihensummation zur Berechnung von e^x für $x \ll 0$: Das Endergebnis ist viel kleiner als die zwischenzeitlich entstehenden Summanden, im schlimmsten Fall hat es überhaupt keine gültigen Stellen mehr.

2.3.7 Vektornormen

Definition: Sei V ein Vektorraum über $\mathbb{R}$. Eine Norm auf V ist eine Abbildung $V \to \mathbb{R}_0^+$ mit

- $\|x\| > 0$, genau dann wenn $0 \neq x \in V$.
- $\|\lambda x\| = |\lambda| \|x\|$ für $x \in V$, $\lambda \in \mathbb{R}$.
- Dreiecksungleichung: $\|x + y\| \leq \|x\| + \|y\|$ für $x, y \in V$.

Beispiele:

- Auf $\mathbb{R}^n$ für $1 \leq p < \infty$ die p-Normen

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

  Besonders wichtige Spezialfälle sind die euklidische Norm $\|\cdot\|_2 = (\sum_{i=1}^n |x_i|^2)^{\frac{1}{2}}$ und die Summennorm $\|x\|_1 = \sum_{i=1}^n |x_i|$.

- Die Maximumsnorm $\|x\|_\infty := \max\limits_{i=1,\ldots,n} |x_i|$ auf dem $\mathbb{R}^n$.

- Die Supremumsnorm $\|f\|_\infty := \sup\limits_{x \in \mathbb{R}} |f(x)|$ auf dem Vektorraum aller beschränkten Funktionen $f : \mathbb{R} \to \mathbb{R}$ zeigt, dass diese Konzepte auch für unendlich-dimensionale Vektorräume Sinn machen.

2.3.8 Matrixnormen/Operatornormen

Zur Definition von Normen für Matrizen $A \in \mathbb{R}^{m \times n}$ gibt es folgende Möglichkeiten:
Möglichkeit 1 Man fasst $\mathbb{R}^{m \times n}$ als Vektorraum $\mathbb{R}^{mn}$ auf und definiert darin Normen. Ein Beispiel ist die euklidische Norm $\|A\|_F := \left(\sum_{i,j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}}$, die man auch die Frobenius-Norm der Matrix A nennt.
Möglichkeit 2 Man nimmt an, dass man auf $U = \mathbb{R}^n$ eine Norm $\|\cdot\|_U$ gegeben hat und auf $V = \mathbb{R}^m$ eine Norm $\|\cdot\|_V$. Dann wird $\|A\|_{V \leftarrow U}$ definiert als die durch

$$\|A\|_{V \leftarrow U} = \sup_{0 \neq x \in U} \frac{\|Ax\|_V}{\|x\|_U} = \max_{\|x\|_U = 1} \|Ax\|_V$$

definierte Norm des durch $x \mapsto Ax$ gegebenen linearen Operators.

Bemerkung: Die Operatornormen sind normalerweise nützlicher. Dies liegt vor allem an der offensichtlichen Gültigkeit von Abschätzungen der Form

$$y = Ax \quad \Rightarrow \quad \|y\|_V \leq \|A\|_{V \leftarrow U} \|x\|_U \,.$$

Beispiele:

- Die Frobenius-Norm von $A = \begin{pmatrix} 1 & 2 \\ -3 & 4 \end{pmatrix}$ ist $\|A\|_F = \sqrt{1+4+9+16} = \sqrt{30}$.
- Die zur Maximumsnorm gehörende Operatornorm lässt sich ebenfalls einfach berechnen als
$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|$$
Für die obige Matrix ergibt sich daher der Wert $\|A\|_\infty = 7$.
- Die von der euklidischen Vektornorm $\|\cdot\|_2$ hergeleitete Spektralnorm ist definiert als
$$\|A\|_2 = \sup_{0 \neq x \in \mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2} = \max_{(x,x)=1} (Ax, Ax)^{\frac{1}{2}} = \rho(A^t A)^{\frac{1}{2}} \quad \left[= \rho(AA^t)^{\frac{1}{2}}\right]$$
wobei $(\cdot,\cdot)$ das euklidische Skalarprodukt und $\rho(\cdot)$ den Spektralradius einer Matrix bezeichnet (also den Betrag des größten Eigenwerts). Für dieselbe Matrix A wie oben erhält man
$$A = \begin{pmatrix} 1 & 2 \\ -3 & 4 \end{pmatrix} \quad \Rightarrow \quad A^t A = \begin{pmatrix} 10 & -10 \\ -10 & 20 \end{pmatrix}$$
Die Eigenwerte von $A^t A$ berechnen sich zu $\lambda_{1/2} = 15 \pm 5\sqrt{5}$, so dass $\|A\|_2 = \sqrt{15 + 5\sqrt{5}}$.
- Für *symmetrische* Matrizen gilt sogar einfach $\|A\|_2 = \rho(A)$.

2.3.9 Anwendung für Konditionsabschätzungen

Beobachtung: Anstelle von komponentenweisen Fehlerabschätzungen kann man mit Hilfe von Normen die Abschätzungen
$$\|f(x+\Delta x) - f(x)\|_V \;\dot{\leq}\; \|Df(x)\|_{V \leftarrow U} \|\Delta x\|_U$$
und
$$\frac{\|f(x+\Delta x) - f(x)\|_V}{\|f(x)\|_V} \;\dot{\leq}\; \frac{\|Df(x)\|_{V \leftarrow U} \|x\|_U}{\|f(x)\|_V} \frac{\|\Delta x\|_U}{\|x\|_U}$$
erhalten. Die skalaren Größen
$$\kappa_{\text{abs}}(f,x) := \|Df(x)\|_{V \leftarrow U}\,, \quad \kappa_{\text{rel}}(f,x) := \frac{\|Df(x)\|_{V \leftarrow U} \|x\|_U}{\|f(x)\|_V}$$
bezeichnen dann wieder absolute bzw. relative Kondition.

2.3.10 Stabilität von Algorithmen

Beobachtung: Die Berechnung von e^x für $x \ll 0$ ist im Prinzip gut konditioniert! Es gilt ja

$$\kappa_{\text{rel}}(e^x, x) = e^x \frac{|x|}{e^x} = |x|$$

was nicht sehr schlimm ist. Schließlich konnten wir e^x ja auch für $x \ll 0$ mittels $\exp(x) = \frac{1}{\exp(-x)}$ problemlos berechnen.

Frage: Wo liegt dann der Grund für das schlechte Verhalten unseres Programms?

Antwort: Das Problem liegt darin, dass ein Algorithmus zur Berechnung von $f : \mathbb{R}^n \to \mathbb{R}^m$ diese Funktion als Verkettung

$$\mathbb{R}^n = \mathbb{R}^{k_0} \xrightarrow{f_1} \mathbb{R}^{k_1} \xrightarrow{f_2} \mathbb{R}^{k_2} \xrightarrow{f_3} \dots \xrightarrow{f_N} \mathbb{R}^{k_N} = \mathbb{R}^m$$

berechnet. In jedem Schritt können Fehler durch Rundung entstehen, die beliebig verstärkt werden können, wenn irgendwelche der f_i schlecht konditioniert sind.

Definition: Ein Algorithmus heißt stabil, wenn der durch akkumulierte Rundungsfehler entstehende Fehler in derselben Größenordnung wie der Fehler liegt, der durch die Kondition des Problems zu erwarten ist. Dies ist insbesondere der Fall, wenn alle $f_1, \dots, f_N$ in der obigen Zerlegung gut konditioniert sind.

3 Lineare Gleichungssysteme

3.1 Problemdefinition

Problem: Gegeben sei eine Matrix $A \in \mathbb{R}^{n \times n}$ und ein Vektor $b \in \mathbb{R}^n$. Finde ein $x \in \mathbb{R}^n$, so dass

$$Ax = b\,.$$

Dies nennt man ein lineares Gleichungssystem (LGS).

Bezeichnung: Mit Φ_A bezeichnen wir die von A beschriebene lineare Abbildung

$$\Phi_A : \mathbb{R}^n \to \mathbb{R}^n\,, \quad x \mapsto Ax\,.$$

Satz: Eine *eindeutige* Lösung des LGS $Ax = b$ mit $A \in \mathbb{R}^{n \times n}$ und $b \in \mathbb{R}^n$ existiert genau dann, wenn eine der folgenden äquivalenten Charakterisierungen gilt:

1. Φ_A ist injektiv, d.h. $\mathrm{Kern}(\Phi_A) = \{x : Ax = 0\} = \{0\}$
2. Φ_A ist surjektiv, d.h. $\mathrm{Bild}(\Phi_A) = \mathbb{R}^n$
3. Die Determinante $\det(A)$ ist nicht Null.

Bezeichnung: Wenn eine Bedingung des Satzes zutrifft (und damit alle Bedingungen gelten), nennt man A und Φ_A regulär.

Bemerkung: Leider ist der obige Satz in der Praxis nicht direkt anwendbar, weil er nicht robust gegen Störungen der Daten ist.

Beispiel: Das System

$$Ax = b\,, \quad A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\,, \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

ist nicht lösbar. Sobald man aber A in einem beliebigen Element um einen beliebig kleinen Wert $\varepsilon \neq 0$ stört, wird es plötzlich lösbar:

$$\begin{pmatrix} 1+\varepsilon & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad \Leftrightarrow \quad x = \underbrace{\frac{1}{\varepsilon}\begin{pmatrix} 1 & -1 \\ -1 & 1+\varepsilon \end{pmatrix}}_{=A^{-1}} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \frac{1}{\varepsilon}\begin{pmatrix} -1 \\ 1+2\varepsilon \end{pmatrix}$$

Bemerkung: Man beachte, dass die Inverse und die berechnete Lösung von der Ordnung $O(\frac{1}{\varepsilon})$ sind.

Erinnerung: Für $A \in \mathbb{R}^{2\times 2}$ berechnet sich A^{-1} als

$$A^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} .$$

Beweis: Ausmultiplizieren.

3.2 Die Kondition der Matrixmultiplikation

Bemerkungen:

- Obwohl wir eigentlich am Lösen des LGS $Ax = b$ interessiert sind (also an der Abbildung $b \mapsto x = A^{-1}b$), ist es sinnvoll, sich zuerst das direkte Problem $x \mapsto b = Ax$ anzuschauen.
- Im folgenden werden wir Fehler mit Hilfe von Normen messen, also nicht komponentenweise. Dies ist für allgemeine Störungen von Matrizen, rechten Seiten und Lösungen meist die richtige Wahl. Nur für spezielle Gleichungssysteme kann eine komponentenweise Analyse sinnvoller sein.
- Mit $\|\cdot\|$ bezeichnen wir im folgenden sowohl eine beliebige Vektornorm im $\mathbb{R}^n$ als auch (wenn das Argument eine Matrix ist) die zugehörige Operatornorm.
  Beispiel: $\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$ und $\|A\|_\infty = \sup_{x\neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty}$.

3.2.1 Abschätzung des absoluten Fehlers

Satz: Es gelte $b = Ax$. Nach Störung von A mit ΔA und x mit Δx erhalten wir den gestörten Wert $b + \Delta b = (A + \Delta A)(x + \Delta x)$. Der Fehler Δb kann dann abgeschätzt werden als

$$\|\Delta b\| \;\dot{\leq}\; \|A\|\|\Delta x\| + \|\Delta A\|\|x\| .$$

Beweis: Ausmultiplizieren der Bestimmungsgleichung führt zu:

$$\begin{aligned} b + \Delta b &= Ax + \Delta A \cdot x + A \cdot \Delta x + \Delta A \cdot \Delta x \qquad \Leftrightarrow \\ \Delta b &= \Delta A \cdot x + A \cdot \Delta x + \Delta A \cdot \Delta x \end{aligned}$$

Wenn man nun zu Normen übergeht und den quadratischen Fehlerterm $\|\Delta A\|\|\Delta x\|$ vernachlässigt, erhält man die Behauptung.

3.2.2 Abschätzung des relativen Fehlers

Beobachtung: Offensichtlich kommen bei der Matrix-Multiplikation

$$b_i = \sum_{j=1}^{n} A_{ij} x_j$$

Additionen vor. Daher ist zu erwarten, dass sich die schlechte relative Kondition der Addition auch hier zeigen kann.

Satz: A sei *invertierbar*, und es gelte $b = Ax$ sowie $b + \Delta b = (A + \Delta A)(x + \Delta x)$. Dann gilt

$$\frac{\|\Delta b\|}{\|b\|} \dot{\leq} \underbrace{\|A\|\|A^{-1}\|}_{=:\kappa(A)} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta x\|}{\|x\|} \right) .$$

Beweis: Aus der Abschätzung für den absoluten Fehler erhalten wir

$$\begin{aligned} \|\Delta b\| &\dot{\leq} \|A\|\|\Delta x\| + \|\Delta A\|\|x\| \\ &\leq (\|A\|\|\Delta x\| + \|\Delta A\|\|x\|) \frac{\|A^{-1}\|\|b\|}{\|x\|} \\ &\leq \|A\|\|A^{-1}\| \left(\frac{\|\Delta x\|}{\|x\|} + \frac{\|\Delta A\|}{\|A\|} \right) \|b\| . \end{aligned}$$

Hieraus folgt nach Division durch $\|b\|$ die Behauptung.

Definition: Die Größe $\kappa(A) = \|A\|\|A^{-1}\|$ beschreibt offenbar die Fortpflanzung von relativen Fehlern sowohl in der Matrix als auch der rechten Seite. Man nennt sie die Kondition der Matrix A. Sie hängt auch von der gewählten Norm ab. Die Kondition einer Matrix A bezüglich der Norm $\|\cdot\|_p$ bezeichnen wir mit $\kappa_p(A)$.

Beispiel: Betrachte die lineare Abbildung

$$b = Ax\,, \quad A = \begin{pmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{pmatrix}\,, \quad x = \begin{pmatrix} 1 \\ -1 \end{pmatrix} .$$

Hier ist $\|A\|_\infty = 2 + \varepsilon$ und wegen $A^{-1} = \frac{1}{\varepsilon} \begin{pmatrix} 1+\varepsilon & -1 \\ -1 & 1 \end{pmatrix}$ gilt $\|A^{-1}\|_\infty = \frac{2+\varepsilon}{\varepsilon}$. Somit ist $\kappa_\infty(A) = \frac{(2+\varepsilon)^2}{\varepsilon} \approx \frac{4}{\varepsilon}$ für $\varepsilon \ll 1$.
Für $\varepsilon \approx 10^{-8}$ und einer Genauigkeit von $eps = 10^{-16}$ in x ergibt sich beispielsweise eine Genauigkeit von nur etwa acht Dezimalstellen in b.

3.3 Die Kondition der Lösung linearer Gleichungssysteme

Bemerkung: Weil normalerweise der relative Fehler bei der Lösung von linearen Gleichungssystemen von größerem Interesse als der absolute Fehler ist, beschränken wir uns im folgenden auf diesen.

3.3.1 Variation der rechten Seite

Satz: Es gelte

$$Ax = b\,, \quad A(x + \Delta x) = b + \Delta b\,.$$

Dann gilt für den relativen Fehler die Abschätzung

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\Delta b\|}{\|b\|}\,.$$

Beweis: Die Lösung eines LGS ist gleichbedeutend mit der Matrixmultiplikation $b \mapsto A^{-1}b$. Die Ergebnisse für die Matrixmultiplikation übertragen sich daher sofort. Man beachte auch, dass $\kappa(A) = \|A\|\|A^{-1}\| = \kappa(A^{-1})$.

3.3.2 Variation der Matrix

Lemma: Für eine Matrix $B \in \mathbb{R}^{n\times n}$ gelte $\|B\| < 1$. Dann ist $\mathrm{Id} + B$ invertierbar, und es gilt die Fehlerabschätzung

$$\|(\mathrm{Id} + B)^{-1}\| \leq \frac{1}{1 - \|B\|}\,.$$

Für $\|B\| \leq \delta < 1$ gilt mit einem Fehler der Größe $O(\|B\|^2)$

$$(\mathrm{Id} + B)^{-1} \doteq \mathrm{Id} - B$$

Beweis: Wegen $\|B\| < 1$ gilt

$$\|(\mathrm{Id} + B)x\| = \|x + Bx\| \geq \|x\| - \|Bx\| \geq \|x\| - \|B\|\|x\| = (1 - \|B\|)\|x\|\,.$$

Somit folgt aus $\|x\| \neq 0$ auch $\|(\mathrm{Id} + B)x\| \neq 0$, so dass $I + B$ injektiv und damit regulär ist. Die Normabschätzung folgt aus der Operatornorm-Definition

$$\|(\mathrm{Id} + B)^{-1}\| = \sup_{0\neq y\in\mathbb{R}^n} \frac{\|(\mathrm{Id} + B)^{-1}y\|}{\|y\|} = \cdots$$

was mit Hilfe der Variablentransformation $y = (\mathrm{Id} + B)x$ umgeformt wird in

$$\cdots = \sup_{0\neq x\in\mathbb{R}^n} \frac{\|x\|}{\|(\mathrm{Id} + B)x\|} \leq \sup_{0\neq x\in\mathbb{R}^n} \frac{\|x\|}{(1 - \|B\|)\|x\|} = \frac{1}{1 - \|B\|}$$

Die Approximation $(\mathrm{Id} + B)^{-1} \doteq \mathrm{Id} - B$ folgt, weil man $(I + B)^{-1}$ darstellen kann als die absolut konvergente Reihe $\sum_{k=0}^{\infty}(-X)^k = I - X + X^2 - X^3 \pm \cdots$.

Satz: Es gelte

$$Ax = b \quad \text{und} \quad (A + \Delta A)(x + \Delta x) = b$$

Dann gilt für $\|A^{-1}\Delta A\| \leq \delta < 1$

$$\frac{\|\Delta x\|}{\|x\|} \;\dot{\leq}\; \kappa(A)\frac{\|\Delta A\|}{\|A\|}$$

wobei der Fehler von der Ordnung $O(\|A^{-1}\Delta A\|^2)$ ist.

Beweis:

$$\begin{aligned}(A+\Delta A)(x+\Delta x) = b \quad &\Leftrightarrow \quad x+\Delta x = (A+\Delta A)^{-1}b \\ &\Leftrightarrow \quad x+\Delta x = (\mathrm{Id}+A^{-1}\Delta A)^{-1}A^{-1}b\end{aligned}$$

Die Anwendung des Lemmas mit $B = A^{-1}\Delta A$ liefert nun

$$x+\Delta x \doteq (\mathrm{Id}-A^{-1}\Delta A)x \quad \Leftrightarrow \quad \Delta x \doteq -A^{-1}\Delta A\, x$$

Übergang zu Normen führt dann zur Abschätzung

$$\|\Delta x\| \;\dot{\leq}\; \|A^{-1}\|\|\Delta A\|\|x\| \quad \Leftrightarrow \quad \frac{\|\Delta x\|}{\|x\|} \;\dot{\leq}\; \underbrace{\|A^{-1}\|\|A\|}_{\kappa(A)}\frac{\|\Delta A\|}{\|A\|}\,.$$

Bemerkung: Die Einschränkung $\|A^{-1}\Delta A\| \leq \delta < 1$ ist nicht wesentlich: Schließlich gilt $\|A^{-1}\Delta A\| \leq \kappa(A)\frac{\|\Delta A\|}{\|A\|}$, und diese Größe sollte für eine nützliche Anwendung des Satzes sowieso klein sein.

3.4 Das Gaußsche Eliminationsverfahren

Das Gaußsche Eliminationsverfahren löst ein lineares Gleichungssystem, indem dieses durch elementare Umformungen auf eine obere Dreiecksgestalt gebracht wird. Die erlaubten Umformungen sind dabei:

1. Addition des Vielfachen einer Gleichung zu einer anderen
2. Vertauschen zweier Gleichungen

Bemerkung: Auch in der Numerik ist dieses Verfahren das „Arbeitspferd“, um lineare Gleichungssysteme moderater Größe zu lösen, allerdings in abgewandelter Form:

- Es wird eine *Zerlegung* der Matrix A durchgeführt. Das Ergebnis ist eine obere Dreiecksmatrix, es werden aber zusätzlich auch die Transformationsschritte in einer unteren Dreiecksmatrix und einem Vertauschungsvektor abgespeichert.
- Mit dieser Zerlegung wird dann in einem zweiten Schritt die rechte Seite transformiert und die Lösung berechnet.

Dies erlaubt insbesondere die Wiederverwendung der Zerlegung bei einer Änderung der rechten Seite.

3.4.1 Klassisches Vorgehen

Beispiel: Die Transformation des Systems

$$\begin{aligned} 2x_1 - 2x_2 + 3x_3 &= 1 \\ -4x_1 + 4x_2 + 3x_3 &= 1 \\ 3x_1 - 2x_2 + 1x_3 &= -1 \end{aligned}$$

auf Dreiecksgestalt kann man wie folgt durchführen:

$$\underbrace{\left(\begin{array}{ccc|c} 2 & -2 & 3 & 1 \\ -4 & 4 & 3 & 1 \\ 3 & -2 & 1 & -1 \end{array}\right)}_{=(A^{(0)}|b^{(0)})} \sim \underbrace{\left(\begin{array}{ccc|c} 2 & -2 & 3 & 1 \\ 0 & 0 & 9 & 3 \\ 0 & 1 & -\frac{7}{2} & -\frac{5}{2} \end{array}\right)}_{=(A^{(1)}|b^{(1)})} \sim \underbrace{\left(\begin{array}{ccc|c} 2 & -2 & 3 & 1 \\ 0 & 1 & -\frac{7}{2} & -\frac{5}{2} \\ 0 & 0 & 9 & 3 \end{array}\right)}_{=(A^{(2)}|b^{(2)})}$$

Rückwärtssubstitution liefert dann die Lösung

$$\begin{aligned} 9x_3 = 3 \quad &\Rightarrow \quad x_3 = \frac{1}{3} \\ x_2 - \frac{7}{2}x_3 = -\frac{5}{2} \quad &\Rightarrow \quad x_2 = -\frac{5}{2} + \frac{7}{6} = \frac{4}{6} = -\frac{4}{3} \\ 2x_1 - 2x_2 + 3x_3 = 1 \quad &\Rightarrow \quad x_1 = \frac{1}{2}(1 - \frac{8}{3} - 1) = -\frac{4}{3} \end{aligned}$$

Bemerkung: In Matlab/Scilab/Octave ist der Befehl zur Lösung eines linearen Gleichungssystems mit Matrix A und rechter Seite b der Backslash-Operator \ (Idee: b wird durch A dividiert, die Reihenfolge der Operanden ist wie in $Ax = b$).
So löst etwa folgende Befehlssequenz das obige Beispiel:

```
A=[2,-2,3;-4,4,3;3,-2,1];  b=[1;1;-1];  A\b
```

3.4.2 Motivation der LR-Zerlegung

Beobachtung:

- Die elementare Operation „Addiere ein α-faches von Zeile k zu Zeile j“ mit $k < j$ kann durch die Multiplikation mit einer unteren Dreiecksmatrix erreicht werden, bei der die Diagonale aus Einsen besteht und nur das Element $a_{jk} = \alpha \neq 0$. Mehrere solcher Operationen liefern dann eine allgemeine untere Dreiecksmatrix.
- Die elementare Operation „Vertausche Zeile k mit Zeile l“ sowie die Verkettung beliebig vieler solcher Vertauschungen ergibt sich als Multiplikation mit einer Permutationsmatrix P. Dies ist eine Matrix, bei der in jeder Zeile und jeder Spalte genau eine 1 steht, ansonsten aber nur Nullen.

Folgerung: Das Gaußsche Eliminationsverfahren berechnet also eine sogenannte LR-Zerlegung

$$PA = LR$$

der Matrix PA. Hierbei ist P eine geeignete Permutationsmatrix, L eine linke untere Dreiecksmatrix mit Einsen auf der Diagonale und R eine rechte obere Dreiecksmatrix.

3.4.3 Die LR-Zerlegung anhand eines Beispiels

Beispiel: Wir berechnen die LR-Zerlegung, indem wir die entstehenden Nullen in der Restmatrix durch den Multiplikationsfaktor $L_{jk} = \frac{A_{jk}^{(k)}}{A_{kk}^{(k)}}$ (rot) ersetzen. Die Permutationsmatrix P führen wir als Indexvektor (blau) ebenfalls mit.

Schritt 1
$$\left(\begin{array}{ccc||c} 2 & -2 & 3 & 1 \\ -4 & 4 & 3 & 2 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

Schritt 2
$$\left(\begin{array}{ccc||c} 2 & -2 & 3 & 1 \\ -2 & 0 & 9 & 2 \\ \frac{3}{2} & 1 & -\frac{7}{2} & 3 \end{array}\right)$$

Schritt 3
$$\left(\begin{array}{ccc||c} 2 & -2 & 3 & 1 \\ \frac{3}{2} & 1 & -\frac{7}{2} & 3 \\ -2 & 0 & 9 & 2 \end{array}\right)$$

Schritt 4
$$\left(\begin{array}{ccc||c} 2 & -2 & 3 & 1 \\ \frac{3}{2} & 1 & -\frac{7}{2} & 3 \\ -2 & 0 & 9 & 2 \end{array}\right)$$

Hinweis: Man beachte, dass in Schritt 3 ganze Zeilen (inklusive der L-Matrix-Elemente (rot) und der Indizes (blau)) vertauscht wurden.

Interpretation: Diese Kurzschreibweise muss interpretiert werden als die Zerlegung

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{P} \underbrace{\begin{pmatrix} 2 & -2 & 3 \\ -4 & 4 & 3 \\ 3 & -2 & 1 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} 2 & -2 & 3 \\ 0 & 1 & -\frac{7}{2} \\ 0 & 0 & 9 \end{pmatrix}}_{R}$$

Die Permutationsmatrix P berechnet sich aus dem mitgeführten Vektor $p = (p_1, \ldots, p_n)^t$ gemäß

$$P_{ij} = \begin{cases} 1 & j = p_i \\ 0 & \text{sonst} \end{cases}.$$

Bemerkung: Eine Überprüfung mit Scilab ergibt tatsächlich, dass die Zerlegung gültig ist:

```
A = [2,-2,3;-4,4,3;3,-2,1]
P = [1,0,0;0,0,1;0,1,0]
L = [1,0,0;3/2,1,0;-2,0,1]
R = [2,-2,3;0,1,-7/2;0,0,9]
P*A
L*R
```

Bemerkung: Scilab kann auch selber eine LR-Zerlegung $PA = LR$ berechnen. Der Aufruf dazu lautet

```
[L,U,p] = lu([2,-2,3;-4,4,3;3,-2,1])
```

Allerdings beobachten wir ein anderes Ergebnis, weil Scilab aus Stabilitätsgründen eine andere Permutationsmatrix P wählt. (Der Name `lu` kommt von *lower-upper*. Man beachte auch die Syntax der Rückgabe mehrerer Funktionswerte.)

3.4.4 Verwendung der LR-Zerlegung

Anwendung: Mit der Zerlegung $PA = LR$ können wir das LGS umschreiben:

$$Ax = b \quad \Leftrightarrow \quad LRx = PAx = Pb$$

Diese Form kann wie folgt verwendet werden:

1. Löse $Lz = Pb$.
2. Löse $Rx = z$.

Bemerkung: Die Auflösung der beiden Dreieckssysteme ist einfach und geschieht durch Vorwärtssubstitution im ersten Schritt (die Lösung wird von der ersten Komponente ausgehend sukzessive berechnet), bzw. durch Rückwärtssubstitution im zweiten Schritt (die Lösung wird von der letzten Komponente ausgehend berechnet).

Beispiel: Der erste Schritt des Algorithmus angewendet auf unser Beispielsystem ist

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}}_{L} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{P} \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -\frac{5}{2} \\ 3 \end{pmatrix}.$$

Der zweite Schritt ist dann

$$\underbrace{\begin{pmatrix} 2 & -2 & 3 \\ 0 & 1 & -\frac{7}{2} \\ 0 & 0 & 9 \end{pmatrix}}_{R} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -\frac{5}{2} \\ 3 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(1 - 2 \cdot \frac{4}{3} + 1) \\ -\frac{5}{2} + \frac{7}{2} \cdot \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} = \begin{pmatrix} -\frac{4}{3} \\ -\frac{4}{3} \\ \frac{1}{3} \end{pmatrix}$$

Bemerkung: Der Nutzen dieser Vorgehensweise (zuerst zerlegen, dann auflösen) liegt in folgenden Punkten:

- Für große Gleichungssysteme ist der Aufwand für die Zerlegung von der Ordnung $O(n^3)$, der Aufwand für die Auflösung der Dreieckssysteme aber nur von der Ordnung $O(n^2)$. Bei der Anwendung auf mehrere rechte Seiten ergeben sich daher erhebliche Einsparungen.

- Die Zerlegung kann auch noch für weitere wichtige Operationen verwendet werden. So ist zum Beispiel die Determinante von A das Produkt der Diagonalelemente von R multipliziert mit der leicht zu berechnenden Determinante von P (die ist einfach das Signum der entsprechenden Permutation).
- Die Analyse des Algorithmus wird einfacher und klarer, weil eine Beschreibung in Matrixsprache vorliegt.

3.4.5 Theorie der LR-Zerlegung ohne Zeilentausch

Satz: Die invertierbaren unteren (bzw. oberen) $n \times n$-Dreiecksmatrizen bilden eine Gruppe bezüglich der Matrix-Multiplikation. Dasselbe gilt für untere (bzw. obere) Dreiecksmatrizen mit Einsen auf der Diagonale.

Beweis: Abgeschlossenheit: Seien A, B linke untere Dreiecksmatrizen. Dann gilt für $C = AB$, dass $C_{ik} = \sum_{j=1}^{n} A_{ij}B_{jk} = 0$, wenn $i < k$. Die Inverse einer unteren Dreiecksmatrix L ist ebenfalls eine untere Dreiecksmatrix (man kann die Einträge explizit aus denen von L berechnen).

Satz: Wenn für $A \in \mathbb{R}^{n\times n}$ die Zerlegung $A = LR$ mit einer unteren Dreiecksmatrix L, bei der auf der Diagonalen Einsen stehen, und einer rechten oberen Dreiecksmatrix R existiert, dann ist sie eindeutig.

Beweis: Wenn es zwei Zerlegungen gäbe $A = L_1R_1 = L_2R_2$, so wäre $L_2^{-1}L_1 = R_2R_1^{-1}$ sowohl obere als auch untere Dreiecksmatrix, also eine Diagonalmatrix. $L_2^{-1}L_1$ hat zudem nur Einsen auf der Diagonale, also muss es die Einheitsmatrix sein, d.h. $L_1 = L_2$ und $R_1 = R_2$.

Beobachtung: Nicht jede invertierbare Matrix A besitzt eine Zerlegung $A = LR$. Das einfachste Gegenbeispiel ist

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

Eine LR-Zerlegung erfüllt immer $A_{11} = R_{11}$, so dass hier $R_{11} = 0$ und somit R nicht invertierbar wäre. Dies wäre aber ein Widerspruch zur Invertierbarkeit von A.

3.4.6 Die LR-Zerlegung für symmetrisch positiv-definite Matrizen

Definition: Eine Matrix A heißt symmetrisch, wenn $A = A^t$. Sie heißt symmetrisch positiv definit (spd), wenn sie symmetrisch ist und zusätzlich alle ihre Eigenwerte positiv sind (reell sind sie sowieso wegen der Symmetrie).

Satz: A sei spd. Dann existiert eine LR-Zerlegung $A = LR$ (ohne dass ein Zeilentausch notwendig ist!). Wenn D die Diagonale von R ist, so gilt $R = DL^t$, so dass die Zerlegung auch als $A = LDL^t$ geschrieben werden kann.

Beweisidee: Wenn man in der symmetrischen Gestalt $A = LDL^t$ zerlegt, so kann man zeigen, dass die Restmatrix immer symmetrisch positiv definit bleibt.

Bemerkung: Die sogenannte Cholesky-Zerlegung $A = \tilde{L}\tilde{L}^t$ entsteht aus der LR-Zerlegung $A = LR$, wenn man $\tilde{L} = L\sqrt{D}$ setzt, wobei D die Diagonale von R ist. $\tilde{L}$ ist hier eine untere Dreiecksmatrix, hat aber normalerweise keine Einsen auf der Diagonalen. $\sqrt{D} = \mathrm{diag}(\sqrt{D_{11}}, \ldots, \sqrt{D_{nn}})$.

3.4.7 Theorie der LR-Zerlegung mit Zeilentausch

Satz: Jede reguläre Matrix besitzt eine Zerlegung $PA = LR$.

Beweisskizze: Der anfangs skizzierte Algorithmus zur Berechnung einer solchen Zerlegung bricht nur dann ab, wenn eine Spalte der Restmatrix Null ist. Dann ist aber die Restmatrix singulär. Da diese nur durch reguläre Transformationen aus A hervorgeht, muss auch schon A singulär gewesen sein.

3.4.8 Implementation der LR-Zerlegung

Programm: (LR-Zerlegung) Folgendes Scilab-Programm berechnet die LR-Zerlegung nach dem anfangs anhand eines Beispiels beschriebenen Algorithmus. Das Ergebnis sind L und R, in einer Matrix gespeichert.

```
function A=LR(A0)
    A=A0;
    n = size(A,1);
    for i=1:n
        if A(i,i)==0
            error('Matrix singular or pivoting necessary');
        else
            for j=i+1:n
                A(j,i)=A(j,i)/A(i,i);
                for k=i+1:n
                    A(j,k)=A(j,k)-A(j,i)*A(i,k);
                end
            end
        end
    end
endfunction
```

Programm: (LR-Zerlegung, Version 2) Wir ersetzen die innerste Schleife durch eine Vektor-Operation:

```
function A=LR2(A0)
    A=A0;
    n = size(A,1);
    for i=1:n
```

```
        if A(i,i)==0
            error('Matrix singular or pivoting necessary');
        else
            for j=i+1:n
                A(j,i)=A(j,i)/A(i,i);
                A(j,i+1:n)=A(j,i+1:n)-A(j,i)*A(i,i+1:n);
            end
        end
    end
endfunction
```

Programm: (LR-Zerlegung, Version 3) Wir ersetzen eine weitere Schleife durch Verwendung von Vektor- und Matrixoperationen:

```
function A=LR3(A0)
    A=A0;
    n = size(A,1);
    for i=1:n
        if A(i,i)==0
            error('Matrix singular or pivoting necessary');
        else
            A(i+1:n,i)=A(i+1:n,i)/A(i,i);
            A(i+1:n,i+1:n)=A(i+1:n,i+1:n)-A(i+1:n,i)*A(i,i+1:n);
        end
    end
endfunction
```

3.4.9 Aufwandsbetrachtungen

Beobachtung: Man sieht sofort, dass der Schleifenkörper von `LR3` den Aufwand von jeweils $(n-i)^2$ Additionen und Multiplikationen hat (und dazu noch $(n-i)$ Divisionen, die wir aber vernachlässigen). Der Hauptteil des Aufwands besteht daher aus

$$\sum_{i=1}^{n-1}(n-i)^2 = \sum_{k=1}^{n-1} k^2 = \frac{n^3}{3} + O(n^2)$$

Additionen und Multiplikationen.

Bezeichnung: Da bei Matrizenoperationen oft gleich viel Additionen und Multiplikationen auftreten, definiert man eine arithmetische Operation AO als 1 Addition + 1 Multiplikation. Damit hat die Gauß-Zerlegung in führender Ordnung $\frac{n^3}{3}$ AO.

Experiment: Wir prüfen dies mit folgendem Konstrukt zur Zeitmessung nach:

```
n=100; tic; LR(eye(n,n)); toc
```

Auf einem Pentium-Laptop (DualCore, ca. 2GHz) ergeben sich folgende Zeiten (in Sekunden):

n	100	200	400	800
Scilab/LR	6.7	50.6	-	-
Scilab/LR2	0.35	1.48	8.14	51.3
Scilab/LR3	0.07	0.24	1.76	12.7
Matlab/LR	0.06	0.27	1.78	14.8
Matlab/LR2	0.12	0.41	2.36	15.2
Matlab/LR3	0.02	0.15	1.06	8.65
Scilab/lu	0.01	0.05	0.30	1.92
Matlab/lu	0.01	0.01	0.07	0.35

Bemerkungen:

- Matrix- und Vektoroperationen können vor allem in Scilab/Octave erheblich Zeit sparen gegenüber den interpretierten Schleifen. Die Matlab-Sprache ist allerdings erstaunlich schnell.
- Die `lu`-Funktion (wenigstens die von Matlab) ist um ein Vielfaches schneller als die naiven Implementationen. Der Hauptvorteil kommt dabei von einer komplizierteren Schleifenstruktur, die blockweise arbeitet. Dadurch wird eine viel bessere Ausnutzung des Cache erreicht.
- Solche Cache-Optimierung ist in der numerischen linearen Algebra generell wichtig, wenn viele Operationen (z.B. $O(n^3)$) auf relativ wenig Daten (z.B. $O(n^2)$) ausgeführt werden. Beispiele: Matrix-Matrix-Multiplikation, LR-Zerlegung, ...
- Cache-optimierte lineare Algebra ist auch für andere Sprachen als Matlab verfügbar (ATLAS).

3.4.10 Die Stabilität der LR-Zerlegung

Beispiel: Ohne geeigneten Zeilentausch ist die LR-Zerlegung in vielen Fällen instabil. Wenn man etwa die Matrix

$$A = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix}$$

betrachtet, so ist deren LR-Zerlegung in exakter Arithmetik

$$L = \begin{pmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{pmatrix}$$

und die Kondition dieser beider Matrizen ist sehr schlecht (was dann wiederum zu großen Fehlern bei der Lösung von Gleichungssystemen führen kann).

Idee: Suche in jedem Schritt in der ersten Spalte der Restmatrix nach dem betragsgrößten Element und tausche die entsprechende Zeile an die aktuelle Position. Dadurch werden die in dieser Spalte entstehenden Elemente von L möglichst klein gehalten.

Bezeichnung: Den entstehenden Algorithmus nennt man die LR-Zerlegung mit Spaltenpivotsuche.

Bemerkungen:

- Die LR-Zerlegung mit Spaltenpivotsuche arbeitet in sehr vielen Fällen äußerst zuverlässig. Sie wird daher auch beim Matlab/Scilab/Octave-Befehl `lu` verwendet, ebenso wie beim Backslash-Lösungsoperator.
- Man kann jedoch „pathologische“ Matrizen $A \in \mathbb{R}^{n\times n}$ konstruieren, deren Kondition gut ist, für welche jedoch die LR-Zerlegung $PA = LR$ mit dem durch Spaltenpivotsuche erhaltenen P nicht stabil ist (genauer gesagt, die Kondition von R ist von der Ordnung $O(2^n)$).
- In solchen Fällen (oder wenn bereits die Kondition der Matrix A sehr schlecht ist) verzichtet man am besten ganz auf die LR-Zerlegung und verwendet statt dessen die im folgenden Kapitel beschriebenen orthogonalen Zerlegungen zum Lösen des Gleichungssystems.

4 Lineare Ausgleichsprobleme

4.1 Ein eindimensionales Problem

Aufgabe: Ein Physiker will den Widerstand R eines Drahts unter Verwendung des Gesetzes

$$U = RI$$

messen. Er legt daher verschiedene Spannungen U_k an und misst die entsprechenden Stromstärken I_k. Er erhält folgende Tabelle:

k	1	2	3	4
U_k [V]	10	20	30	40
I_k [A]	0.16	0.31	0.45	0.6

All diese Messungen sind natürlich mit einem Fehler behaftet. Was ist der beste Wert für R?

Frage: Was heißt „beste“?

Antwort: Eine mögliche Form der Quantifizierung ist folgende: Man fügt alle Spannungen U_k zu einem Vektor $\vec{U} \in \mathbb{R}^4$ zusammen, ebenso alle Stromstärken I_k zu einem Vektor $\vec{I} \in \mathbb{R}^4$ und sucht dann ein $R \in \mathbb{R}$, so dass

$$F(R) = \|\vec{U} - R\vec{I}\|$$

minimal wird. Hierbei soll $\|\cdot\|$ irgendeine Vektornorm im $\mathbb{R}^4$ sein.

Spezialisierung: Jetzt und im folgenden betrachten wir den einfachsten Fall $\|\cdot\| = \|\cdot\|_2$. Ferner suchen wir statt dem Minimum von $F(R)$ das Minimum der differenzierbaren Funktion $\Phi(R) = \frac{1}{2}F(R)^2$ (das liefert dasselbe, weil $F(R) > 0$). Es gilt

$$\Phi(R) = \frac{1}{2}(\vec{U} - R\vec{I}, \vec{U} - R\vec{I}) = \frac{1}{2}(\vec{U}, \vec{U}) - (\vec{I}, \vec{U})R + \frac{1}{2}(\vec{I}, \vec{I})R^2$$

Hierbei bezeichnet (x, y) immer das Euklidische Skalarprodukt (hier im $\mathbb{R}^m$).

Satz: Das Minimum der obigen quadratischen Funktion erhält man durch Lösen der Gleichung

$$\Phi'(R) = (\vec{I}, \vec{I})R - (\vec{I}, \vec{U}) = 0\,.$$

Beispiel: Für die Zahlen des obigen Beispiels erhält man (in $\mathrm{Ohm} = \frac{\mathrm{Volt}}{\mathrm{Ampere}}$)

$$R = \frac{(\vec{I}, \vec{U})}{(\vec{I}, \vec{I})} = \frac{1.6 + 6.2 + 13.5 + 24}{0.16^2 + 0.31^2 + 0.45^2 + 0.6^2} \approx 66.208711$$

Bemerkung: Fast immer muss in Anwendungen nicht nur ein Parameter (hier R) optimiert werden, sondern mehrere Parameter gleichzeitig. Daher müssen wir unsere Formulierung des Problems verallgemeinern.

4.2 Mehrdimensionale Probleme

4.2.1 Problem der kleinsten Fehlerquadrate

Problem: (Problem der kleinsten Fehlerquadrate, least squares problem) Sei $A \in \mathbb{R}^{m \times n}$ mit $\mathrm{Rang}(A) = n$ (woraus folgt $m \geq n$). Zu $b \in \mathbb{R}^m$ suchen wir $x \in \mathbb{R}^n$, so dass die Funktion

$$\Phi : \mathbb{R}^n \mapsto \mathbb{R}\,, \quad x \mapsto \frac{1}{2}\|Ax - b\|^2$$

minimal wird.

Satz: Die Lösung $x \in \mathbb{R}^n$ des obigen Problems ist eindeutig bestimmt. Sie erfüllt die sogenannte Normalgleichung

$$\nabla\Phi(x) = A^t A x - A^t b = 0\,.$$

Beweis: Diese Bedingung ist notwendig für ein Extremum (Analysis). Eine eindeutige Lösung $x \in \mathbb{R}^n$ existiert, weil $A^t A$ symmetrisch positiv definit und damit invertierbar ist. Auch ist $A^t A$ gerade die zweite Ableitung, so dass x ein Minimum sein muss.

4.2.2 Ein mehrdimensionales Beispiel

Beispiel: Für einen Automotor misst man (Zahlen sind erfunden):

Drehzahl ω (1000 $\frac{\mathrm{Umdrehungen}}{\mathrm{Minute}}$)	1	1.5	2	2.5	3
Leistung P (kW)	50	100	120	130	125

Wir wollen die Kennlinie durch eine Parabel $P(\omega) = c_2\omega^2 + c_1\omega + c_0$ approximieren. Daher minimieren wir

$$\Phi(x) = \frac{1}{2}\|Ax - b\|^2 \text{ mit } A = \begin{pmatrix} 1 & 1 & 1 \\ 2.25 & 1.5 & 1 \\ 4 & 2 & 1 \\ 6.25 & 2.5 & 1 \\ 9 & 3 & 1 \end{pmatrix}, \; b = \begin{pmatrix} 50 \\ 100 \\ 120 \\ 130 \\ 125 \end{pmatrix}$$

um die Koeffizienten $x = (c_2, c_1, c_0)^t$ zu erhalten.

4.3 Lösung der Normalgleichung

Programm: (Lösung der Normalgleichung) Gegeben seien A und b. Dann liefert folgender Scilab-Code die Lösung der Normalgleichung:

```
x = (A'*A) \ (A'*b)
```

Beispiel: Das Motorbeispiel lässt sich daher wie folgt lösen:

```
A = [1,1,1; 2.25,1.5,1; 4,2,1; 6.25,2.5,1; 9,3,1]
b = [50;100;120;130;125]
x = (A'*A) \ (A'*b)
```

liefert die Lösung $x = (-34.286, 173.143, -87.000)^t$.

4.3.1 Kondition des Problems

Satz: Die Kondition des Minimierungsproblems von $\Phi(x)$ lässt sich im Normalfall (d.h., wenn das zugrunde liegende Modell überhaupt Sinn macht) durch die Größe

$$\kappa_2(A) := \frac{\max_{\|x\|_2=1}\|Ax\|_2}{\min_{\|x\|_2=1}\|Ax\|_2}$$

beschreiben. (Man beachte, dass dieser Ausdruck, den wir aus der Übung kennen, auch für $A \in \mathbb{R}^{m\times n}$ mit $m > n$ Sinn macht, und auch, dass wir mit $\|\cdot\|_2$ die Euklidische Norm im $\mathbb{R}^n$ und $\mathbb{R}^m$ bezeichnen).

Beweis: Siehe [Golub-van-Loan].

Problem: Wenn man x über die Normalgleichung bestimmt, ist die Fortpflanzung des relativen Fehlers durch $\kappa_2(A^tA) = \kappa_2(A)^2$ gegeben (wegen der Multiplikation A^tb muss man insgesamt sogar mit $\kappa_2(A)^3$ rechnen). Für viele Anwendungsprobleme ist aber schon $\kappa_2(A)$ groß und $\kappa_2(A)^2$ oder gar $\kappa_2(A)^3$ inakzeptabel!

4.4 Lösung mittels Orthogonaltransformationen

4.4.1 Orthogonale Matrizen

Definition: Eine Matrix $Q \in \mathbb{R}^{n\times n}$ (bzw. die zugehörige lineare Abbildung) heißt orthogonal, wenn sie das Euklidische Skalarprodukt $(x, y) = \sum_{i=1}^{n} x_i y_i$ erhält, d.h.

$$(Qx, Qy) = (x, y) \quad \forall x, y \in \mathbb{R}^n\,.$$

Interpretation: Q ist längen- und winkelerhaltend.

Bemerkung: Äquivalent zur obigen Bedingung ist die Forderung $QQ^t = Q^tQ = \mathrm{Id}$ oder anders geschrieben $Q^{-1} = Q^t$.

Beispiele: Drehungen, Spiegelungen und Kombinationen davon.

4.4.2 QR-Zerlegung

Definition: Eine QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m\times n}$ hat die Form $A = QR$, wobei $Q \in \mathbb{R}^{m\times m}$ eine orthogonale Matrix und $R \in \mathbb{R}^{m\times n}$ eine obere Dreiecksmatrix ist (d.h. $R_{ij} = 0$ für $i > j$).

Satz: Zu jedem $A \in \mathbb{R}^{m\times n}$ gibt es eine QR-Zerlegung.

Beweis: Man kann die QR-Zerlegung auf verschiedene Weise konstruieren, etwa durch Spiegelungen, Drehungen oder auch die Gram-Schmidt-Orthogonalisierung.

Bemerkung:

- Der Scilab/Matlab-Befehl zur Berechnung einer QR-Zerlegung heißt `qr`.
- In manchen Fällen (z.B. wenn $A \in \mathbb{R}^{n\times n}$ invertierbar ist und wenn $R_{ii} > 0$ gefordert wird), ist die QR-Zerlegung sogar eindeutig (siehe Übung).

4.4.3 Anwendung der QR-Zerlegung

Beobachtung: Weil Q orthogonal ist und $m \geq n$ ist, gilt mit $\tilde{b} = Q^tb$

$$\begin{aligned}
\|Ax - b\|_2^2 &= \|QRx - b\|_2^2 \\
&= \|Rx - Q^tb\|_2^2 \\
&= \|\begin{pmatrix} R_{1:n,1:n}x - \tilde{b}_{1:n} \\ -\tilde{b}_{n+1:m} \end{pmatrix}\|_2^2 \\
&= \|R_{1:n,1:n}x_{1:n} - \tilde{b}_{1:n}\|_2^2 + \|\tilde{b}_{n+1:m}\|_2^2
\end{aligned}$$

Falls $\mathrm{Rang}(A) = n$, so ist auch $\mathrm{Rang}(R) = n$, und das Minimum erhält man offenbar durch Lösung des Dreieckssystems

$$R_{1:n,1:n}x_{1:n} = \tilde{b}_{1:n}\,.$$

Notation: Wir haben die Matlab-Notation zur Bezeichnung von Untermatrizen durch Indexbereiche verwendet.

Bemerkung: Man beachte, dass man zur Lösung von

$$R_{1:n,1:n}x_{1:n} = \tilde{b}_{1:n}$$

nur Teile der QR-Zerlegung braucht, nämlich $Q_{1:m,1:n}$ und $R_{1:n,1:n}$. Es gibt daher den Scilab-Befehl `qr(A,"e")` (Matlab: `qr(A,0)`), der genau diese Matrizen berechnet.

Programm: Die Lösung des Kleinste-Quadrate-Problems mit Matrix $A \in \mathbb{R}^{m\times n}$ und $b \in \mathbb{R}^m$ erhält man daher in Scilab auch wie folgt:

```
[Q,R] = qr(A,"e")
x = R \ (Q'*b)
```

Beispiel: Die Anwendung auf das Motorenbeispiel zeigt, dass man dieselbe Lösung erhält wie vorher.

4.4.4 Anwendung auf Gleichungssysteme

Beobachtung: Im Fall $m = n$ entspricht das Kleinste-Fehlerquadrate-Problem der Lösung des Gleichungssystems $Ax = b$. Die QR-Zerlegung ist daher auch für die Lösung eines Gleichungssystems $Ax = b$ geeignet (man löst einfach das Dreieckssystem $Rx = Q^t b$).

Satz: Im Gegensatz zur LR-Zerlegung ist die Stabilität der QR-Zerlegung garantiert: es gilt $\kappa_2(Q) = 1$ und $\kappa_2(R) = \kappa_2(A)$.

Beweis: Da Q längenerhaltend ist, folgt $\kappa_2(Q) = 1$. Wegen

$$\|Ax\|_2 = \|QRx\|_2 = \|Rx\|_2$$

folgt $\kappa_2(A) = \kappa_2(R)$ (siehe Übungsaufgabe).

Aber: Der Rechenaufwand zur Berechnung einer QR-Zerlegung ist 2-3 mal höher als bei der LR-Zerlegung. Man verwendet die QR-Zerlegung daher meist nur bei schlecht konditionierten Problemen.

5 Nichtlineare Gleichungssysteme

Problem: Gegeben $F : \mathbb{R}^n \to \mathbb{R}^n$ und $y \in \mathbb{R}^n$. Suche $x \in \mathbb{R}^n$ mit $F(x) = y$.

Bemerkung: Ohne wesentliche Einschränkung könnte man sich mittels $F \leftarrow F - y$ auf den Fall $y = 0$ beschränken. In Theorie und Praxis ist allerdings ein allgemeines y recht praktisch.

Anwendungen: In dieser Allgemeinheit fällt fast alles darunter. Oft kommt das Gleichungssystem aus einem Optimierungsproblem, weil ein Extremum einer *Funktion* $\Phi : \mathbb{R}^n \to \mathbb{R}$ das *System* $F(x) := \nabla\Phi(x) = 0$ lösen muss.

5.1 Der eindimensionale Fall

Problem: Sei $f : \mathbb{R} \to \mathbb{R}$ und $y \in \mathbb{R}$. Suche ein $x \in \mathbb{R}$ mit $f(x) = y$.

Beispiele:

- Berechne $\sqrt{2}$ numerisch $\Rightarrow$ Finde $x \in \mathbb{R}$ mit $x^2 = 2$.
- Wann lag der Ölpreis bei 70 Dollar/Barrel? $\Rightarrow$ Finde Zeit mit Ölpreis(Zeit) $= 70$.

5.1.1 Halbierungsverfahren

Satz: (Zwischenwertsatz) Sei $f : \mathbb{R} \to \mathbb{R}$ stetig und $y, a, b \in \mathbb{R}$ mit $a \leq b$ und $f(a) \leq y \leq f(b)$ oder $f(b) \leq y \leq f(a)$. Dann gibt es $x \in [a, b]$ mit $f(x) = y$.

Beweis: Analysis ([HM-Skript], Satz 7.11).

Frage: Kann man diesen Satz zur Lösung von $f(x) = y$ verwenden?

Idee: (Halbierungsverfahren) Nimm $c = \frac{a+b}{2}$ und berechne $f(c)$. Je nach dem Vorzeichen von $y - f(c)$ kann man eines der Teilintervalle $[a, c]$ oder $[b, c]$ so auswählen, dass darin ein x mit $f(x) = c$ liegt. Dann kann man sich aber auf dieses kleinere Intervall beschränken, und mehrfache Intervallhalbierung bestimmt eine Lösung x beliebig genau.

Programm:

```
function solution = interval_solve(f,y,eps,a,b)
while abs(b-a)>eps
  c = (a+b)/2;
  if (f(a)-y)*(f(c)-y)<0
```

```
        b=c;
    else
        a=c;
    end
  end
  solution=a;
endfunction

function y = square (x)
  y=x*x;
endfunction
```

Bemerkung:

- Die Grenzen der berechneten Intervalle $I_k = [a_k, b_k]$ bilden Folgen $(a_k)_{k\in\mathbb{N}}$ und $(b_k)_{k\in\mathbb{N}}$, die beide gegen einen Wert x mit $f(x) = y$ konvergieren.

  (BEWEIS: Das Intervall $I_k = [a_k, b_k]$ hat die Länge $\frac{|b-a|}{2^k}$. Weil die Intervalle immer ineinander enthalten sind (Intervallschachtelung), sieht man sofort, dass die Folgen $(a_k)_{k\in\mathbb{N}}$ und $(b_k)_{k\in\mathbb{N}}$ Cauchy-Folgen sind und gegen ein und dasselbe $x \in \mathbb{R}$ konvergieren. Weil jedes Intervall I_k einen Punkt ξ_k mit $f(\xi_k) = y$ einschließt, muss wegen der Stetigkeit auch $f(x) = y$ sein.)

- Das Verfahren braucht als Startwert geeignete Intervallgrenzen. Dies macht die Anwendung auf mehrdimensionale Probleme problematisch.

Definition: Numerische Verfahren, die anstelle einer Zahl eine approximierende Folge liefern, nennt man Iterationen oder iterative Verfahren.

5.1.2 Das Newton-Verfahren

Voraussetzung: Mindestens $f \in C^1(\mathbb{R}, \mathbb{R})$, am besten $f \in C^2(\mathbb{R}, \mathbb{R})$ (dann hat man oft sehr schnelle Konvergenz). Außerdem brauchen wir einen Startwert $x_0 \in \mathbb{R}$.

Erinnerung: $f \in C^k(\mathbb{R}, \mathbb{R})$ heißt, dass f k-mal stetig differenzierbar ist.

Idee: Mit Hilfe der Werte $f(x_0)$ und $f'(x_0)$ kann man f lokal um x_0 durch eine lineare Funktion approximieren (Definition der Ableitung!), d.h.

$$y \stackrel{!}{=} f(x) \approx f(x_0) + f'(x)(x - x_0) \quad \Leftrightarrow \quad f(x_0) + f'(x)(x - x_0) \approx y\,.$$

Hieraus ersieht man eine Iterationsvorschrift für eine Verbesserung der Lösungsapproximation x_0:

$$x_1 = x_0 + \frac{1}{f'(x_0)}(y - f(x_0))\,.$$

Algorithmus: (Verallgemeinertes Newton-Verfahren) Gegeben seien $f \in C^1(\mathbb{R}, \mathbb{R})$, $f' \in C^0(\mathbb{R}, \mathbb{R})$, $y \in \mathbb{R}$, sowie ein Startwert $x_0 \in \mathbb{R}$. Das Newton-Verfahren konstruiert hieraus die Folge

$$x_{k+1} = x_k + \frac{1}{f'(x_k)}(y - f(x_k)) \quad (k = 1, 2, \ldots).$$

Bezeichnung: Unter dem einfachen Newton-Verfahren wird oft der Fall $y = 0$ verstanden, d.h. man sucht eine Nullstelle von f. In diesem Skript betrachten wir die allgemeinere Form mit beliebiger rechter Seite y, die wir verallgemeinertes Newton-Verfahren nennen.

Programm: (Scilab-Code für das 1D-Newton-Verfahren)

```
function x = newton (f,Df,y,x0,eps)
  x=x0; res=y-f(x);
  while abs(res)>eps
    x=x+res/Df(x);
    disp(x);
    res=y-f(x);
  end
endfunction

function y=square(x)
  y=x*x;
endfunction

function y=twice(x)
  y=2*x;
endfunction
```

Anwendungsbeispiel: Berechnung von $\sqrt{2}$:

```
function y = square(x)
   y=x*x;
endfunction

function y = twice(x);
   y=2*x;
endfunction

newton (square, twice, 2, 1, 1e-12)
```

Beobachtungen:

- Für positive Startwerte konvergiert das Verfahren gegen $\sqrt{2}$, für negative gegen $-\sqrt{2}$. Für den Startwert 0 ist es nicht definiert.
- Die Konvergenz ist sehr schnell, wenn der Startwert nahe bei einer der Lösungen $\pm\sqrt{2}$ ist.

Anwendungsbeispiel: Berechnung der Nullstelle $x = 0$ von $f(x) = \tanh(x)$:

```
function y = f(x)
   y=tanh(x)
endfunction

function y = Df(x)
   y=1-tanh(x)^2
endfunction

newton (f, Df, 0, 1, 1e-12)
```

Beobachtung: Hier konvergiert das Verfahren nur, wenn der Startwert „nahe“ bei 0 ist ($|x| \leq 1$ reicht). Für „große“ Startwerte (z.B. $x = 2$) werden die Iterierten mit wechselndem Vorzeichen sehr schnell größer.

5.2 Der mehrdimensionale Fall

Problem: Zu $F \in C^1(\mathbb{R}^n, \mathbb{R}^n)$ und $y \in \mathbb{R}^n$ wollen wir das Problem $F(x) = y$ lösen.

Erinnerung: Die Ableitung von $F \in C^1(\mathbb{R}^n, \mathbb{R}^n)$ bezeichnen wir mit DF. Es ist eine stetige matrixwertige Funktion, also $DF \in C^0(\mathbb{R}^n, \mathbb{R}^{n\times n})$.

5.2.1 Beispiel

Aufgabe: Schneide den Kreis

$$x_1^2 + x_2^2 = 1$$

mit der Kurve

$$x_2 = e^{x_1}\,.$$

Zusammengefasst führt das zum Gleichungssystem

$$F(x) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{wobei} \quad F : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1^2 + x_2^2 \\ x_2 - e^{x_1} \end{pmatrix}\,.$$

Bemerkung: Eine Grafik zeigt, dass es zwei Lösungen gibt: die eine ist der Punkt $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, die andere liegt in der Nähe von $\begin{pmatrix} -1 \\ 1/e \end{pmatrix}$.

Idee: Nach Definition der Ableitung gilt wieder die lineare Approximation

$$F(x) \approx F(x_0) + DF(x_0)(x - x_0)\,.$$

Daher sollte sich das Newton-Verfahren einfach übertragen lassen.

Algorithmus: (Verallgemeinertes Newton-Verfahren für n-dimensionale Probleme) Gegeben seien $F \in C^1(\mathbb{R}^n, \mathbb{R}^n)$, die Ableitung $DF \in C^0(\mathbb{R}^n, \mathbb{R}^{n\times n})$ und $y \in \mathbb{R}^n$, sowie ein Startwert $x_0 \in \mathbb{R}^n$. Dann erhalten wir die Folge der Newton-Iterierten durch

$$x_{k+1} = x_k + (DF(x_k))^{-1}(y - F(x_k))\,.$$

Bemerkung: Die Ableitung $DF(x_k)$ ist eine Matrix (die sogenannte Jacobi-Matrix) und muss invertiert werden. Das Verfahren bricht zusammen, wenn es auf einen Punkt mit singulärer Jacobi-Matrix läuft.

Programm: (Newton-Verfahren)

```
function x = newton (f,Df,y,x0,eps)
  x=x0;
  res=y-f(x);
  while norm(res)>eps
    x=x+(Df(x)) \ res;
    disp(x);
    res=y-f(x);
  end
endfunction
```

Bemerkung: Es waren nur zwei Änderungen des 1D-Programms nötig: erstens die Verwendung von `norm` anstelle von `abs`, zweitens die Verwendung des Backslash-Operators. Das entstandene Programm funktioniert für beliebige Dimensionen.

Beispiel: Den zweiten Schnittpunkt des Einheitskreises mit dem Graph $x_2 = e^{x_1}$ berechnet der Code

```
function y = f(x)
   y=[x(1)^2+x(2)^2; x(2)-exp(x(1))];
endfunction

function D = Df(x)
   D=[2*x(1),2*x(2); -exp(x(1)),1];
endfunction

newton (f, Df, [1;0], [1;0], 1e-12)
```

Man erhält die Lösung $x \approx \begin{pmatrix} -0.91656258 \\ 0.39989127 \end{pmatrix}$.

5.3 Theorie des Newton-Verfahrens

5.3.1 Kondition des Problems

Satz: (Umkehrfunktion) Sei $F \in C^1(\mathbb{R}^n, \mathbb{R}^n)$ und $y = F(x)$. Wenn $DF(x)$ invertierbar ist, so existieren Umgebungen $U \subset \mathbb{R}^n$ von x und $V \subset \mathbb{R}^n$ von y sowie eine Abbildung $F^{-1} : V \to U$, so dass $F^{-1} \circ F \mid_U = \mathrm{Id}_U$ und $F \circ F^{-1} \mid_V = \mathrm{Id}_V$. Außerdem gilt $D(F^{-1})(y) = (DF(x))^{-1}$.

Beweis: Analysis ([HM-Skript, Satz 9.9 und Satz 22.13]).

Frage: Wie groß ist die Änderung Δx der Lösung x der nichtlinearen Gleichung $F(x) = y$ bei Störungen der rechten Seite y um Δy?

Antwort: Wenn $DF(x)$ regulär ist, so existiert lokal F^{-1} und es gilt nach Kapitel 1

$$\|\Delta x\| \stackrel{\cdot}{\leq} \|D(F^{-1})(y)\| \|\Delta y\| = \|(DF(x))^{-1}\| \|\Delta y\| \,.$$

Wieder bezeichnet $\|\cdot\|$ eine beliebige Vektornorm sowie die zugehörige Operatornorm.

Bezeichnung: Wir nennen die Größe

$$\kappa_{\mathrm{abs}}(F(x) = y) = \|D(F^{-1})(y)\| = \|(DF(x))^{-1}\| \,.$$

die Kondition der Lösung x des Problems $F(x) = y$. Sie hängt von der gewählten Norm ab.

Bemerkungen:

- Für nichtlineare Probleme kann es zu einem y mehrere Lösungen geben (z.B. hat $x^2 = 2$ die Lösungsmenge $\{+\sqrt{2}, -\sqrt{2}\}$). Die Kondition einer bestimmten Lösung x quantifiziert das lokale Verhalten einer Umkehrfunktion, die auf eine Umgebung von x abbildet!
- Man beachte, dass man für die Fehlerfortpflanzung bei nichtlinearen Problemen normalerweise absolute Fehler betrachtet, und dementsprechend auch die absolute Kondition verwendet. Der Grund ist einfach: Im allgemeinen gilt nicht $F(0) = 0$, so dass der Begriff der relativen Kondition wenig Sinn macht.

Beispiel: Wir betrachten wieder das Problem des Schnitts von Einheitskreis und Graph der Exponentialfunktion. Die Ableitung hatte hier die Form

$$DF(x) = \begin{pmatrix} 2x_1 & 2x_2 \\ -e^{x_1} & 1 \end{pmatrix}$$

Die Konditionen der Schnittpunkte $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ und $\begin{pmatrix} -0.91656258\ldots \\ 0.39989127\ldots \end{pmatrix}$ berechnen sich dann wie

- $\|DF(\begin{pmatrix} 0 \\ 1 \end{pmatrix})^{-1}\|_\infty = \|\begin{pmatrix} 0 & 2 \\ -1 & 1 \end{pmatrix}^{-1}\|_\infty = \|\begin{pmatrix} \frac{1}{2} & -1 \\ \frac{1}{2} & 0 \end{pmatrix}\|_\infty = \frac{3}{2}$
- $\|DF(\begin{pmatrix} -0.91656258\ldots \\ 0.39989127\ldots \end{pmatrix})^{-1}\|_\infty = 1.4755949\ldots$

Für beide Schnittpunkte ist der Wert moderat, die Schnittpunkte sind also gut konditioniert.

Beispiel: Lineare Probleme sind ein Spezialfall von nichtlinearen. Wir wollen den Schnittpunkt der Geraden $y = kx + m$ mit der x-Achse bestimmen. Dazu suchen wir die Lösung von

$$F(\xi) = \begin{pmatrix} 0 \\ m \end{pmatrix}, \quad F : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} y \\ y - kx \end{pmatrix}.$$

Die Lösung ist natürlich $\xi = \begin{pmatrix} -\frac{m}{k} \\ 0 \end{pmatrix}$, die Ableitung $DF = \begin{pmatrix} 0 & 1 \\ -k & 1 \end{pmatrix}$. Für kleines k ist diese Matrix fast singulär und die Norm der Inversen wird beliebig groß. Der Schnittpunkt ist also schlecht konditioniert. Geometrisch bedeutet dies, dass die Lage des Schnittpunkts beim schleifenden Schnitt sehr schlecht bestimmt ist, was ja der Erfahrung entspricht.

Beispiel: Wir betrachten das Problem

$$F(x) = y\,, \qquad \text{mit } F(x) = x^2 \text{ und } y = 0\,.$$

Es hat die eindeutig bestimmte Lösung $x = 0$, allerdings ist $F'(x) = 0$, die Kondition von x also nicht definiert (oder unendlich groß). Solche Probleme nennt man schlecht gestellt, und das Verhalten der Lösungsmenge bei Variation von y kann beliebig bösartig sein. Im obigen Fall beobachten wir folgendes:

- Für $\Delta y = \varepsilon > 0$ gibt es *zwei Lösungen* $x = \pm\sqrt{\varepsilon}$. Es gilt $|\Delta x| = \frac{1}{\sqrt{\varepsilon}}|\Delta y|$, was für kleines ε eine *beliebig große Fehlerverstärkung* bedeutet.
- Für $\Delta y = \varepsilon < 0$ gibt es *gar keine reelle Lösung*.

Bemerkung: Wenn man in der Praxis auf ein solch schlecht gestelltes Problem stößt, sollte man seine Herkunft zurückverfolgen. Oft wird man feststellen, dass es durch einen Fehler zustande kam, und der Anwender eigentlich die Lösung eines ganz anderen Problems wünscht.

5.3.2 Fixpunktiterationen

Beobachtung: Die Newton-Iteration

$$x_{k+1} = x_k + (DF(x_k))^{-1}(y - F(x_k))$$

kann man auch als eine spezielle Fixpunkt-Iteration der Form

$$x_{k+1} = \Phi(x_k)\,, \quad \Phi(x_k) := x_k + (DF(x_k))^{-1}(y - F(x_k))$$

auffassen. Wenn die Folge $(x_k)_{k\in\mathbb{N}}$ konvergiert und Φ stetig ist, so ist der Grenzwert ein Fixpunkt von Φ, d.h. es gilt $\Phi(x) = x$.

5.3.3 Kontraktionen und Fixpunktsatz

Definition: Sei $U \subset \mathbb{R}^n$ und $\Phi : U \to U$ erfülle

$$\|\Phi(x) - \Phi(y)\| \leq \theta \|x - y\|$$

für ein $\theta < 1$. Dann nennen wir Φ eine Kontraktion auf U mit Kontraktionsrate θ bezüglich der Vektornorm $\|\cdot\|$.

Bemerkungen:

- Offenbar ist jede Kontraktion (Lipschitz-)stetig.
- Die Kontraktionseigenschaft hängt für $n \geq 2$ von der gewählten Norm ab.
- Nur im Fall $n = 1$ ist jede Norm ein Vielfaches der üblichen Betragsfunktion, so dass man sie dort nicht unbedingt spezifizieren muss.

Beispiele:

- Die Abbildung $\Phi(x) = \frac{x}{2}$ ist eine Kontraktion mit Rate $\frac{1}{2}$.
- Die Abbildung $\Phi(x) = x^2$ ist im Intervall $[0, 1]$ keine Kontraktion wegen

$$|\Phi(1) - \Phi(0)| = 1 = 1|1 - 0|$$

- Weiter unten werden wir zeigen, dass dasselbe $\Phi(x) = x^2$ eingeschränkt auf das Intervall $[0, \alpha]$ für $\alpha < \frac{1}{2}$ eine Kontraktion mit Rate 2α ist.

Satz: (Banachscher Fixpunktsatz im $\mathbb{R}^n$) Sei $U \subset \mathbb{R}^n$ abgeschlossen und $\Phi : U \to U$ eine Kontraktion bezüglich einer beliebigen Vektornorm $\|\cdot\|$. Dann gibt es einen eindeutig bestimmten Fixpunkt $x_* \in U$ mit $\Phi(x_*) = x_*$.
Genauer: Jede *beliebige* Folge der Form

$$x_0 \in U\,, \quad x_{k+1} = \Phi(x_k)$$

konvergiert gegen diesen eindeutig bestimmten Fixpunkt $x_* \in U$.

Beweis: Existenz: Die Folge der Iterierten ist wegen

$$\|x_{k+1} - x_k\| \leq \theta \|x_k - x_{k-1}\| \leq \ldots \leq \theta^k \|x_1 - x_0\|$$

und für $l \geq k$

$$\|x_l - x_k\| \leq \|x_l - x_{l-1}\| + \ldots + \|x_{k+1} - x_k\| \leq \|x_1 - x_0\| \theta^k \frac{1}{1 - \theta}$$

eine Cauchy-Folge. Daher konvergiert sie gegen ein $x_* \in U$ (Vollständigkeit von $\mathbb{R}^n$, Abgeschlossenheit von U). Weil Φ als Kontraktion stetig ist, gilt $x_* = \Phi(x_*)$.

Eindeutigkeit: Seien x_1^* und x_2^* Fixpunkte. Dann gilt

$$\|x_1^* - x_2^*\| \leq \|\Phi(x_1^*) - \Phi(x_2^*)\| \leq \theta \|x_1^* - x_2^*\| .$$

Weil $\theta < 1$, geht das nur wenn $\|x_1^* - x_2^*\| = 0$, also $x_1^* = x_2^*$.

Definition: Eine Folge (x_k) heißt linear konvergent gegen ein x_* mit der Konvergenzrate $\theta < 1$, wenn

$$\|x_{k+1} - x_*\| \leq \theta \|x_k - x_*\| .$$

Interpretation: Der Fehler verringert sich in jedem Schritt mindestens um den Faktor θ.

Bemerkung: Wenn $\Phi : U \to U$ eine Kontraktion mit Rate θ ist, so konvergiert eine durch $x_{k+1} = \Phi(x_k)$ definierte Iteration linear mit Rate θ gegen den Fixpunt x_*.

Beweis: Es gilt

$$\|x_{k+1} - x_*\| = \|\Phi(x_k) - \Phi(x_*)\| \leq \theta \|x_k - x_*\| .$$

Definition: Eine Menge $U \subset \mathbb{R}^n$ heißt konvex, wenn für $x, y \in U$ das ganze Liniensegment $\{sx + (1-s)y : s \in [0,1]\}$ ebenfalls in U liegt.

Kriterium: Sei $F \in C^1(\mathbb{R}^n, \mathbb{R}^n)$ und für eine konvexe Menge $U \subset \mathbb{R}^n$ gelte $F(U) \subset U$ sowie

$$\|DF(x)\| \leq \theta < 1 , \quad x \in U$$

Dann ist $F|_U : U \to U$ eine Kontraktion mit Kontraktionsrate θ.

Beweis: Es gilt

$$F(y) - F(x) = \int_0^1 DF(x + t(y-x))(y-x)\, dt$$

und somit

$$\|F(y) - F(x)\| \leq \|y - x\| \sup_{t \in [0,1]} \|DF(x + t(y-x))\| \leq \theta \|y - x\| .$$

Frage: Wie groß darf $\alpha > 0$ gewählt werden, damit $F(x) = x^2$ auf dem Intervall $[-\alpha, \alpha]$ eine Kontraktion ist?

Antwort: Wir berechnen $F'(x) = 2x$ und sehen, dass $|F'(x)| \leq \theta < 1$ nur gilt, wenn $\alpha < \frac{1}{2}$ gewählt wird.

Bemerkung: Wir sehen hier auch, dass die Forderung nach einer Kontraktion nur *hinreichend*, aber nicht *notwendig* für die Existenz eines eindeutig bestimmten Fixpunkts ist. Denn offensichtlich hat $F|_{[-\alpha,\alpha]}$ ja für alle $\alpha < 1$ den eindeutig bestimmten Fixpunkt 0.

5.3.4 Anwendung auf das Newton-Verfahren

Erinnerung: Das Newton-Verfahren entsprach der Fixpunktiteration

$$x_{k+1} = \Phi(x_k) \quad \text{mit} \quad \Phi(x) = x + (DF(x))^{-1}(y - F(x))\,.$$

Satz: Sei $F \in C^2(\mathbb{R}^n, \mathbb{R}^n)$ und x_* sei eine Lösung mit $F(x_*) = y$, für die $DF(x_*)$ invertierbar ist. Dann gibt es zu jedem $0 < \theta < 1$ ein $\delta > 0$, so dass das Newton-Verfahren in der Kugel $B_\delta(x_*)$ mit Rate θ konvergiert.

Beweisskizze: Der Einfachheit halber zeigen wir es nur für den 1D-Fall und ohne die Konstanten zu quantifizieren (und bezeichnen daher wieder F mit f).
Weil f zweimal stetig differenzierbar ist, und $f'(x_*) \neq 0$, ist Φ in einer Umgebung von x_* stetig differenzierbar mit Ableitung

$$\begin{aligned} \Phi'(x) &= 1 - f''(x)f'(x)^{-2}(y - f(x)) - f'(x)^{-1}f'(x) \\ &= -f''(x)f'(x)^{-2}(y - f(x))\,. \end{aligned}$$

Wegen $y = f(x_*)$ gilt $\Phi'(x_*) = 0$. Und weil Φ' stetig ist, gibt es daher zu einer beliebigen Kontraktionszahl θ ein $\delta > 0$, so dass $|\Phi'(x)| \leq \theta$ für $x \in B_\delta(x_*)$. Daher ist Φ eine Kontraktion mit Rate θ auf $U = B_\delta(x_*)$. Mit unserer Version des Fixpunktsatzes folgt dann auch die Konvergenz der Fixpunktiteration.

Definition: Eine Folge (x_k) im $\mathbb{R}^n$ heißt quadratisch konvergent gegen ein $x_* \in \mathbb{R}^n$, wenn es ein $C > 0$ gibt mit

$$\|x_{k+1} - x_*\| \leq C\|x_k - x_*\|^2\,.$$

Satz: Unter den obigen Voraussetzungen ist das Newton-Verfahren quadratisch konvergent.

Beweisskizze: In den oben konstruierten Umgebungen von x_* gelten auch folgende Abschätzungen:

$$|y - f(x)| \leq C_1|x - x_*|\,, \quad |f'(x)^{-1}| \leq C_2\,, \quad |f''(x)| \leq C_3\,.$$

Die Kontraktionsrate in der Kugel um x_* mit Radius $\|x - x_*\|$ lässt sich daher abschätzen durch $C\|x - x_*\|$ mit $C = C_1C_2C_3$.

Bemerkung: Diese quadratische Konvergenz in der Nähe der Lösung x_* beobachtet man tatsächlich in der Praxis, vorausgesetzt:

1. F und DF sind ausreichend glatt ($F \in C^2$, $DF \in C^1$),
2. die Lösung x_* ist gut konditioniert ($DF(x_*)$ ist regulär)
3. F und DF werden ausreichend genau berechnet und
4. es wurden keine Fehler in der Implementation gemacht.

5.4 Das globale Newton-Verfahren

Problem: In der Praxis ist es oft schwierig, Startwerte zu finden, die so nahe bei einer Lösung sind, dass das unmodifizierte Newton-Verfahren konvergiert. Wir haben am Beispiel des $\tanh$ gesehen, dass dies sogar schon in sehr einfachen Situationen auftreten konnte.

Beobachtung: Wenn F differenzierbar ist und DF regulär ist, so ist die Newton-Richtung

$$v = DF^{-1}(x_k)(y - F(x_k))$$

auf jeden Fall eine Richtung, in der die Funktion $s \mapsto \|y - F(x_k + s)\|$ kleiner wird. Dies sieht man, weil ja wegen der Differenzierbarkeit in x_k die Approximation

$$F(x) \doteq F(x_k) + DF(x_k)(x - x_k)$$

gilt. Man kann aber nicht garantieren, dass man den Schritt *in voller Länge* durchführen kann, ohne den Gültigkeitsbereich dieser Approximation zu verlassen!

Idee: Dämpfe den Newton-Schritt mit einem Faktor $\omega_k \in]0, 1]$:

$$v_k = DF^{-1}(x_k)(y - F(x_k)), \quad x_{k+1} = x_k + \omega_k v_k$$

wobei ω_k geeignet gewählt wird.

Bezeichnung: Das entstehende Verfahren heißt Newton-Verfahren mit Dämpfungsstrategie oder globales Newton-Verfahren.

Bemerkung: Es gibt verschiedene Strategien, den Dämpfungsfaktor zu wählen. Eine oft angewandte Strategie funktioniert im wesentlichen wie folgt:

1. Ein gegebener Dämpfungsfaktor ω_k wird so lange halbiert, bis er „zulässig“ ist. Dabei bedeutet „zulässig“, dass die Verringerung der Fehlernorm im wesentlichen dem erwarteten linearen Abfall entspricht:

   $$\|F(x_k + \omega_k v_k)\| \leq (1 - \theta\omega_k)\|F(x_k)\|.$$

   Hierbei ist $\theta \in (0, 1)$ beliebig, eine in der Praxis bewährte Wahl ist $\theta = \frac{1}{2}$.

2. Am Anfang wählt man z.B. $\omega_0 = 1$ (oder aber etwas kleineres, wenn das Problem schwierig ist). Später führt man obige Prozedur ausgehend vom Startwert $\omega_{k+1} = \min(2\omega_k, 1)$ aus durch. Durch diese anfängliche Verdopplung wird die Dämpfung allmählich wieder ausgeschaltet, sobald das Verfahren mit größeren Schritten gut funktioniert. Hierdurch man erhält man insbesondere nahe der Lösung die quadratische Konvergenz des ungedämpften Verfahrens zurück.

Programm: (Globales Newton-Verfahren)

```
function x = global_newton (f,Df,y,x0,eps)
  x = x0;
  res = y-f(x);
  omega = 1;
  while norm(res)>eps do
    v = (Df(x)) \ res;
    omega = min(2*omega,1);
    while norm(y-f(x+omega*v)) > (1-theta*omega)*norm(res) do
      omega=omega/2;
    end
    x = x+omega*v;
    disp(omega);
    disp(x);
    res=y-f(x);
  end
endfunction

function y = f(x)
  y = tanh(x);
endfunction

function y = Df(x)
  y = 1-tanh(x)^2;
endfunction

theta = 0.5;

// global_newton(f,Df,0,10,1e-10)
```

Beobachtung: Diese Newton-Variante konvergiert bei der Berechnung der Nullstelle des $\tanh$ für alle Startwerte $x_0 \in \mathbb{R}$ gegen die Lösung $x = 0$.

5.5 Iterative Lösung linearer Gleichungssysteme

5.5.1 Einführung

Beobachtung: Wenn man das Newton-Verfahren auf ein lineares Gleichungssystem

$$F(x) := Ax = y\,, \quad A \in \mathbb{R}^{n\times n}\,, \quad x, y \in \mathbb{R}^n$$

mit invertierbarem A anwendet, so ergibt sich der Newton-Schritt

$$x_{n+1} = x_n + A^{-1}(y - Ax_n) = A^{-1}y\,.$$

Das Verfahren liefert also bereits *nach einem Schritt* die exakte Lösung.

Beobachtung: Wenn man allerdings das Gleichungssystem im Newton-Schritt nur *approximativ* löst, d.h. anstelle von A^{-1} nur eine Approximation B (den sogenannten Vorkonditionierer) anwendet, so liefert die Vorschrift

$$x_{n+1} = x_n + B(y - Ax_n)$$

ein iteratives Verfahren zur Lösung des linearen Gleichungssystems $Ax = y$.

Bemerkung: Eine solche iterative Lösung kann günstig sein, weil insbesondere für hochdimensionale Systeme (d.h. großes n) die Berechnung von A^{-1} oder die Lösung des Gleichungssystems in $O(n^3)$ Operationen zu aufwendig ist.

Beispiel: Wir betrachten die Matrix $A = I + S$ mit $\|S\| \leq \frac{1}{2}$ und setzen einfach $A^{-1} \approx \mathrm{Id} =: B$. Damit erhalten wir die Iteration

$$x_{n+1} = \Phi(x_n) := x_n + B(y - Ax_n) = y - Sx_n\,.$$

Die Abbildung Φ ist aber offenbar wegen

$$\|\Phi(\xi_1) - \Phi(\xi_2)\| = \|S(\xi_1 - \xi_2)\| \leq \|S\|\|\xi_1 - \xi_2\|$$

eine Kontraktion mit Kontraktionsfaktor $\frac{1}{2}$.

Bezeichnung: $\|\cdot\|$ bezeichnet hier wieder sowohl eine (beliebige) Vektornorm als auch die zugehörige Matrixnorm.

Folgerung: Sei A wie oben und $\varepsilon = 2^{-k}$ eine vorgegebene Genauigkeit. Dann approximiert die obige Iteration mit Startwert $x_0 = 0$ die Lösung x von $Ax = y$ in höchstens k Schritten bis auf einen Fehler der Größe $\|x_k - x\| \leq \varepsilon\|x\|$.

Bemerkungen: Der Aufwand für jeden einzelnen Iterationsschritt $x_{n+1} = y - Sx_n$ hängt stark von $S \in \mathbb{R}^{n\times n}$ ab:

- Für allgemeines S ist es im wesentlichen der Aufwand für eine Matrix-Vektor-Multiplikation, also n^2 AO.
- Für dünnbesetzte Matrizen S (d.h. nur $N \ll n^2$ Matrixeinträge von S sind von Null verschieden) ist der Aufwand N AO.
- Manchmal gibt es eine besondere Technik, um die Abbildung $x \mapsto y - Sx$ durchzuführen. Ein Beispiel wäre $S = vw^t$ mit Vektoren $v, w \in \mathbb{R}^n$. Hier braucht man etwa $2n$ AO, um $y - Sx = y - v(w^t x)$ zu berechnen.

5.5.2 Konvergenztheorie

Satz: Wir betrachten die Iteration

$$x_0 \in \mathbb{R}^n\,, \quad x_{n+1} = \Phi(x_n) := x_n + B(y - Ax_n) = By + (\mathrm{Id} - BA)x_n$$

Angenommen, es gibt eine Norm $\|\cdot\|$ (Vektor- und zugehörige Matrixnorm) mit

$$\|\mathrm{Id} - BA\| = \rho < 1\,,$$

so konvergiert die Iteration mit der Konvergenzrate ρ bezüglich dieser Norm.

Beweis: Offenbar ist Φ dann eine Kontraktion mit der Rate ρ bezüglich $\|\cdot\|$.

Interpretation: Gute Konvergenz erhält man offenbar für $\rho = \|\mathrm{Id} - BA\| \ll 1$. Dies bedeutet, dass B nahe bei der Inversen von A liegt!

5.5.3 Die Beziehung zur Kondition

Satz: Es gelte

$$\|I - BA\| \leq \rho < 1\,.$$

Dann gilt

$$\kappa(BA) \leq \frac{1+\rho}{1-\rho}$$

wobei $\kappa(A) = \|A\|\|A^{-1}\|$ die Kondition der Matrix A bezeichnet. Umgekehrt gilt

$$\rho \geq \frac{\kappa(BA)-1}{\kappa(BA)+1}\,.$$

Interpretation: Gute Konvergenz der linearen Iteration ist nur dann möglich, wenn die Kondition der Matrix BA moderat ist. Dies motiviert auch die Bezeichnung Vorkonditionierer für B: eine gute Wahl von B führt nämlich zu $\kappa(BA) \ll \kappa(A)$.

Beweis: Wir setzen $S = BA$ und gehen aus von $\|I - S\| = \rho < 1$. Dies bedeutet

$$\|(I-S)x\| = \|x - Sx\| \leq \rho\|x\|\,, \quad x \in \mathbb{R}^n\,.$$

Die Dreiecksungleichung liefert nun einerseits

$$\|x\| - \|Sx\| \leq \rho\|x\| \quad \Leftrightarrow \quad \|Sx\| \geq (1-\rho)\|x\|\,,$$

andererseits

$$\|Sx\| - \|x\| \leq \rho\|x\| \quad \Leftrightarrow \quad \|Sx\| \leq (1+\rho)\|x\|\,,$$

Nun folgt

$$\kappa := \kappa(S) = \frac{\max_{\|x\|=1}\|Sx\|}{\min_{\|x\|=1}\|Sx\|} \leq \frac{1+\rho}{1-\rho}$$

und Auflösen dieser Ungleichung nach ρ liefert auch $\rho \geq \dfrac{\kappa-1}{\kappa+1}$.

Bemerkung: Wenn $\kappa \gg 1$, so gilt $\dfrac{\kappa-1}{\kappa+1} = \dfrac{1-\frac{1}{\kappa}}{1+\frac{1}{\kappa}} \approx 1 - \dfrac{2}{\kappa}$.

Bemerkung: Die Kunst bei der iterativen Lösung linearer Gleichungssysteme liegt vor allem in der guten Wahl des Vorkonditionierers B. Für einfache Optionen wie

1. $B = \frac{1}{\|A\|_2}$ für symmetrisch positiv definites A oder
2. $B = \frac{1}{\|A^t A\|_2} A^t$ für beliebiges A

konvergiert die Iteration zwar (siehe Übung), aber meist sehr langsam.

5.5.4 Anwendung

Beispiel: Sei $A \in \mathbb{R}^{n \times n}$ gegeben als

$$A_n = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}.$$

Es gilt $\|A_n\|_2 \approx 4$ (siehe Übung).

Ziel: Obwohl man ein LGS mit der obigen Systemmatrix sehr effizient direkt lösen kann (wie?), wollen wir hier die iterative Lösung betrachten. Wir wählen daher $B = \frac{1}{4}\mathrm{Id}$ und untersuchen, wie schnell die folgende Iteration konvergiert:

$$x_{k+1} = x_k + \frac{1}{4}(y - Ax_k).$$

Programm: (Iterative Lösung von $A_n x = (1, \ldots, 1)^t$)

```
function y = apply_A (x)
  n = size(x,1);
  y = 2*x - [x(2:n);0] - [0;x(1:n-1)];
endfunction

function [x,count] = iter_solve (y)
  x = zeros(size(y,1),1);
  res = y-apply_A(x);
  count = 0;
  while (norm(res)>1e-8) do
    x = x+0.25*res;
    res = y-apply_A(x);
    count = count+1;
  end
endfunction
```

Bemerkung: Man beachte, dass die Matrix A nicht einmal abgespeichert werden muss! Zur Durchführung des Verfahrens reicht es vollkommen aus, dass man die Abbildung $x \mapsto Ax$ berechnen kann.

Beobachtung: Die Zahl der notwendigen Iterationen vervierfacht sich bei Verdopplung der Größe des Systems. Der Grund liegt in der Vervierfachung der Kondition der Matrix A_n bei Verdopplung von n, was zu einer entsprechenden Verschlechterung der Konvergenzrate führt.

5.5.5 Optimale „Dämpfung“

Bemerkungen:

- Auch für die iterative Lösung linearer Gleichungssysteme sind Dämpfungsstrategien sinnvoll und wichtig, insbesondere weil sie größere Freiheit in der Wahl des Vorkonditionierers B erlauben.
- Im Gegensatz zur Dämpfung bei nichtlinearen Problemen ist es hier sogar möglich:
  1. einen *optimalen* Dämpfungsfaktor zu *berechnen*,
  2. die Suchrichtung $v = B(y - Ax)$ so zu verändern, dass der Schritt in einem Unterraum, der auch alle alten Suchrichtungen beinhaltet, optimal ist.
- Beispiele solcher Verfahren sind das CG-Verfahren (*conjugate gradient*) für symmetrisch positiv definite Systeme oder das GMRES-Verfahren (*generalized minimum residual*) für eine beliebige Systemmatrix.
- In Scilab/Matlab/Octave gibt es dazu die Befehle `pcg` (preconditioned conjugate gradient) und `gmres`.

Bemerkung: Ähnliche Matrizen wie die Matrix A_n des vorigen Abschnitts entstehen etwa bei der Diskretisierung sogenannter partieller Differentialgleichungen. Mit diesen kann man etwa Strömungen oder die Deformation von Objekten unter Krafteinwirkung beschreiben. Die hohe Zahl der Unbekannten rührt von der Diskretisierung eines mehrdimensionalen Kontinuums mit Hilfe eines feinen Gitters her.

Die schnelle iterative Lösung linearer Systeme mit Hilfe von CG- bzw. GMRES-Verfahren in Kombination mit einem guten Vorkonditionierer ist für eine erfolgreiche numerische Simulation solcher Probleme essentiell.

6 Interpolation

6.1 Einführung

Problem: Eine Funktion $f : [a, b] \to \mathbb{R}$ ist nur an n Punkten $t_1, \ldots, t_n \in [a, b]$ bekannt (z.B. durch Messungen), d.h.

$$f(t_i) = f_i\,, \quad i = 1, \ldots, n\,.$$

Nichtsdestoweniger wissen wir, dass f eigentlich eine stetige (glatte) Größe beschreibt und wollen daher

- die Funktion an einem Wert $t \in [a, b] \setminus \{t_1, \ldots, t_n\}$ approximieren,
- die Funktion graphisch darstellen oder aber
- Ableitungen der Funktion berechnen.

Idee: Wir wählen einen geeigneten n-dimensionalen Vektorraum V bestehend aus Funktionen φ, die so beschaffen sind, dass die n Bedingungen $\varphi(t_i) = f_i$ genau ein Element aus V auswählen.

Beispiele: In dieser Vorlesung betrachten wir nur folgende Situationen:

- f ist glatt und der Raum V besteht aus Polynomen
- f ist glatt und der Raum V besteht aus stückweisen Polynomen (Splines)

Bemerkung: Man beachte, dass die Werte von f an der Stelle t_i hier als genau bekannt angenommen und entsprechend exakt interpoliert werden. Ebenfalls häufig in der Praxis anzutreffen ist das Problem, dass die Messungen der $f_i = f(t_i)$ mit Fehlern behaftet sind. In diesem Fall ist es meist nicht angebracht, die Gleichungen $\varphi(t_i) = f_i$ exakt zu erfüllen. Stattdessen wird zu n Messungen ein k-dimensionaler Funktionenraum V mit $\dim(V) = k \ll n$ gewählt, und dann durch Lösung eines linearen Ausgleichsproblems ein Element aus V ausgewählt (siehe etwa das Automotoren-Beispiel aus Kapitel 3).

6.2 Lagrange-Interpolation

6.2.1 Konstruktion

Voraussetzung: Gegeben sei ein Intervall $I = [a, b]$ und Stützstellen $\{t_0, \ldots, t_n\} \in [a, b]$ mit $t_i \neq t_j$ für $i \neq j$ (d.h. die t_i sind paarweise verschieden).

Aufgabe: (Lagrangesche Interpolationsaufgabe) Zu einer Funktion $f \in C^0([a, b])$ suchen wir nun ein Polynom $p \in \mathcal{P}^n$ mit

$$p(t_i) = f(t_i)\,, \quad i = 0, \ldots, n\,.$$

$\mathcal{P}^n$ bezeichnet hier den Vektorraum aller Polynome mit Grad kleiner oder gleich n.

Beachte: Ab jetzt ist n der Polynomgrad und $n + 1$ die Zahl der Stützstellen.

Definition: Zu den Stützstellen t_i für $i = 0, \ldots, n$ definieren wir die Lagrange-Polynome

$$L_{i,n}(t) = \frac{\prod_{j \neq i}(t - t_j)}{\prod_{j \neq i}(t_i - t_j)}$$

Satz: Es gilt $L_{i,n}(t_j) = \delta_{ij}\,, \quad i, j = 0, \ldots, n$. Hieraus folgt insbesondere, dass die $n + 1$ Lagrange-Polynome eine Basis von $\mathcal{P}^n$ bilden.

Beweis: Wegen $L_{i,n}(t_j) = \delta_{ij}$ sieht man sofort, dass man keines der Lagrange-Polynome aus den anderen durch Linearkombination erhalten kann. Da $\dim(\mathcal{P}^n) = n + 1$ folgt die Behauptung.

Satz: Die Lagrangesche Interpolationsaufgabe besitzt eine eindeutige Lösung, d.h. zu jedem $f \in C^0([a, b])$ gibt es genau ein Polynom $p \in \mathcal{P}^n$, welches die Interpolationsbedingung erfüllt.

Beweis: Existenz: Wir setzen einfach

$$p(x) = \sum_{i=0}^{n} f(t_i) L_{i,n}(t)\,.$$

Eindeutigkeit: Weil jedes Polynom $\tilde{p}$, welches $\tilde{p}(t_i) = f(t_i)$ erfüllt, ebenfalls diese Darstellung in der Lagrange-Basis hat, muss $\tilde{p} = p$ sein.

Beobachtung: Die Interpolationsabbildung

$$I : C^0([a, b]) \to \mathcal{P}^n\,, \quad f \mapsto p$$

die jeder Funktion f das interpolierendes Polynom zu den paarweise verschiedenen Stützstellen $t_0, \ldots, t_n$ zuordnet, ist eine lineare Abbildung. (Es ist eine Projektion von $C^0([a, b])$ auf $\mathcal{P}^n \subset C^0([a, b])$.)

6.2.2 Gute Basiswahl

Frage: Bezüglich welcher Basis von $\mathcal{P}^n$ sollen wir das Interpolationspolynom darstellen? (Diese Frage ist unabhängig davon, ob $\mathcal{P}^n$ an sich geeignet ist.)

Beobachtung: Die Koeffizienten des Interpolationspolynoms in der Lagrange-Basis sind trivialerweise gut konditioniert, weil die lineare Abbildung

$$\tilde{I} : C^0([a,b]) \to \mathbb{R}^{n+1}, \quad f \mapsto \vec{f} := (f(t_i))_{i=0,\ldots,n} \in \mathbb{R}^{n+1}$$

offenbar der Abschätzung

$$\|\vec{f}\|_\infty \leq \|f\|_{C^0([a,b])}$$

genügt, wobei

$$\|f\|_{C^0([a,b])} := \sup_{x\in[a,b]} |f(x)| .$$

Aber: Die Kondition der Koeffizienten eines Interpolationspolynoms in der üblichen Monombasis $\{1, x, x^2, \ldots\}$ ist oft schlecht konditioniert. Diese Koeffizienten $a_0, \ldots, a_n$ ergeben sich nämlich als Lösung des Vandermonde-Systems

$$\begin{array}{rcl} a_0 + a_1 t_0 + \ldots + a_n t_0^n &=& f_0 \\ a_0 + a_1 t_1 + \ldots + a_n t_1^n &=& f_1 \\ \vdots & \vdots & \vdots \\ a_0 + a_1 t_n + \ldots + a_n t_n^n &=& f_n \end{array}$$

welches meist *sehr* schlecht konditioniert ist.

Beispiel: Wir berechnen die Kondition des Vandermonde-Systems für *äquidistante Stützstellen* mit folgendem Scilab-Code:

```
function A = vand(n)
  x=[0:1/n:1]';
  A = zeros(n+1,n+1);
  for i=0:n
    A(1:n+1,i+1) = x .^ i;
  endfor
endfunction

for i=1:10
  disp(i),disp(cond(vand(i)))
endfor
```

Beobachtung: Wir erhalten folgende Werte:

n	1	2	3	4	5	10
$\mathrm{cond}_2(V_n)$	2.6180	15.100	98.868	686.43	4924.4	$1.1558 \cdot 10^8$

Folgerung: Man stellt Interpolationspolynome daher meist nicht bezüglich der Monombasis dar, sondern arbeitet mit der Lagrange-Basis oder der sogenannten Newton-Basis.

Definition: Die Newton-Basis zu den Stützstellen $t_0, \ldots, t_n$ besteht aus den Polynomen

$$\omega_i(t) := \prod_{j=0}^{i-1}(t - t_j) \in \mathcal{P}^i\,, \quad i = 0, \ldots, n\,.$$

Bemerkungen:

- Die Newton-Basis wird bei Hinzufügen neuer Interpolationspunkte erweitert, ohne dass sich die bisherigen Basiselemente verändern.
- Die Bestimmung der Koeffizienten in der Newton-Basis verwendet die sogenannten dividierten Differenzen (siehe [Rannacher, Satz 3.1.2]).
- Die dividierten Differenzen können auch zur *Auswertung* des Interpolationspolynoms an einer bestimmten Stelle x verwendet werden (sog. Schema von Neville).
- Man beachte jedoch, dass die Auswertung an einzelnen Punkten sowohl in der Newton-Basis wie auch in der Lagrange-Basis eine $O(n^2)$-Operation ist, im Gegensatz zum $O(n)$-Aufwand bei Auswertung einer Darstellung in der Monombasis.

6.2.3 Interpolationsfehler

Satz: Sei $f \in C^{n+1}([a,b])$ und sei $p \in \mathcal{P}^n$ das Interpolationspolynom zu f bezüglich der Stellen $t_0, \ldots, t_n$. Dann gibt es zu jedem $x \in [a,b]$ ein $\xi_x \in [a,b]$ mit

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega(x)\,.$$

Hierbei ist $\omega(t) := \prod_{i=0}^{n}(t - t_i)$.

Beweis: Für $x \in \{t_0, \ldots, t_n\}$ ist die Aussage offenbar richtig. Für $x \notin \{t_0, \ldots, t_n\}$ setzen wir $K_x := \frac{f(x)-p(x)}{\omega(x)}$ und betrachten die Funktion

$$\psi(t) = f(t) - p(t) - K_x\omega(t)\,.$$

ψ besitzt mindestens die $n+2$ Nullstellen $\{x, t_0, \ldots, t_n\}$. Nach dem Satz von Rolle hat ψ' dann mindestens $n+1$ Nullstellen, ψ'' mindestens n Nullstellen, usw. bis $\psi^{(n+1)}$ hat mindestens eine Nullstelle $\xi = \xi_x$. Für diese gilt

$$0 = \psi^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - K_x(n+1)!$$

woraus die Behauptung folgt.

Beispiel: Wir betrachten die Funktion

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

im Intervall $[0,1]$ und interpolieren sie an äquidistanten Stützstellen $t_0 = 0$, $t_1 = h$, $\ldots$, $t_n = nh$ mit $h = \frac{1}{n}$. Dann gilt $f^{(k)}(x) = \frac{(-1)^k k!}{(1+x)^k}$. Wie man sich leicht überlegt, kann man $|\omega(x)|$ (sehr grob!) abschätzen als

$$|\omega(x)| \leq h(2h)\cdots(nh) = \frac{1}{n}\cdot\frac{2}{n}\cdots\frac{n}{n} \leq \left(\frac{1}{2}\right)^{n/2} = 2^{-\frac{n}{2}}$$

Somit gilt für den Interpolationsfehler über das ganze Intervall

$$\sup_{x\in[a,b]} |f(x) - p(x)| \leq 2^{-\frac{n}{2}}\,.$$

Wenn man n und x fest vorgegeben hat, ist die Abschätzung noch einfacher. Z.B. ist der Interpolationsfehler für $n = 4$ und $x = \frac{1}{3}$ abschätzbar als

$$|f(x) - p(x)| \leq \frac{1}{3}\left(\frac{1}{3} - \frac{1}{4}\right)\left(\frac{1}{2} - \frac{1}{3}\right)\left(\frac{3}{4} - \frac{1}{3}\right)\frac{2}{3} = \frac{10}{3\cdot 12\cdot 6\cdot 12\cdot 3} \approx 1.3\cdot 10^{-3}$$

Aber: Normalerweise fallen die Ableitungen nicht schnell genug ab, um eine Approximation für $n \to \infty$ durch Polynominterpolation mit äquidistanten Stützstellen zu gewährleisten! Ein bekanntes Gegenbeispiel ist $\frac{1}{1+x^2}$ auf dem größeren Intervall $[-5,5]$, was wir im nächsten Abschnitt sehen werden.

6.2.4 Stabilität

Forderung: Die Interpolationsabbildung $I : C^0([a,b]) \to \mathcal{P}^n$ heißt stabil, wenn eine (moderate) Konstante $C > 0$ existiert, so dass für $p = If$ gilt

$$\|p\|_{C^0([a,b])} \leq C\|f\|_{C^0([a,b])}\,.$$

Bemerkung: Dies kann man auch als eine gute Kondition der Abbildung I von $C^0([a,b])$ in den Raum $\mathcal{P}^n$ (beide versehen mit der Maximumsnorm) auffassen.

Beispiel: Wir interpolieren die Funktion

$$f(x) = \frac{1}{1+x^2}$$

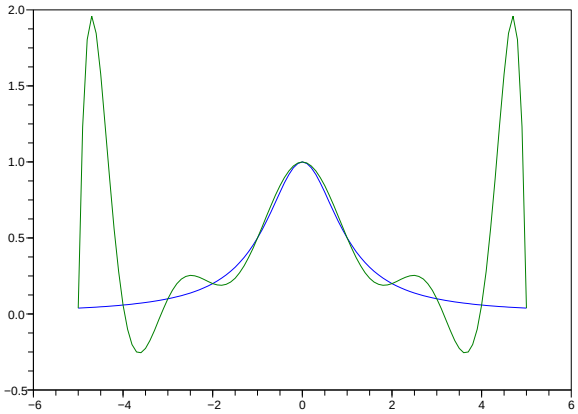
auf dem Intervall $[a,b] = [-5,5]$ zu den äquidistanten Stützstellen

$$t_k = a + kh\,, \quad k = 0,\ldots,n\,, \quad h = \frac{b-a}{n}$$

mit Polynomen n-ten Grades.

Programm: (Polynominterpolation)

Beobachtung:



Für großes n hat das interpolierende Polynom immer größere Oszillationen an den Intervallrändern. Die Stabilität ist offenbar nicht mehr gewährleistet.

Abhilfe:

- Wenn man auf die Äquidistanz der Stützstellen verzichtet, kann man ein wesentlich angenehmeres $\log(n)$-Wachstum der Stabilitätskonstanten erreichen.
- Verwende andere Funktionenräume zur Interpolation! Diesen Weg werden wir in der Folge gehen, wenn wir Räume betrachten, die nur stückweise polynomial (mit niedrigerem Grad) sind.

6.3 Stückweise polynomiale Approximationen (Splines)

Definition: Eine Unterteilung (auch Gitter genannt) $\Delta = \{t_0, \ldots, t_n\}$ des Intervalls $I = [a, b]$ besteht aus Punkten $t_i \in [a, b]$ mit $a = t_0 < t_1 < \ldots < t_n = b$. Wir nennen die Größe $h := \max_{i=1,\ldots,n} |t_i - t_{i-1}|$ die Feinheit der Unterteilung (oder auch Gitterweite).

Ansatz: Sei $\Delta = \{t_0, \ldots, t_n\}$ eine Unterteilung von $I = [a, b]$. Wir suchen zu $f \in C^0(I)$ interpolierende Funktionen in folgendem Raum

$$S_\Delta^{k,r} := \{s \in C^r(I) : s|_{[t_{i-1},t_i]} \in \mathcal{P}^k\}\,.$$

(Also: r-mal stetig differenzierbar und auf jedem Teilintervall ein Polynom k-ten Grades.) Die Interpolationsbedingung ist wie gehabt: $s \in S_\Delta^{k,r}$ soll die Bedingungen

$$s(t_i) = f(t_i)\,, \quad i = 0, \ldots, n$$

erfüllen.

6.3.1 Lineare Splines

Ansatz: Der einfachste Fall ist $k = 1$ und $r = 0$ (d.h. die Funktionen sind stückweise linear und global stetig):

$$S_\Delta^{1,0} := \{s \in C^0(I) : s|_{[t_{i-1},t_i]} \in \mathcal{P}^1,\ i = 1, \ldots, n\}\,.$$

Diesen Raum nennt man stückweise lineare Funktionen oder auch den Raum linearer Splines (Spline bedeutet etwa „Stab“ auf Englisch).

Beobachtung: Der Graph eines linearen Spline ist ein Polygonzug, der an den Stellen t_i Knicke haben darf.

Satz: $S_\Delta^{1,0}$ hat eine Basis bestehend aus den Hutfunktionen

$$\varphi_i(x) = \begin{cases} \frac{x-t_{i-1}}{t_i-t_{i-1}} & x \in [t_{i-1}, t_i] \\ \frac{t_{i+1}-x}{t_{i+1}-t_i} & x \in [t_i, t_{i+1}] \\ 0 & \text{sonst} \end{cases}, \qquad i = 1, \ldots, n-1$$

zusammen mit

$$\varphi_0(x) = \begin{cases} \frac{t_1-x}{t_1-t_0} & x \in [t_0, t_1] \\ 0 & \text{sonst} \end{cases}, \qquad \varphi_n(x) = \begin{cases} \frac{x-t_{n-1}}{t_n-t_{n-1}} & x \in [t_{n-1}, t_n] \\ 0 & \text{sonst} \end{cases}.$$

Diese Basis erfüllt $\varphi_i(t_j) = \delta_{ij}$ (sieht man einfach durch Einsetzen).

Beweis: Die Dimension des Raums aller (auch der unstetigen) stückweise linearen Funktionen auf n Intervallen ist $2n$. $S_\Delta^{1,0}$ ist nun ein Teilraum davon, der durch $n-1$ Stetigkeitsbedingungen

$$\lim_{t \uparrow t_i} s(t) = \lim_{t \downarrow t_i} s(t)\,, \quad i = 1, \ldots, n-1$$

eingeschränkt wird, was zu einer Dimension $n+1$ führt. Dies entspricht aber genau der Zahl der oben definierten Funktionen φ_i. Diese sind wegen $\varphi_i(t_j) = \delta_{ij}$ linear unabhängig und bilden daher eine Basis.

Anwendung: Wegen $\varphi_i(t_j) = \delta_{ij}$ ist zu $f \in C^0([a,b])$ eine Interpolierende $s = I_\Delta^{1,0} f \in S_\Delta^{1,0}$ mit $(I_\Delta^{1,0} f)(t_i) = f(t_i)$ durch

$$I_\Delta^{1,0} f(x) = \sum_{i=0}^{n} f(t_i)\varphi_i(x)$$

definiert.

Satz: Es sei $f \in C^2(I)$ und die Unterteilung Δ von I habe die maximale Gitterweite $h := \max_{i=1,\dots,n} |t_i - t_{i-1}|$. Dann gilt für alle $x \in I$ die Abschätzung

$$|f(x) - I_\Delta^{1,0} f(x)| \leq \frac{h^2}{8} \sup_{\xi \in [a,b]} |f''(\xi)| \,.$$

Beweis: Dies folgt sofort aus der Fehlerabschätzung, die wir für die Lagrange-Interpolation kennen. Auf jedem Teilintervall $[t_{i-1}, t_i]$ gilt für ein $\xi \in [a,b]$

$$|f(x) - I_\Delta^{1,0} f(x)| = \frac{|f''(\xi)|}{2} |(x - t_{i-1})(x - t_i)| \,.$$

Wir müssen also nur noch das Maximum von $|\omega(x)| = |(x - t_{i-1})(x - t_i)|$ berechnen. Das wird aber offenbar beim Scheitelpunkt der Parabel $x = \frac{t_{i-1}+t_i}{2}$ im Intervallmittelpunkt angenommen und hat den Wert $\frac{1}{4}(t_i - t_{i-1})^2 \leq \frac{h^2}{4}$.

6.3.2 Kubische Splines

Beobachtung: Die linearen Splines waren nur stetig, an den Stützstellen können Knicke auftreten. Abgesehen davon, dass die Approximationsordnung relativ niedrig ist, ist das Fehlen an Differenzierbarkeit für verschiedene Anwendungen problematisch. Insbesondere für Graphikanwendungen ist nicht einmal die stetige Differenzierbarkeit völlig zufriedenstellend, weil dem Auge auch ein Sprung in der zweiten Ableitung noch auffällt.

Abhilfe: Wir betrachten den Ansatzraum für $k = 3$ und $r = 2$, d.h.

$$S_\Delta^{3,2} := \{s \in C^2(I) : s|_{[t_{i-1},t_i]} \in \mathcal{P}^3,\ i = 1, \dots, n\} \,.$$

Diese Funktionen nennt man kubische Splines.

Frage:

- Ist der Funktionenraum $S_\Delta^{3,2}$ zur Interpolation brauchbar, d.h. ist das Problem

  $$s(t_i) = f(t_i) \,, \quad i = 0, \dots, n$$

  für beliebiges $f \in C^0(I)$ lösbar?

- Braucht man weitere Bedingungen, um einen interpolierenden Spline $s \in S_\Delta^{3,2}$ *eindeutig* zu definieren?

Lemma: Es sei $f \in C^2([a,b])$ und $s \in S_\Delta^{3,2}$ mit

$$s(t_i) = f(t_i)\,, \quad i = 0, \ldots, n$$

Dann gelten die Gleichungen

$$\int_a^b (f''(x) - s''(x))s''(x)\,dx = [(f'(x) - s'(x))s''(x)]_a^b\,,$$

woraus folgt

$$\int_a^b |f''(x) - s''(x)|^2\,dx = \int_a^b |f''(x)|^2\,dx - \int_a^b |s''(x)|^2\,dx - 2\,[(f'(x) - s'(x))s''(x)]_a^b\,.$$

Beweis: Zweimalige partielle Integration auf den Teilintervallen liefert

$$\begin{aligned}
&\int_a^b (f''(x) - s''(x))s''(x)\,dx \\
&= \sum_{i=1}^n [(f'(x) - s'(x))s''(x)]_{t_{i-1}}^{t_i} - \int_a^b (f'(x) - s'(x))s'''(x)\,dx \\
&= [(f'(x) - s'(x))s''(x)]_a^b - \sum_{i=1}^n [(f(x) - s(x))s'''(x)]_{t_{i-1}}^{t_i} + \int_a^b (f(x) - s(x))s^{(4)}(x)\,dx \\
&= [(f'(x) - s'(x))s''(x)]_a^b
\end{aligned}$$

wegen der Interpolationsbedingungen, und weil $s^{(4)} \equiv 0$ auf jedem Teilintervall ist. Nun folgt die zweite Aussage aus der ersten wegen

$$\int_a^b |f''(x) - s''(x)|^2\,dx = \int_a^b |f''(x)|^2\,dx - \int_a^b |s''(x)|^2\,dx - 2\int_a^b (f''(x) - s''(x))s''(x)\,dx$$

Folgerung: Der die Funktion $f \in C^2([a,b])$ interpolierende Spline $s \in S_\Delta^{3,2}$, der eine der Randbedingungen

1. $s'(a) = f'(a)$ und $s'(b) = f'(b)$ (vollständige Splineinterpolation)
2. $s''(a) = s''(b) = 0$ (natürliche Splineinterpolation)

erfüllt, minimiert das Funktional (Funktional: Abbildung von Funktionen in die reellen Zahlen)

$$Q(u) := \int_a^b |u''(x)|^2\,dx$$

unter allen Funktionen $u \in C^2(I)$, die 1. für alle $t_i \in \Delta$ mit f übereinstimmen und 2. im Fall der vollständigen Splineinterpolation auch $u'(a) = f'(a)$ und $u'(b) = f'(b)$ erfüllen. (BEWEIS: s interpoliert auch u.)

Interpretation: Die Größe $|u''(x)|$ kann man für kleine Ableitung $|u'(x)|$ als die Krümmung des Graphen von u interpretieren. Ein kubischer Spline nimmt also (unter Einhalten der Interpolationsbedingungen) eine Art minimaler Krümmung an.

Folgerung: Der eine Funktion $f \in C^2([a,b])$ interpolierende Spline $s \in S_\Delta^{3,2}$, welcher zusätzlich eine der Randbedingungen

1. $s'(a) = f'(a)$ und $s'(b) = f'(b)$ (vollständige Splineinterpolation)
2. $s''(a) = s''(b) = 0$ (natürliche Splineinterpolation)

erfüllt, ist eindeutig bestimmt.

Beweis: Wenn es zwei solche Splines s_1 und s_2 gäbe, so müssten beide $Q(u) = \int_a^b |u''(x)|^2\,dx$ minimieren, woraus insbesondere $Q(s_1) = Q(s_2)$ folgt. Nach dem Lemma ergibt sich dann auch

$$Q(s_1 - s_2) = \int_a^b |s_1''(x) - s_2''(x)|^2\,dx = 0$$

folgt. Die Funktion $w = s_1 - s_2$ ist damit linear auf jedem Teilintervall $[t_{i-1}, t_i]$ und erfüllt obendrein $w(t_i) = 0$ für $i = 0, \ldots, n$. Hieraus folgt aber $w \equiv 0$.

Folgerung: Für jedes $f \in C^2([a,b])$ gibt es genau eine Splineinterpolierende $s \in S_\Delta^{3,2}$ zu vollständigen (alternativ: natürlichen) Randbedingungen.

Beweis: Der Raum $S_\Delta^{3,2}$ ist ein Teilraum der stückweise kubischen Polynome, welcher $4n$ Freiheitsgerade (Dimensionen) hat. Eingeschränkt wird er durch $3n-3$ lineare Übergangsbedingungen (Stetigkeit von Funktion, erster und zweiter Ableitung), was eine Dimension von $S_\Delta^{3,2}$ von $n+3$ liefert. Diese werden dann durch $n+1$ Interpolationsbedingungen und zwei Randbedingungen spezifiziert. Dieser letzte Prozess entspricht dem Lösen eines linearen Systems der Dimension $n+3$, für welches die Lösungen (wie wir gezeigt haben) eindeutig sind. Ein Standardergebnis der linearen Algebra sagt, dass dies äquivalent zur Existenz einer Lösung für alle Vorgaben ist.

6.3.3 Berechnung eines interpolierenden kubischen Splines

Beobachtung: Die zweite Ableitung eines kubischen Splines $s \in S_\Delta^{3,2}$ ist ein stückweise linearer Spline $s'' \in S_\Delta^{1,0}$. s'' ist daher durch die Werte $M_i = s''(t_i)$ eindeutig bestimmt. Man nennt diese Größen die Momente.

Satz: Für einen Spline $s \in S_\Delta^{3,2}$ seien die Momente $m = s'' \in S_\Delta^{1,0}$ und die Werte $f_i = s(t_i)$ gegeben. Auf dem Teilintervall $[t_{i-1}, t_i]$ gilt mit $\hat{t} = \frac{t - t_{i-1}}{t_i - t_{i-1}} = \frac{t - t_{i-1}}{h_i}$

$$m_i(t) = \hat{m}_i(\hat{t}) = (1 - \hat{t})M_{i-1} + \hat{t}M_i$$

und

$$s_i(t) \;=\; \hat{s}_i(\hat{t}) \;=\; (1-\hat{t})f_{i-1} + \hat{t}f_i - \frac{h_i^2\hat{t}(1-\hat{t})}{6}\left((1+\hat{t})M_i + (1+(1-\hat{t}))M_{i-1}\right)$$

Beweisskizze: Bei gegebenen Momenten $m = s''$ kann der Spline s auf jedem Teilintervall durch zweifache Integration erhalten werden: es gilt nämlich

$$s(t) = \int_{t_{i-1}}^{t}\int_{t_{i-1}}^{\tau} s''(r)\,dr\,d\tau + At + B$$

und die Werte von A und B können so gewählt werden, dass s die Interpolationsbedingungen an den Intervallenden erfüllt. Für die genaue Rechnung wird auf die Übung verwiesen.

Satz: Es sei $I = [a,b]$ und $\Delta = \{t_0,\dots,t_n\}$ wie oben. Dann sind die Momente $(M_i)_{i=0,\dots,n}$ des eine Funktion $f \in C^1([a,b])$ vollständig interpolierenden kubischen Splines $s \in S^{3,2}_\Delta$ die Lösung des Gleichungssystems

$$\begin{aligned}
\frac{h_i}{6}M_{i-1} + \frac{h_i+h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} &= \frac{f_{i+1}-f_i}{h_{i+1}} - \frac{f_i-f_{i-1}}{h_i}\,, && i = 1,\dots,n-1\\
\frac{h_1}{3}M_0 + \frac{h_1}{6}M_1 &= \frac{f_1-f_0}{h_1} - f'(a)\,, && i = 0\\
\frac{h_n}{3}M_n + \frac{h_n}{6}M_{n-1} &= f'(b) - \frac{f_n-f_{n-1}}{h_n}\,, && i = n\,.
\end{aligned}$$

Beweis: Die Gleichungen ergeben sich durch die Forderung nach stetiger Differenzierbarkeit und den vollständigen Randbedingungen. Auf $[t_{i-1},t_i]$ gilt

$$s_i'(t) = \frac{1}{h_i}\hat{s}'(\hat{t}) = \\ \frac{f_i-f_{i-1}}{h_i} - \frac{h_i}{6}\left(\frac{d}{d\hat{t}}(\hat{t}(1-\hat{t})(1+\hat{t}))M_i + \frac{d}{d\hat{t}}(\hat{t}(1-\hat{t})(1+(1-\hat{t})))M_{i-1}\right)$$

Insbesondere haben wir

$$\begin{aligned}
s_i'(t_{i-1}) &= \frac{1}{h_i}\hat{s}'(0) = \frac{f_i-f_{i-1}}{h_i} - \frac{h_i}{6}(M_i + 2M_{i-1})\\
s_i'(t_i) &= \frac{1}{h_i}\hat{s}'(1) = \frac{f_i-f_{i-1}}{h_i} + \frac{h_i}{6}(2M_i + M_{i-1})
\end{aligned}$$

Die Forderung $s_i'(t_i) = s_{i+1}'(t_i)$ übersetzt sich dann in die Gleichungen für $i = 1,\dots,n-1$, die Randbedingung $s_1'(a) = f'(a)$ liefert die Gleichung für $i = 0$ und $s_n'(b) = f'(b)$ die Gleichung für $i = n$.

Bemerkung: Das entstehende Gleichungssystem hat eine stark diagonaldominante Systemmatrix A, d.h. $\sum_{j\neq i}|A_{ij}| < A_{ii}$. Hieraus folgt insbesondere die Regularität (siehe Übung).

6.3.4 Fehlerabschätzung der kubischen Splines

Satz: Seien I, Δ wie bisher, und $k, l \in \mathbb{N}$ mit $0 \leq l \leq 2 \leq k \leq 4$ und $f \in C^k(I)$. Dann erfüllt der f vollständig *(!)* interpolierende kubische Spline $s \in S_\Delta^{3,2}$ die Abschätzung

$$\max_{x \in I} |\frac{d^l s}{dx^l}(x) - \frac{d^l f}{dx^l}(x)| \leq C h^{k-l} \max_{x \in I} |\frac{d^k f}{dx^k}(x)| \,.$$

Spezialfall: Für $k = 4$ und $l = 0$ ergibt sich, dass für $f \in C^4(I)$ gilt

$$\max_{x \in I} |f(x) - s(x)| \leq C h^4 \max_{x \in I} |\frac{d^4 f}{dx^4}(x)|$$

Bemerkung: Für den interpolierenden Spline mit natürlichen Randbedingungen gelten die obigen optimalen Abschätzungen nur, wenn auch f die „natürlichen“ Randbedingungen $f''(a) = f''(b) = 0$ erfüllt.

6.3.5 Anwendung

Beispiel: Wir interpolieren wieder die Funktion $f(x) = \frac{1}{1+x^2}$ auf dem Intervall $[-5, 5]$, diesmal mit Hilfe von vollständigen kubischen Splines.

Programm:

```
function y = f (x)
  y = 1 ./ (1 + x .* x);
endfunction

function M = compute_moments (t, f, Dfa, Dfb)
  n = size(t,1);
  A = zeros(n,n);
  b = zeros(n,1);
  // setup of the interior rows
  for i=2:n-1
    A(i,i) = (t(i+1)-t(i-1))/3;
    A(i,i-1) = (t(i)-t(i-1))/6;
    A(i,i+1) = (t(i+1)-t(i))/6;
    b(i) = (f(i+1)-f(i))/(t(i+1)-t(i)) ...
      - (f(i)-f(i-1))/(t(i)-t(i-1));
  end
  // setup of the boundary rows
  h = (t(2)-t(1));
  A(1,1) = h/3;
  A(1,2) = h/6;
```

```
  b(1) = (f(2)-f(1))/h-Dfa;
  h = (t(n)-t(n-1));
  A(n,n) = h/3;
  A(n,n-1) = h/6;
  b(n) = Dfb-(f(n)-f(n-1))/h;
  // finally we solve the system
  M = A\b;
endfunction

function y = spline_eval (t, f, M, x)
  // evaluates the spline specified by t,f,M at x
  n = size(t,1);
  // find the subinterval in which x lies
  for i=2:n
    if (t(i)>=x)
      break;
    end
  end
  // evaluation of the local cubic
  h = t(i)-t(i-1);
  tau = (x-t(i-1))/h;
  y = (1-tau)*f(i-1) + tau*f(i) ...
    - h*h*tau*(1-tau)/6 * (M(i)*(1+tau)+(2-tau)*M(i-1));
endfunction

function y = spline_multiple_eval (t,f,M,x)
  // evaluates a spline for each vector component of x
  n = size(x,1);
  y = zeros(n,1);
  for i=1:n
    y(i) = spline_eval (t,f,M,x(i));
  end
endfunction

function testit (t,f)
  // we use numerical differentiation for obtaining
  // the boundary values of the complete spline
  x=-5; Dfa=(f(x+1e-8)-f(x))/1e-8;
  x=5;  Dfb=(f(x+1e-8)-f(x))/1e-8;
  m = compute_moments(t,f(t),Dfa,Dfb);
  x=[-5:0.1:5]';
  y = spline_multiple_eval(t,f(t),m,x);
  plot(x,[y,f(x)]);
endfunction

testit([-5:0.25:5],f)
```

Beobachtung: Schon bei 11 Stützstellen ($h = 1$) sieht man kaum einen Unterschied mehr zwischen der Funktion und dem Spline.

7 Numerische Quadratur

7.1 Einführung

Das Ziel dieses Kapitels ist die Berechnung des Riemann-Integrals

$$I(f) = \int_a^b f(x)\, dx\,.$$

Bezeichnung: Manchmal werden wir statt I auch $I_{[a,b]}$ schreiben, insbesondere wenn wir unterschiedliche Integrationsbereiche betrachten.

Beobachtung: Die Integration I ist ein Funktional auf $C^0([a,b])$ mit

1. I ist linear: $I(\alpha f + \beta g) = \alpha I(f) + \beta I(g)$
2. I ist positiv: $I(f) \geq 0$ für $f \geq 0$.
3. Es sei
$$\Phi : [0,1] \to [a,b]\,, \quad \hat{t} \mapsto \hat{t} = a + \hat{t}(b-a)\,.$$
Dann gilt nach dem Transformationssatz
$$I_{[a,b]}(f) = (b-a) I_{[0,1]}(f \circ \Phi)\,.$$

Problem: Normalerweise gibt es zu $f \in C^0([a,b])$ keine in elementaren Funktionen ausdrückbare Stammfunktion F, mit Hilfe derer man $I(f) = F(b) - F(a)$ ausrechnen könnte.

Abhilfe: Man berechnet das Integral näherungsweise. Dies nennt man numerische Integration oder numerische Quadratur.

7.2 Quadraturformeln

Definition: Eine Quadraturformel Q zur Approximation von $\int_a^b f(t)\, dt$ hat die Form

$$Q(f) = \sum_{i=0}^{n} \lambda_i f(t_i)$$

mit den paarweise verschiedenen Knoten $\{t_0, \ldots, t_n\} \subset [a,b]$ und den Gewichten $\lambda_0, \ldots, \lambda_n \in \mathbb{R}$.

Bemerkung: Eine solche Quadraturformel Q ist offenbar linear in f.

Kriterien: Eine gute Wahl der λ_i sollte folgendes erfüllen:

1. $\sum_{i=0}^{n} \lambda_i = b - a$. Das stellt sicher, dass wenigstens $f(t) \equiv 1$ korrekt integriert wird.
2. $\lambda_i > 0$ für $i = 0, \ldots, n$. Das sichert $f \geq 0 \Rightarrow Q(f) \geq 0$.
   (Umgekehrt: Wenn ein $\lambda_i < 0$ ist, kann man sehr leicht ein $f \in C^0([a,b])$ finden mit $f \geq 0$ und $Q(f) < 0$. Wie?)

Bemerkung: Mit Hilfe des oben definierten linearen Isomorphismus $\Phi : [0,1] \to [a,b]$ kann man ausgehend von einer Quadraturformel $Q_{[0,1]}(f) = \sum_{i=0}^{n} \hat{\lambda}_i f(\hat{t}_i)$ für $I_{[0,1]}$ eine Quadraturformel $Q_{[a,b]}$ für $I_{[a,b]}$ erzeugen als

$$Q_{[a,b]}(f) = \sum_{i=0}^{n} \lambda_i f(t_i)\,, \quad t_i = \Phi(\hat{t}_i)\,,\ \lambda_i = (b-a)\hat{\lambda}_i\,.$$

7.3 Konstruktion durch Polynominterpolation

Idee: Interpoliere den Integranden durch ein Polynom n-ten Grades und integriere dieses interpolierende Polynom.

Definition: Zu den paarweise verschiedenen Stützstellen $\{t_0, \ldots, t_n\} \subset [a,b]$ seien $L_{i,n}$ wieder die Lagrange-Polynome, und $p_f = \sum_{i=0}^{n} f(t_i) L_{i,n} \in \mathcal{P}^n$ sei das interpolierende Polynom zu einer Funktion $f \in C^0([a,b])$ mit $p_f(t_i) = f(t_i)$. Mit

$$Q(f) = \int_a^b p_f(t)\,dt = \sum_{i=0}^{n} f(t_i) \int_a^b L_{i,n}(t)\,dt$$

haben wir offenbar eine Quadraturformel mit Knoten t_i und Gewichten $\lambda_i = \int_a^b L_{i,n}(t)\,dt$ gefunden.

Satz: Das so konstruierte Q ist exakt für alle Polynome n-ten Grades, d.h. für alle $p \in \mathcal{P}^n$ gilt $Q(p) = I(p)$.

Beweis: $\{L_{i,n}\}_{i=0,\ldots,n}$ ist eine Basis von $\mathcal{P}^n$. Nach Definition ist Q exakt für jedes der $L_{i,n}$ und daher wegen der Linearität von I und Q auch für alle $p \in \mathcal{P}^n$.

Bemerkung: Es gilt offenbar auch die Umkehrung: Wenn $Q(f) = \sum_{i=0}^{n} \lambda_i f(t_i)$ für alle $f \in \mathcal{P}^n$ exakt ist, so folgt

$$\lambda_i = Q(L_{i,n}) = I(L_{i,n}) = \int_a^b L_{i,n}(t)\,dt\,.$$

Satz: Es sei wieder $\Phi : [0,1] \to [a,b]$ der oben definierte lineare Isomorphismus. $\{\hat{t}_0, \ldots, \hat{t}_n\} \subset [0,1]$ und $\{t_0, \ldots, t_n\} \subset [a,b]$ seien Knoten, welche $\Phi(\hat{t}_i) = t_i$ für $i = 0, \ldots, n$ erfüllen. Dann

gilt für die zugehörigen Lagrange-Polynome $\hat{L}_{i,n}(\hat{t}) = L_{i,n}(t)$ für $t = \Phi(\hat{t})$. Für die über Polynominterpolation zu den obigen Knoten erzeugten Quadraturformeln $Q_{[0,1]} = \sum_{i=0}^{n} \hat{\lambda}_i f(\hat{t}_i)$ und $Q_{[a,b]} = \sum_{i=0}^{n} \lambda_i f(t_i)$ gilt daher

$$\lambda_i = \int_a^b L_{i,n}(t)\, dt = (b-a) \int_0^1 \hat{L}_{i,n}(\hat{t})\, dt = (b-a)\hat{\lambda}_i\,.$$

Beweis: Da Φ linear ist, ist $L_{i,n} \circ \Phi$ wieder ein Polynom n-ten Grades, welches $(L_{i,n} \circ \Phi)(\hat{t}_j) = L_{i,n}(t_j) = \delta_{ij}$ erfüllt. Wegen der eindeutigen Lösbarkeit der Interpolationsaufgabe muss $L_{i,n} \circ \Phi = \hat{L}_{i,n}$ gelten. Dann folgt aber aus dem Transformationssatz

$$\lambda_i = \int_a^b L_{i,n}(t)\, dt = (b-a) \int_0^1 \hat{L}_{i,n}(\hat{t})\, d\hat{t}\,.$$

7.3.1 Newton-Cotes-Formeln

Definition: Die Quadraturformeln, die durch Polynominterpolation mit Verwendung äquidistanter Stützstellen $t_i = a + i\frac{b-a}{n}$, $i = 0, \ldots, n$ entstehen, heißen Newton-Cotes-Formeln.

Bezeichnung: Für $u \in C^0([a,b])$ setzen wir $\|u\|_\infty := \max\limits_{x\in[a,b]} |u(x)|$.

Beispiel: Für $n = 1, 2, 3, 4$ erhalten wir mit $h = b - a$

n	$\hat{\lambda}_i = \lambda_i/h$	Name	Fehler $\lvert I(f) - Q(f)\rvert$
1	$\left(\frac{1}{2} \quad \frac{1}{2}\right)$	Trapezregel	$\frac{h^3}{12}\|f^{(2)}\|_\infty$
2	$\left(\frac{1}{6} \quad \frac{4}{6} \quad \frac{1}{6}\right)$	Simpson-Regel	$\frac{h^5}{2880}\|f^{(4)}\|_\infty$
3	$\left(\frac{1}{8} \quad \frac{3}{8} \quad \frac{3}{8} \quad \frac{1}{8}\right)$	Newtonsche $\frac{3}{8}$-Regel	$\frac{h^5}{6480}\|f^{(4)}\|_\infty$
4	$\left(\frac{7}{90} \quad \frac{32}{90} \quad \frac{12}{90} \quad \frac{32}{90} \quad \frac{7}{90}\right)$	Milne-Regel	$\frac{h^7}{1935360}\|f^{(6)}\|_\infty$

Bemerkungen:

- Für die Trapezregel $T(f) = \frac{1}{2}(f(a) + f(b))$ folgt aus der Interpolationsabschätzung

$$\|f - p_f\|_\infty \le \frac{h^2}{8}\|f''\|_\infty$$

für das an den Endpunkten interpolierende lineare Polynom $p_f \in \mathcal{P}^1$ sofort die schon recht gute Fehlerabschätzung

$$|I(f) - T(f)| = |\int_a^b (p_f(x) - f(x))\, dx| \le \int_a^b |p_f(x) - f(x)|\, dx \le \frac{h^3}{8}\|f''\|_\infty\,.$$

- Analog erhält man für die Newton-Cotes-Formel der Ordnung n die Fehlerabschätzung

$$|I(f) - Q_{NC,n}(f)| \leq C(n)h^{n+2}\|f^{(n+1)}\|_\infty .$$

  Man beachte, dass man durch die Integration einen Faktor h mehr erhält als bei der Interpolationsfehlerabschätzung.

- Durch eine genauere Analyse kann man noch die Fehlerkonstante verbessern. Unten zeigen wir diese Technik exemplarisch für die Trapezregel.

- Interessant ist zudem die erhöhte Ordnung der Newton-Cotes-Regeln für gerades n. Unten beweisen wir diese Ordnungsverbesserung exemplarisch für die Simpson-Regel.

- Ab $n = 7$ treten negative Gewichte auf. Spätestens dann werden diese Newton-Cotes-Formeln problematisch, weil Auslöschung schon bei der Integration einfacher konstanter Funktionen auftreten kann.

Satz: Für $f \in C^2([a,b])$ genügt die Trapezregel $T(f) = \frac{b-a}{2}(f(a) + f(b))$ der Abschätzung

$$|T(f) - I(f)| \leq \frac{h^3}{12}\|f''\|_\infty .$$

Beweis: Es sei $p_f \in \mathcal{P}^1$ die lineare Interpolierende von f zu den Stützstellen a und b. Nach unserem Approximationssatz gilt für $x \in [a,b]$:

$$|f(x) - p_f(x)| \leq \frac{\|f''\|_\infty}{2}(x-a)(b-x) .$$

Somit folgt aber auch

$$\begin{aligned} |T(f) - I(f)| &= |\int_a^b f(x)\,dx - \int_a^b p_f(x)\,dx| \leq \int_a^b |f(x) - p_f(x)|\,dx \\ &\leq h^2\frac{\|f''\|_\infty}{2}\int_a^b (x-a)(b-x)\,dx = h^3\frac{\|f''\|_\infty}{2}\int_0^1 \hat{x}(1-\hat{x})\,dx = h^2\frac{\|f''\|_\infty}{12} . \end{aligned}$$

Satz: Die Simpsonregel

$$S(f) = \frac{b-a}{6}(f(a) + 4f(\frac{b+a}{2}) + f(b))$$

integriert sogar alle Polynome *dritten (!)* Grades exakt. Hieraus folgt die Fehlerabschätzung

$$|S(f) - I(f)| \leq Ch^5\|f^{(4)}\|_\infty .$$

Beweis: Ohne Einschränkung sei $a = -b$ (eine einfache Verschiebung ändert nichts an $I(f)$, $Q(f)$ oder $f^{(4)}$). Wir wissen, dass S alle Polynome zweiten Grades exakt integriert. Der Clou ist nun, dass es auch das Polynom x^3 wegen $Q(f) = I(f) = 0$ exakt integriert (Symmetrie!). Nun ist aber $\mathcal{P}^3 = \mathcal{P}^2 \oplus \mathrm{Span}(x^3)$, somit werden alle Polynome dritten Grades exakt integriert. Die Fehlerabschätzung folgt dann wie folgt: p_f sei die Interpolierende *dritten* Grades zu äquidistanten Stützstellen. Für diese gilt

$$\|f - p_f\|_\infty \leq Ch^4 \|f^{(4)}\|_\infty \,,$$

woraus wie gehabt folgt

$$|I(f) - S(f)| = |\int_a^b (f - p_f)\, dx| \leq \int_a^b |p_f(x) - f(x)|\, dx \leq Ch^5 \|f^{(4)}\|_\infty \,.$$

7.4 Zusammengesetzte Formeln

Problem: Die Polynominterpolation ist für hohen Polynomgrad (wenigstens für äquidistante Knoten) instabil. Dies überträgt sich auf die Quadraturformeln.

Idee: Sei $a = x_0 < x_1 < \ldots < x_n = b$ eine Zerlegung des Intervalls $[a, b]$ in Teilintervalle $[x_{i-1}, x_i]$. Das Gesamtintegral $I_{[a,b]}$ ergibt sich dann als Summe der $I_{[x_{i-1},x_i]}$, die lokal mit einer Newton-Cotes-Formel niedrigen Grades approximiert werden können.

Bezeichnung: Eine solche Quadraturformel nennt man eine zusammengesetzte Quadraturformel.

Beispiele: Wir betrachten äquidistante Unterteilungen $x_i = a + ih$ mit $h = \frac{b-a}{n}$.

- Zusammengesetzte Trapezregel:

$$T(h) = \sum_{i=1}^{n} Q_{[x_{i-1},x_i]} = \sum_{i=1}^{n} \frac{h}{2}(f(x_{i-1}) + f(x_i)) = h\big(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\big)$$

- Zusammengesetzte Simpsonregel:

$$\begin{aligned} S(h) = \sum_{i=1}^{n} Q_{[x_{i-1},x_i]} &= \sum_{i=1}^{n} \frac{h}{6}(f(x_{i-1}) + 4f(\frac{x_{i-1}+x_i}{2}) + f(x_i)) \\ &= \frac{h}{3}\big(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) + 2\sum_{i=1}^{n} f(\frac{x_{i-1}+x_i}{2})\big) \,. \end{aligned}$$

Satz: Die zusammengesetzte Trapezregel für eine äquidistante Unterteilung $x_i = a + ih$ mit $h = \frac{b-a}{n}$ genügt der Fehlerabschätzung

$$|T(h) - I(f)| \leq (b-a)\frac{h^2}{12}\|f''\|_\infty \,.$$

Beweis: Es gibt $n = \frac{b-a}{h}$ Teilintervalle und auf jedem kann der Fehler durch $\frac{h^3}{12}\|f''\|_\infty$ abgeschätzt werden.

Bemerkung: Ebenso ergibt sich für die Simpson-Regel die Abschätzung

$$|S(h) - I(f)| \leq (b-a)\frac{h^4}{2880}\|f^{(4)}\|_\infty \,.$$

Bemerkung: Man beachte den „Verlust“ einer h-Potenz gegenüber der „lokalen“ Abschätzung.

7.5 Ausblicke

Wir konnten einige interessante und wichtige Bereiche nicht behandeln. Dazu gehören unter anderem folgende Punkte:

- Wenn man auf die Äquidistanz der Knoten verzichtet, kann man durch geschickte Wahl derselben durch Polynominterpolation Quadraturformeln erzeugen, die bei s Knoten noch Polynome vom Grad $2s-1$ exakt integrieren. Dies ist die sogenannte Gauß-Quadratur. Die einfachste solche Formel ist die Mittelpunktsregel

$$Q_{\mathrm{MP}}(f) = f\big(\frac{a+b}{2}\big)(b-a)\,.$$

- Wenn der Integrand glatt ist und die Unterteilungen regelmäßig sind, kann man etwa für die zusammengesetzte Trapezregel ein Fehlerverhalten wie

$$Q(h) = I(f) + C_1 h^2 + C_2 h^4 + \ldots$$

  zeigen. Dies kann man ausnutzen, indem man sukzessive Approximationen $Q(h)$, $Q(\frac{h}{2})$, ... durch ein Polynom in der Variablen $x = h^2$ interpoliert, welches man dann an $x = 0$ auswertet, um eine Approximation höherer Ordnung zu $I(f)$ zu erhalten. Diese Technik heißt Extrapolation und kann in verschiedenen Situationen angewendet werden, z.B. auch bei der numerischen Differentiation (siehe Übung 10/1). Die Anwendung auf die numerische Quadratur heißt Romberg-Quadratur. Auch Übung 11/1 kann man als eine Stufe einer solchen Extrapolation ansehen.

- Für viele Integranden ist eine nicht äquidistante Wahl des Gitters von Vorteil. Die automatische Gitteranpassung an einen bestimmten Integranden führt dabei zu adaptiven Quadraturverfahren. Allerdings geht dadurch normalerweise die Möglichkeit zur Extrapolation verloren.

8 Numerische Lösung gewöhnlicher Differentialgleichungen

8.1 Grundlagen

Definition: Es sei $f \in C^0(\mathbb{R} \times \mathbb{R}^n, \mathbb{R}^n)$. Zu einem gegebenen Startwert $u_0 \in \mathbb{R}^n$ und einem $T > 0$ suchen wir eine Funktion $u \in C^1([0,T], \mathbb{R}^n)$ mit

$$\begin{aligned} \frac{du}{dt}(t) &= f(t, u(t))\,, & t \in]0, T[\,, \\ u(0) &= u_0\,. \end{aligned}$$

Dies nennen wir eine gewöhnliche Differentialgleichung.

Bemerkung: Manchmal bezeichnet man nur den Fall $n = 1$ als Differential*gleichung* und nennt den Fall $n > 1$ ein System gewöhnlicher Differentialgleichungen.

Bemerkungen:

- Wenn f nicht von x abhängt, so kann man die Lösung der Differentialgleichung durch eine einfache Integration erhalten:
$$u(t) = \int_0^t f(t)\, dt\,.$$
- Wenn f nicht von t abhängt, so nennt man die zugehörige Differentialgleichung autonom.
- In vielen Anwendungen hat die Variable t die Bedeutung der Zeit. Die Ableitung nach der Zeit bezeichnen vor allem Physiker oft mit einem Punkt ($\dot{u}(t) := \frac{du}{dt}(t)$).

8.2 Beispiele

8.2.1 Pendelgleichung

Erinnerung: In der Einführung hatten wir bereits Differentialgleichungen für das Stabpendel hergeleitet. Diese waren

$$\begin{aligned} \dot{\theta}(t) &= \omega(t) \\ \dot{\omega}(t) &= -\frac{g}{L} \sin \theta(t) \end{aligned}$$

wobei θ den Auslenkungswinkel, ω die Winkeländerungsgeschwindigkeit, g die Gravitationskonstante, und L die Länge des Stabs bezeichnet.

Transformation: Wir setzen $u(t) = \big(\theta(t), \omega(t)\big)^t$ und erhalten

$$\dot{u}(t) = f(u(t)) \quad \text{mit} \quad f(u) = \begin{pmatrix} u_2 \\ -\frac{g}{L} \sin u_1 \end{pmatrix} .$$

8.2.2 Räuber-Beute-Modell

Modell: Auf einer Insel gebe es Populationen von Raubtieren $R = R(t)$ und Beutetieren $B = B(t)$, die sich mit der Zeit ändern können. Ein einfaches Modell für die zeitliche Veränderung der Populationen ist

$$\begin{aligned} \dot{B}(t) &= \mu_B B(t) - \alpha B(t) R(t) \\ \dot{R}(t) &= -\mu_R R(t) + \beta B(t) R(t) \end{aligned}$$

Hierbei sind $\mu_B, \mu_R, \alpha, \beta$ positive reelle Konstanten.

Skalierung: Um die Eigenschaften des Modells zu studieren, wählt man oft die Einheiten so, dass die Gleichung möglichst einfach wird. Im obigen Fall lässt sich durch Wahl der Einheiten von B, R und t folgende Form erreichen:

$$\begin{aligned} \dot{B}(t) &= B(t) - B(t) R(t) \\ \dot{R}(t) &= -\mu R(t) + B(t) R(t) \end{aligned}$$

Bemerkungen:

- Man sieht, dass das Modell in seinem Verhalten im wesentlichen von einem einzigen Parameter $\mu \in \mathbb{R}$ abhängt.
- In Anwendungen ist es oft praktischer, die üblichen Einheiten zu verwenden. Hier verzichtet man daher meist auf solche Vereinfachungen.

Transformation: Wir setzen $u(t) = \big(B(t), R(t)\big)^t$ und erhalten

$$\dot{u}(t) = f(u(t)) \quad \text{mit} \quad f(u) = \begin{pmatrix} u_1 - u_1 u_2 \\ -\mu u_2 + u_1 u_2 \end{pmatrix} .$$

8.3 Geometrische Interpretation

Problem: Der Einfachheit halber betrachten wir die autonome Differentialgleichung

$$\dot{u}(t) = f(u(t)), \ t \in [0, T], \quad u(0) = u_0$$

mit $f \in C^0(\mathbb{R}^n, \mathbb{R}^n)$.

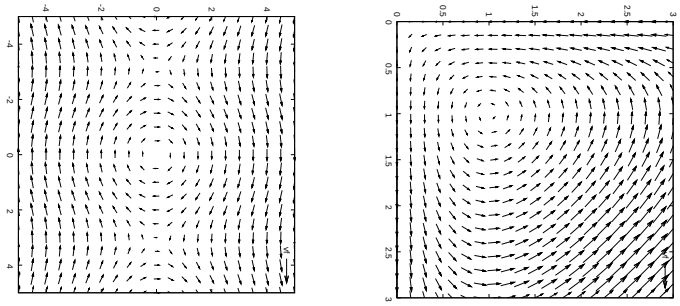
Bemerkung: Sowohl die Pendelgleichung als auch das Räuber-Beute-Modell sind autonom. Nicht autonome Varianten würden sich ergeben, wenn etwa der Aufhängungspunkt des Stabpendels sich bewegen würde, oder wenn Geburts- oder Sterberaten im Räuber-Beute-Modell von der Zeit abhingen.

Interpretation: Die Lösung $u : [0, T] \to \mathbb{R}^n$ beschreibt ein Kurvenstück, welches

1. den Anfangspunkt $u(0) = u_0$ hat, und welches
2. für alle t die Ableitung $\frac{du}{dt}(t) = f(u(t))$ hat.

Bezeichnung: Die Abbildung $f : \mathbb{R}^n \to \mathbb{R}^n$ nennt man ein Vektorfeld. Kurvenstücke $u : [a, b] \to \mathbb{R}^n$, die in jedem ihrer Punkte die Gleichung $\frac{du}{dt}(t) = f(u(t))$ erfüllen, nennt man Integralkurven des Vektorfelds.

Beispiel: Die Vektorfelder für Pendel ($\frac{g}{L} = 1$) und Räuber-Beute-Modell ($\mu = 1$) sehen folgendermaßen aus:



8.4 Numerische Lösung

Problem: *Sehr selten* kann man für eine gewöhnliche Differentialgleichung eine geschlossene Lösung der Form

$$u(t) = \ldots \text{Ausdruck in } t \ldots$$

angeben. Genau wie bei der Integration ist man meist auf die numerische Lösung angewiesen.

8.4.1 Das explizite Euler-Verfahren

Problem: Wir suchen eine Approximation der Lösung $u : [0,T] \to \mathbb{R}^n$ der gewöhnlichen Differentialgleichung

$$\dot{u}(t) = f(t, u(t)) \,, \; t \in [0,T] \,, \quad u(0) = u_0$$

mit $f \in C^0(\mathbb{R} \times \mathbb{R}^n, \mathbb{R}^n)$.

Algorithmus: (Explizites Euler-Verfahren) Unterteile $[0,T]$ in N Intervalle der Länge $h = \frac{T}{N}$. Approximiere $u : [0,T] \to \mathbb{R}^n$ dann an den Stellen $t_k = kh$ durch folgende Werte $y_k \in \mathbb{R}^n$:

$$y_0 = u_0 \,, \quad y_k = y_{k-1} + hf(t_{k-1}, y_{k-1}) \,, \quad k = 1, \ldots, N \,.$$

Bemerkungen:

- Der Aufwand des Verfahrens ist im wesentlichen gegeben durch N Auswertungen der Funktion f.
- Wenn man $u : [0,T] \to \mathbb{R}^n$ nicht nur an den Stellen $t_k = kh$ sondern durch eine auf ganz $[0,T]$ definierte Funktion approximieren will, kann man die Punkte (t_k, y_k) einfach linear interpolieren.
- Angewandt auf ein Integrationsproblem führt das explizite Euler-Verfahren auf die Quadraturformel

  $$\int_0^T f(t)\, dt \sim h(f(t_0) + f(t_1) + \ldots + f(t_{N-1})) =: Q_{\mathrm{EE}}(h) \,.$$

  Diese Formel hat im wesentlichen den gleichen Aufwand wie Trapez- oder Mittelpunktregel, besitzt aber im Gegensatz zu diesen Verfahren nur die Fehlerordnung $O(h)$ statt $O(h^2)$. Es macht daher für die einfache Quadratur wenig Sinn.

8.4.2 Verfahren höherer Ordnung

Problem: Die Fehlerordnung $O(h)$ ist für viele Anwendungen nicht genau genug.

Abhilfe: Man konstruktiert Verfahren höherer Ordnung. Hierfür gibt es verschiedene Ansätze, von denen wir nur wenige Verfahren vorstellen.

Algorithmus: Eine Erweiterung der Trapezregel ist das Verfahren von Heun:

$$y_0 = u_0 \,, \quad y_k = y_{k-1} + \frac{h}{2}(\underbrace{f(t_{k-1}, y_{k-1})}_{f_{k-1}} + f(t_k, y_{k-1} + hf_{k-1})) \,.$$

Dies hat pro Schritt den doppelten Aufwand wie das explizite Euler-Verfahren, aber auch die Fehlerordnung $O(h^2)$.

Algorithmus: (Runge-Kutta-Formel vierter Ordnung) Dies ist folgende Erweiterung der Simpson-Regel:

$$y_0 = u_0\,, \quad y_k = y_{k-1} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)\,,$$

wobei

$$k_1 = f(t_{k-1}, y_{k-1}) \qquad k_2 = f(t_{k-1} + \frac{h}{2}, y_{k-1} + \frac{h}{2}k_1)$$
$$k_3 = f(t_{k-1} + \frac{h}{2}, y_{k-1} + \frac{h}{2}k_2) \qquad k_4 = f(t_k, y_{k-1} + hk_3)$$

Dieses Verfahren hat pro Schritt im wesentlichen den vierfachen Aufwand wie das explizite Euler-Verfahren, dafür aber auch die Fehlerordnung $O(h^4)$.

8.4.3 Implementation

Programm: (ode.sci)

```
//
   ////////////////////////////////////////////////////////////////////////

/// ordinary differential equations
//
   ////////////////////////////////////////////////////////////////////////


function y = EE_step (f,t,x,dt)
  y = x+dt*f(t,x);
endfunction

function y_ = RK4_step (f, t_n, y_n, h_n)
  h2 = h_n/2;
  k1 = f(t_n,y_n);
  k2 = f(t_n+h2, y_n+h2*k1);
  k3 = f(t_n+h2, y_n+h2*k2);
  k4 = f(t_n+h_n, y_n+h_n*k3);
  y_ = y_n + h_n/6*(k1+2*(k2+k3)+k4);
endfunction

global stepper;
stepper = EE_step;

function [x,t] = trajectory (f,x0,t0,t1,N)
  global stepper;
  dim = size(x0,1);
  t = linspace(t0,t1,N);
  x = zeros (dim,N);
```

```
  x(:,1) = x0;
  for i=2:N
    dt = t(i)-t(i-1);
    x(:,i)=stepper(f,t(i),x(:,i-1),dt);
  end
endfunction

function plot_trajectory (f,x0,t0,t1,N)
  [x,t] = trajectory(f,x0,t0,t1,N);
  plot(x(1,:),x(2,:));
endfunction

function plot_paths (f,x0,t0,t1,N)
  [x,t] = trajectory(f,x0,t0,t1,N);
  plot(t,x);
endfunction

// Predator-prey model
global mu;
mu = 1.0;
function y = lotka_volterra (t,x)
  global mu;
  y = [x(1)*(1.0-x(2)); ...
       x(2)*(x(1)-mu)];
endfunction

stepper = EE_step;
//plot_paths (lotka_volterra ,[1.0;1.0],0,20,100)

// Konvergenzuntersuchung anhand e^x

function f = identity (t,x)
  f = x;
endfunction

function errors = errors (N)
  for i=2:N
    x = trajectory(identity ,[1],0,1,2^i);
    errors(i) = x(2^i)-exp(1);
  end
endfunction

//stepper=EE_step;
//errors(8)
//stepper=RK4_step;
//errors(8)
```

8.4.4 Anwendung

Beobachtung: Bei Anwendung auf das Räuber-Beute-Modell mit $\mu = 1$ sehen wir:

- $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ ist ein stationärer Punkt, das heißt er bleibt für alle Zeiten konstant. Wir haben ein perfektes biologisches Gleichgewicht.
- Wenn der Anfangswert u_0 nicht dieses Gleichgewicht ist, so beobachten wir Oszillationen, die umso stärker werden, je weiter u_0 vom Gleichgewicht entfernt ist. Diese Oszillationen beobachtet man auch in der Natur!
- Das explizite Euler-Verfahren ist viel schlechter als das Runge-Kutta-Verfahren.

Beobachtung: Die Untersuchung des Konvergenzverhaltens von explizitem Euler-Verfahren und Runge-Kutta-Verfahren angewandt auf das Problem

$$\dot{u}(t) = u(t)\,,\ t \in [0,1]\,, \quad u(0) = 1$$

mit exakter Lösung $u(t) = e^t$ liefert folgende Tabelle für den Fehler $y_N - e^1$ ($h = \frac{1}{N}$):

N	EE	RK-4
4	- 0.3479115	- 0.0002121
8	- 0.1717821	- 0.0000084
16	- 0.0854031	- 0.0000004
32	- 0.0425855	- 2.388E-08
64	- 0.0212644	- 1.419E-09
128	- 0.0106252	- 8.651E-11
256	- 0.0053109	- 5.341E-12

Wir sehen hier auch zahlenmäßig die viel bessere Konvergenz des Runge-Kutta-Verfahrens vierter Ordnung.

8.4.5 Ausblicke

- Die numerische Quadratur ist ein Spezialfall der numerischen Lösung von gewöhnlichen Differentialgleichungen. Vieles dort im Ausblick erwähnte macht daher auch hier Sinn, insbesondere Extrapolation und adaptive Verfahren.
- Es gibt gut gestellte Probleme (sogenannte steife Differentialgleichungen), bei denen die hier vorgestellten expliziten Zeitschrittverfahren versagen (genauer: viel zu kleine Zeitschritte benötigen). Die Abhilfe sind hier implizite Zeitschrittverfahren, bei denen in jedem Schritt ein nichtlineares Gleichungssystem gelöst werden muss.
- Eine weitaus komplexere Klasse von Problemen als die gewöhnlichen Differentialgleichungen sind partielle Differentialgleichungen, bei denen Ableitungen in mehrere Koordinatenrichtungen auftreten. Deren numerische Behandlung übersteigt den Rahmen dieser Vorlesung leider erheblich.

Index