Numerische Lineare Algebra im WS 2012/2013

Frank Wübbeling

29. März 2013

Inhaltsverzeichnis

Kapitel 0

Einleitung

Der vorliegende Text entstand als Begleitmaterial zur Vorlesung Numerische Lineare Algebra im Wintersemester 2012/1013. Die Vorlesung richtet sich an Studierende des Bachelorstudiengangs Mathematik im dritten Semester sowie Studierende in den Lehramtsstudiengängen Mathematik. Für die Korrektheit des Textes wird keinerlei Garantie übernommen, vermutlich sind noch reichlich Schreibfehler enthalten. Für Bemerkungen und Korrekturen bin ich dankbar.
Macht es Sinn, der großen, bereits existierenden Zahl von Skripten zu Einführungsveranstaltungen der Numerischen Mathematik noch ein weiteres hinzuzufügen? Die Antwort ist wohl ja, denn zumindest die Auswahl der Themen und vor allem Schwerpunkte im breiten Spektrum geschieht subjektiv durch den Dozenten.
Da der Großteil der Studierenden heute kaum noch einen physikalischen Hintergrund hat, habe ich auf die Darstellung der Beziehungen zwischen Angewandter Mathematik und Physik, wie sie in den klassischen Lehrbüchern und Vorlesungen üblich war, größtenteils verzichtet. Übungen zu den einzelnen Kapiteln finden sich im Netz, ebenso eine ausführliche (subjektive) Literaturliste.
Ich habe mich bemüht, zu den vorgestellten Algorithmen eine Beispiel–Implementation in Matlab zu liefern. Einige Programme nutzen dabei die Imaging–Toolbox oder die SymbolicMath–Toolbox. Die zugehörigen Dateien sind in der PDF–Datei enthalten. Klick auf die jeweilige Textstelle öffnet die Beispielimplementation in Matlab. Ebenso sind alle Bilder beigefügt, Klick liefert jeweils das zugehörige Bild. Dies funktioniert in Acrobat (Reader) und in einigen anderen PDF–Readern, in vielen PublicDomain–Readern aber nicht.

Billerbeck, im Herbst 2012

Frank Wübbeling

Kapitel 1

Angewandte Mathematik

Angewandte Mathematik überträgt mathematische Konzepte auf die Realität und macht sie so für praktische Probleme nutzbar. Das Zusammenspiel zwischen Theorie (Mathematik) und Praxis (Anwendung) ist der Reiz dieser Disziplin und bildet die Grundlage für die modernen Naturwissenschaften (Physik, Biologie, Chemie, Geophysik, Medizin, ...), aber auch für andere Gebiete wie die analytischen Wirtschaftswissenschaften.
Wichtige Aufgaben sind dabei

Simulation: Ein Prozess wird auf dem Rechner nachgebildet, z.B. in der Wettervorhersage oder der Klimaforschung.

Optimierung: Es werden optimale Parameter für einen beeinflussbaren Prozess gesucht, z.B. in der Produktionsplanung oder der Strahlentherapie.

Ursachenforschung: Ermittlung von Größen, die nur indirekt gemessen werden können, z.B. in der Tomographie oder beim Scharfrechnen von geglätteten Bildern.

Die Vorgehensweise bei der Lösung eines Problems mit Hilfe der Mathematik ist:

1. Genaue Formulierung des Problems
2. Übertragung in die Mathematik (Modellierung)
3. Vereinfachung des Modells, so dass es lösbar wird (Diskretisierung)
4. Design eines Lösungswegs (Algorithmus)
5. Implementation des Algorithmus
6. Interpretation

7. Anwendung

Die numerische Mathematik ist dabei für die Schritte 3-5 zuständig. Jeder der mathematischen Schritte unterliegt dabei einer Untersuchung mit analytischen Methoden:

- Wie genau ist das Modell?
- Wie genau ist das vereinfachte Modell?
- Liefert der Algorithmus das richtige Ergebnis?
- Was passiert bei Messfehlern?
- Wie effizient (schnell) ist der Algorithmus?

Wir betrachten einige Beispiele.

Beispiel 1.1 *(Computertomographie)*
Ein Röntgenbild zeigt immer zweidimensionale Schattenbilder. Eine dreidimensionale Lagebeziehung (etwa: liegt der Tumor vor oder hinter dem Knochen?) kann man den Bildern nicht entnehmen. Mitte des letzten Jahrhunderts kam die Idee auf, viele Röntgenbilder aufzunehmen und daraus eine dreidimensional Darstellung zu berechnen. Ein einfaches mathematisches Modell: Sei R die Position der Röntgenquelle, P eine Position auf der Fotoplatte. Sei weiter $g(x)$, $g : \mathbb{R}^3 \mapsto \mathbb{R}$ die Stärke, mit der ein Röntgenstrahl am Punkt x geschwächt wird.
Die Helligkeit der Fotoplatte am Punkt P ist umso größer, je weniger der Röntgenstrahl auf seinem (geraden) Weg von R nach P geschwächt wurde: Ging der Strahl durch Knochen (dort ist g groß), so bleibt die Fotoplatte schwarz, ging er durch Luft, so wird die Platte weiß. Auf diese Weise bekommen wir eine zweidimensionale Projektion von g auf die Fotoplatte. Wir hätten aber gern nicht die Projektion, sondern g selbst - die Fragestellung ist daher: Wie berechnet man g aus seinen zweidimensionalen Projektionen?
Mathematisch ist die Schwärzung proportional zum Linienintegral von g über die Linie zwischen R und P. Die mathematische Fragestellung lautet daher: Kann man eine Funktion von $\mathbb{R}^n$ nach $\mathbb{R}$ aus Linienintegralen über die Funktion berechnen? Diese pure mathematische Fragestellung wurde weitgehend schon 1905 von Radon bei der Untersuchung der später nach ihm benannten Radon–Transformation beantwortet, der sogar eine Inversionsformel angeben konnte. Leider kann man zeigen, dass diese Inversionsformel nicht praktikabel ist (ihre direkte Implementation ist langsam und liefert große Fehler), siehe hierzu die Diskussionen in Natterer [2001] und Natterer and Wübbeling [2001].
Sehr erfolgreich war dagegen eine viel einfachere Vorgehensweise: Man teilt den

gesamten Raum in Würfel (Voxel) auf und nimmt an, dass g auf jedem Voxel konstant ist. Man macht sich schnell klar (am einfachsten in 2D), dass die Werte in jedem Voxel dann Lösung eines linearen Gleichungssystems sind, das nur noch invertiert werden muss. Effiziente Verfahren zur Lösung dieses Gleichungssystems (das ca. 512^3 Unbekannte hat) bilden heute den Kern der meisten Computertomographie–Geräte. Eine genauere Diskussion finden Sie zuhauf in der Literatur, z.B. in Natterer and Wübbeling [2001].

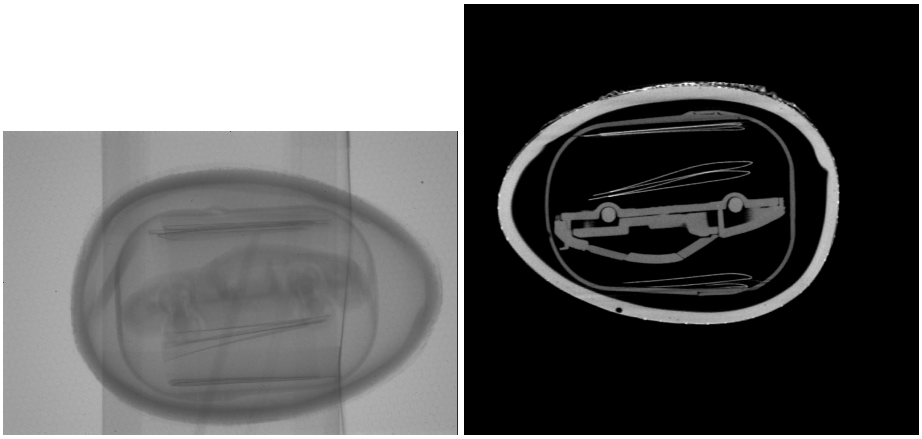

Abbildung 1.1: Röntgenbild/Tomographie eines Überraschungseis. Nur in der Tomographie sind Details erkennbar.

Klick für Bild autoxray
Klick für Bild autotomo

Beispiel 1.2 *Berechnung der Ableitung einer Funktion*
Die Funktion f sei auf dem Intervall I differenzierbar. Ihre Ableitung an der Stelle $x \in I$ ist, wenn sie existiert, definiert als

$$f'(x) = \lim_{h \mapsto 0,\, h \neq 0} \frac{f(x+h) - f(x)}{h}.$$

Diese Definition ist für die Praxis, in der die Ableitung etwa als Geschwindigkeit als Ableitung der zurückgelegten Wegstrecke auftritt, nutzlos, wenn nur diskrete (endlich viele) Funktionsauswertungen vorliegen. Es liegt in diesem Fall nahe, die Ableitung durch die Approximation

$$f'(x) \sim \frac{f(x+h) - f(x)}{h}$$

für ein kleines h zu ersetzen (Modellvereinfachung, Diskretisierung). Der Fehler dieses Modells kann für zweimal stetig differenzierbare Funktionen einfach angegeben

werden. Mit der Taylorentwicklung und dem Lagrange-Restglied gilt

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(\xi)$$

mit einem ξ zwischen x und $x+h$. Der maximale Fehler kann also abgeschätzt werden durch

$$\left| f'(x) - \frac{f(x+h) - f(x)}{h} \right| \leq \frac{|h|}{2} ||f''||_\infty \tag{1.1}$$

Entsprechend zeigt man für viermal stetig differenzierbare Funktionen eine Approximationsformel für die zweite Ableitung:

$$\left| f''(x) - \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \right| \leq \frac{h^2}{12} ||f''''||_\infty \tag{1.2}$$

Hierbei steht natürlich jeweils die Unendlichnorm für das Betragsmaximum auf I. Nach dieser Analyse scheint klar: Je kleiner das h, desto besser das Ergebnis. Dies berücksichtigt aber natürlich die Messfehler nicht. Ist h sehr klein, so führt offensichtlich schon ein kleiner Fehler im Zähler zu riesigen Fehlern. Ist h zu groß, ist der Modellfehler, den wir angegeben haben, zu groß. Diesen Zusammenhang werden wir im nächsten Kapitel genauer untersuchen.

Beispiel 1.3 *Wärmeleitung in einem isolierten Stab*
Als Beispiel für ein komplexeres Anwendungsproblem betrachten wir einen wärmeisolierten Stab, der an beiden Enden auf eine feste Temperatur gebracht wird. Der Stab befinde sich im Intervall $[0, \pi]$ auf der x-Achse und sei homogen. Die Anfangstemperatur zum Zeitpunkt $t = 0$ sei bekannt. Es bezeichne $T(x,t)$ die Temperatur zum Zeitpunkt t an der Stelle x, $t \geq 0$, $x \in [0, \pi]$. Mögliche Fragestellungen:

1. *Bestimme den Temperaturverlauf T unter Berücksichtigung einer externen Wärmequelle $q(x,t)$.*
2. *Nach längerer Zeit stellt sich ein fester Endzustand $T_0(x)$ ein. Bestimme T_0.*

Zunächst benötigen wir eine Mathematisierung (Modellierung). Unter Vernachlässigung vieler physikalischer und mathematischer Gesichtspunkte und aller Konstanten können wir diese leicht motivieren. Sei dazu $[a,b]$ ein Teilintervall von $[0,\pi]$. Sei weiter

$$Q(t) = \int_a^b T(x,t) dx$$

die Wärmeenergie im Intervall $[a,b]$ zum Zeitpunkt t. Es ist anschaulich, dass die Wärmeenergie, die zu einem Zeitpunkt t durch einen Punkt x läuft, proportional zur

Ableitung von T nach x ist: Ist die Ableitung 0, so ändert sich nichts, und es wird auch keine Wärme verschoben. Ist die Ableitung groß, hat man einen großen Temperaturunterschied links und rechts des Punkts, und Wärmeenergie fließt durch diesen Punkt in einer Richtung, die vom Vorzeichen des Unterschieds abhängt (Fouriersches Gesetz).
Da sich Q nur durch Zufluss oder Abfluss von Energie am linken oder rechten Rand oder durch externe Wärmezufuhr ändert, gilt

$$Q(t_2) - Q(t_1) = \int_{t_1}^{t_2} T_x(b,\tau) - T_x(a,\tau) d\tau + \int_{t_1}^{t_2} \int_a^b q(x,t) \mathrm{d}x\mathrm{d}t$$

wobei T_x für die partielle Ableitung von T nach x steht und $q(x,t)$ für die Stärke einer externen Wärmequelle zum Zeitpunkt t an der Stelle x. Durch Differentiation nach t_2 und Einsetzen der Definition von Q erhalten wir unter der Voraussetzung, dass alle Größen ausreichend glatt sind

$$\begin{aligned}
\int_a^b T_t(x,t) \mathrm{d}x &= Q'(t) \\
&= T_x(b,t) - T_x(a,t) + \int_a^b q(x,t) \mathrm{d}x \\
&= \int_a^b T_{xx}(x,t) + q(x,t) \mathrm{d}x
\end{aligned}$$

oder

$$\int_a^b T_t(x,t) - T_{xx}(x,t) - q(x,t) \mathrm{d}x = 0.$$

Dies muss nun aber für jeden Zeitpunkt t und für jedes Intervall $[a,b]$ in $[-\pi,\pi]$ gelten. Das ist nur möglich, falls der Integrand komplett verschwindet, also

$$T_t = T_{xx} + q. \tag{1.3}$$

Eine genauere Herleitung bekommen Sie zum Beispiel in der Vorlesung Modellierung.
Wir haben damit den kompletten physikalischen Vorgang in einer mathematischen Beschreibung verstecken können, in einer Gleichung, die für alle x und t erfüllt sein muss und die die Ableitungen der gesuchten Funktion T enthält (partielle Differentialgleichung, Wärmeleitungsgleichung). Streng analytisch kann man nun zeigen: Die Wärmeleitungsgleichung mit bekanntem Temperaturverlauf $T(x,0) = t_0(x)$ und festen Temperaturen am Rand ($T(0,t) = C_1$, $T(\pi,t) = C_2$) hat eine eindeutige Lösung (mit Bedingungen an q).

Sätze dieser Art sind Inhalt der Vorlesungen partielle Differentialgleichungen und Modellierung.
Für den Endzustand $T_0(x)$ gilt, dass der Temperaturverlauf von der Zeit nicht mehr abhängt, also $(T_0)_t = 0$, er erfüllt damit

$$-(T_0)'' = q \tag{1.4}$$

und die Randbedingung (stationäre Wärmeleitungsgleichung).
Für sehr einfache Funktionen q lässt sich der Endzustand T_0 direkt angeben. Wir setzen der Einfachheit halber $C_1 = C_2 = 0$, am linken und rechten Ende des Stabes wird also auf 0 Grad gekühlt.
Eine in der Physik beliebte Methode, Aufgaben dieser Art zu lösen, bestimmt zunächst einmal die Eigenvektoren (Eigenfunktionen) der Abbildung auf der linken Seite der Differentialgleichung, also der zweiten Ableitung. Wir suchen also nicht–verschwindende Funktionen u_k mit $u_k(0) = u_k(\pi) = 0$ und $(u_k)_{xx} = \lambda_k u_k$. Man zeigt leicht, dass

$$u_k(x) = \sin kx,\ \lambda_k = -k^2$$

für $k \in \mathbb{N}$ dies leisten. Sei nun

$$T_0(x) = \sum_k a_k u_k(x)$$

eine Lösung der Differentialgleichung. Dann gilt

$$q(x) = -(T_0)_{xx}(x) = -\sum_k a_k \lambda_k u_k(x) = \sum_k a_k k^2 \sin kx.$$

Umgekehrt gilt: Hat q eine solche Reihenentwicklung, so ist $\sum_k a_k u_k(x)$ eine (die) Lösung der stationären Wärmeleitungsgleichung. Die Koeffizienten lassen sich mit Hilfe der Fouriertransformation berechnen.
Damit ist das Problem mathematisch eigentlich komplett gelöst: Wir haben gezeigt, dass es eine eindeutige Lösung gibt, und können diese sogar aus der Fourierreihendarstellung von q direkt berechnen. Sei etwa q die charakteristische Funktion des Intervalls $[\pi/2 - \epsilon, \pi/2 + \epsilon]$ für ein $1 > \epsilon > 0$. Dann gilt

$$q(x) = \sum_k A_k \sin(kx),\ A_k = \frac{2}{\pi} \int_{\pi/2-\epsilon}^{\pi/2+\epsilon} \sin(ky) dy = \frac{2}{k\pi} \cos(ky) \Big|_{\pi/2-\epsilon}^{\pi/2+\epsilon}.$$

und die Lösung des stationären Wärmeleitungsproblems ist

$$T_0(x) = \sum_{k=1}^{\infty} \frac{A_k}{k^2} \sin(kx).$$

Wir können die Lösung also exakt angeben. Dies geht aber offensichtlich nur, weil wir die Wärmequelle unrealistisch vereinfacht haben. Möglicherweise ist diese nur gemessen und besitzt keine geschlossene Darstellung - in diesem Fall können wir auch die Fourierreihe nicht berechnen und die analytische Lösung wird wertlos.
Bemerkung: Die so erzielte Lösung ist zwar physikalisch absolut sinnvoll, aber nicht differenzierbar und damit keine Lösung der Differentialgleichung. Dies zeigt, dass unsere mathematische Modellierung nicht vollständig ist.
Es gibt aber eine sehr einfache numerische Lösung für unser Problem durch Diskretisierung. Hierzu verteilen wir zunächst $N+1$ Gitterpunkte x_k gleichmäßig im Intervall $[0,\pi]$, also $x_k = kh$, $h = \pi/N$, $k = 0 \dots N$. Wir beschränken uns darauf, Näherungen u_k für $T_0(x_k)$ zu bestimmen. Mit 1.4 gilt an jedem Gitterpunkt

$$-T_0''(x_k) = q(x_k).$$

Wir approximieren die Differentialgleichung mit 1.2, also

$$-u_{k-1} + 2u_k - u_{k+1} = h^2 q(x_k),\ k = 1 \dots N-1.$$

Zusätzlich wissen wir wegen der Randbedingung $u_0 = T_0(0) = 0$ und $u_N = T_0(\pi) = 0$. Insgesamt erhalten wir damit $N-1$ lineare Gleichungen für die $N-1$ Unbekannten u_1 bis u_{N-1}:

$$\begin{aligned}
-0 + 2u_1 - u_2 &= h^2 q(x_1) \\
-u_1 + 2u_2 - u_3 &= h^2 q(x_2) \\
-u_2 + 2u_3 - u_4 &= h^2 q(x_3) \\
&\vdots \\
-u_{N-3} + 2u_{N-2} - u_{N-1} &= h^2 q(x_{N-2}) \\
-u_{N-2} + 2u_{N-1} - 0 &= h^2 q(x_{N-1})
\end{aligned} \tag{1.5}$$

oder in Matrixschreibweise

$$\frac{1}{h^2}\begin{pmatrix}
2 & -1 & & & & \\
-1 & 2 & -1 & & & \\
 & -1 & 2 & -1 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -1 & 2 & -1 \\
 & & & & -1 & 2
\end{pmatrix}
\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix}
=
\begin{pmatrix} q(x_1) \\ q(x_2) \\ q(x_3) \\ \vdots \\ q(x_{N-2}) \\ q(x_{N-1}) \end{pmatrix} \tag{1.6}$$

Zur numerischen Lösung müssen wir also nur die Matrix invertieren und das Gleichungssystem lösen. Dies ist in Matlab schnell getan, Abbildung 1 zeigt den Vergleich zweier Lösungen, die jeweils mit der analytischen und diskreten Methode erzielt wurden. Bei großem N (hoher Auflösung) sind die Kurven praktisch gleich.

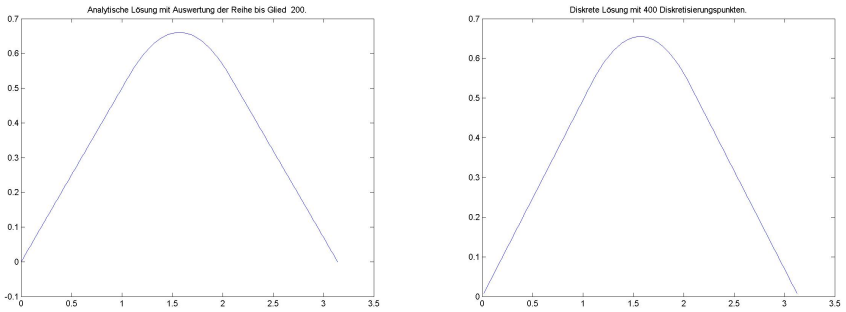

Abbildung 1.2: Analytische/Diskrete Lösung der stationären Wärmeleitungsgleichung

Klick für Bild heatanalytic
Klick für Matlab Figure heatanalytic
Klick für Bild heatdiscrete
Klick für Matlab Figure heatdiscrete

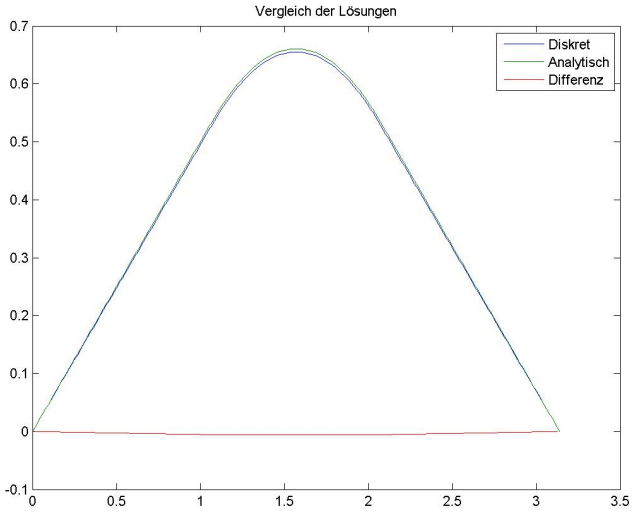

Abbildung 1.3: Vergleich der diskreten/analytischen Lösung der stationären Wärmeleitungsgleichung

Klick für Bild heatcompare
Klick für Matlab Figure heatcompare

```
function [x,y] = analytisch( N, M )
%Analytische Loesung der stationaeren Waermeleitungsgleichung
%fuer einen an den Raendern gekuehlten, isolierten Stab, der in
%der Mitte erwaermt wird
%N – Anzahl der Auswertungspunkte, M – Reihen–Auswertungsgrenze
```

Listing 1.1: Analytische Lösung der stationären Wärmeleitungsgleichung (Waermeleitung/analytisch.m)

Klicken für den Quellcode von Waermeleitung/analytisch.m

```
function [x,y] = diskret( N )
%Diskrete Loesung der stationaeren Waermeleitungsgleichung fuer
%einen an den Raendern gekuehlten, isolierten Stab, der in der
%Mitte erwaermt wird. N – Anzahl der Auswertungspunkte

h=pi/N;
```

Listing 1.2: Diskrete Lösung der stationären Wärmeleitungsgleichung (Waermeleitung/diskret.m)

Klicken für den Quellcode von Waermeleitung/diskret.m

```
function doit1d
%Treiber fuer analytisch, diskret
global epsilon;
epsilon=0.5;
N=200;
close all;
```

Listing 1.3: Rahmen zur stationären Wärmeleitungsgleichung (Waermeleitung/-doit1d.m)

Klicken für den Quellcode von Waermeleitung/doit1d.m

Wir halten also fest: Zur Diskretisierung praktischer Probleme müssen am Ende meist (große) lineare Gleichungssysteme gelöst werden. Dies möglichst genau und effizient zu tun, wird den Großteil dieser Vorlesung einnehmen.
Bemerkung: *Die Cramersche Regel wäre aus mathematischer Sicht hierzu bereits absolut ausreichend, leider ist sie weder effizient noch liefert ihre direkte Implementation genaue Werte.*

Warnung: *Dies ist eine reine Motivation. Das analytische Modell durch ein diskretes zu ersetzen, scheint hier eine gute Idee zu sein. Tatsächlich kann aber natürlich erst eine genaue mathematische Analyse zeigen, ob das Ergebnis brauchbar ist bzw. mit welchem Fehler die diskrete Lösung behaftet ist. Zur Warnung schauen wir uns daher auch noch die zeitabhängige Wärmeleitungsgleichung an. Wir diskretisieren die Zeit an den Zeitpunkten* $t_k = k$*dt,* $k \in \mathbb{N}_0$*. Sei* $u(t_k, x)$ *die gesuchte Näherung für die Temperatur am Punkt* x *zum Zeitpunkt* t_k*. Mit Hilfe der Formel für die Diskretisierung der ersten Ableitung erhalten wir also*

$$u(t_{k+1}, x) = u(t_k + dt, x) = u(t_k, x) + dt(u_{xx} + q).$$

Wir können also eine Approximation für die Temperatur zum Zeitpunkt t_{k+1} *angeben, wenn wir die Temperatur zum Zeitpunkt* t_k *kennen. Für* $t = 0$ *ist die Temperatur bekannt (hier konstant o), zur Berechnung der zweiten Ableitung verwenden wir wieder unsere Formel für die Diskretisierung der zweiten Ableitung, und wir erhalten sofort die im Programm implementierte Formel. Um unser Programm zu testen, lassen wir es für einige Zeit laufen und vergleichen den Endzustand mit dem vorher berechneten aus der stationären Wärmeleitungsgleichung.*

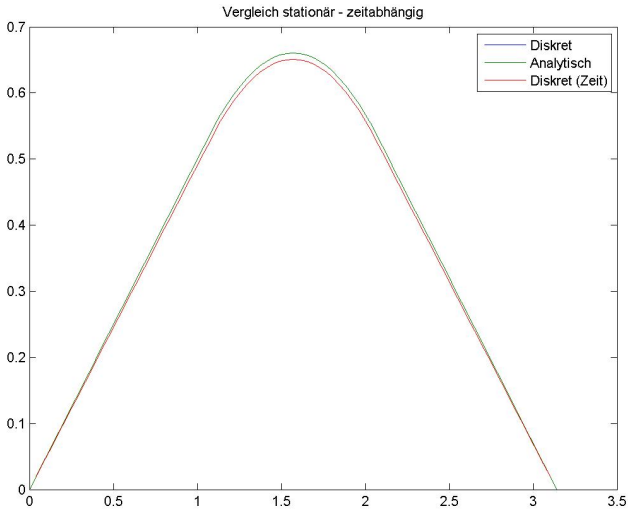

Abbildung 1.4: Vergleich der stationären Lösung mit der zeitabhängigen Lösung

Klick für Bild heattime1
Klick für Matlab Figure heattime1

```
function [x,y] = diskrettime( NT,N,T )
%Loesung der Waermeleitungsgleichung in der Zeit,
%diskret in Ort und Raum.
%NT - Zeitschritte pro Sekunde, N - Raumschritte,
%T - Endzeit. Achtung: Stabilitaetsbedingung beachten!
```

Listing 1.4: Lösung der zeitabhängigen Wärmeleitungsgleichung (Waermeleitung/-diskrettime.m)

Klicken für den Quellcode von Waermeleitung/diskrettime.m

Alles ist so, wie wir es erwarten: Der Endzustand liegt nah an der Lösung der stationären Gleichung. Wir wollen nun etwas genauer werden und erhöhen die Diskretisierung auf der Raumachse wenig, statt 80 wählen wir nun 100 Diskretisierungspunkte und lassen unsere Simulation wieder laufen.

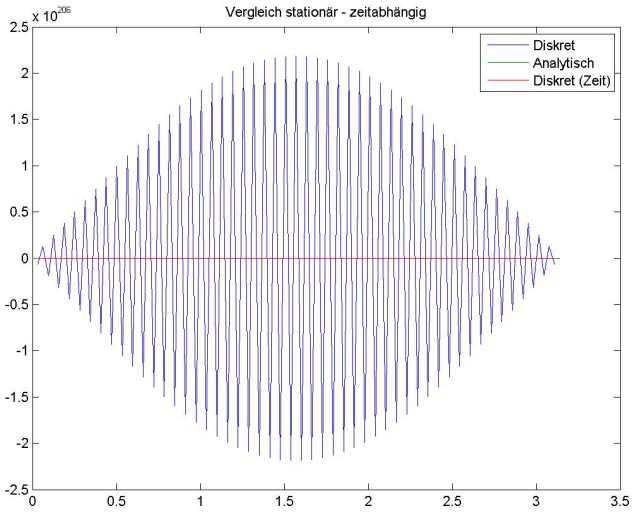

Abbildung 1.5: Vergleich der stationären Lösung mit der zeitabhängigen Lösung, instabil

Klick für Bild heattime2
Klick für Matlab Figure heattime2

```
function [ output_args ] = doittime(  )
%Treiber fuer diskrettime
global epsilon;
epsilon=0.5;
close all;
[x1,y1]=diskrettime(1300,80,10);
```

Listing 1.5: Lösung der zeitabhängigen Wärmeleitungsgleichung mit instabilen Parametern (Waermeleitung/doittime.m)

Klicken für den Quellcode von Waermeleitung/doittime.m

Das ist definitiv nicht, was wir erwarten. Eine Verbesserung der Approximation führt zu einem völlig chaotischen (in der Numerik: instabilen) Verhalten. Dies zeigt deutlich, dass blinde Diskretisierung ohne zusätzliche mathematische Analyse zu unsinnigen Ergebnissen führen kann.
Die genaue Erklärung für **dieses** *Phänomen erhalten Sie in den Vorlesungen Numerische Analysis und Numerik Partieller Differentialgleichungen.*

Kapitel 2

Grundlagen der LA und der Fehlerrechnung

2.1 Lineare Algebra

Wir erinnern zunächst an einige Grundbegriffe der linearen Algebra. Wir beschränken uns grundsätzlich auf die Betrachtung von Vektorräumen über $K = \mathbb{R}$ oder $K = \mathbb{C}$. Seien also im Folgenden immer U und V Vektorräume über K.

2.1.1 Normierte Vektorräume

Grundlegend für alle numerischen Überlegungen ist der Begriff der Norm, denn nur so lassen sich Fehler messen.

Definition 2.1 *(normierte Vektorräume)*
Sei V ein Vektorraum. $||\cdot|| : V \mapsto \mathbb{R}^{\geq 0}$ heißt Norm, falls

1) $||\alpha x|| = |\alpha|||x|| \, \forall \alpha \in K, \, x \in V$.

2) $||x|| = 0 \Leftrightarrow x = 0$.

3) $||x + y|| \leq ||x|| + ||y|| \, \forall x, y \in V$.

$(V, ||\cdot||)$ heißt normierter Vektorraum.

Beispiel 2.2 *Sei $V = \mathbb{R}^n$, $p \in [1, \infty]$, $v = (v_1, \ldots, v_n) \in V$.*

$$||v||_p := \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} (p < \infty), \; ||v||_\infty = \max_i |v_i|$$

heißt p–Norm (und ist eine Norm).

Beispiel 2.3 *Sei $V = C^0(I)$ der Raum der stetigen Funktionen auf einer kompakten Teilmenge $I \subset \mathbb{R}^n$, $f \in V$.*

$$||f||_p = \left(\int_I |f(x)|^p dx \right)^{1/p} (p < \infty),\ ||f||_\infty = \sup_{x \in I} |f(x)|$$

heißt p–Norm (und ist eine Norm).

Definition 2.4 *(Banachraum) Sei $(V, ||\cdot||)$ normierter Vektorraum. V heißt vollständig oder Banachraum, falls jede Cauchyfolge in V einen Grenzwert in V besitzt (bzgl. $||\cdot||$).*

Beispiel 2.5
$(C^0(I), ||\cdot||_2)$ ist nicht vollständig.
$(C^0(I), ||\cdot||_\infty)$ ist vollständig.

Definition 2.6 *(Vektorräume mit Skalarprodukt, euklidische Vektorräume)*
$(\cdot, \cdot) : V \times V \mapsto K$ heißt Skalarprodukt, falls

1) $(v, v) \geq 0$ *und* $(v, v) = 0 \Leftrightarrow v = 0\ \forall v \in V$.

2) $(u, v) = \overline{(v, u)}\ \forall u, v \in V$.

3) $(\cdot, v)$ *ist linear für alle festen* $v \in V$.

Üblicherweise wird auf euklidischen Räumen die induzierte Norm

$$||v|| = (v, v)^{1/2},\ v \in V$$

benutzt. V heißt dann Prä–Hilbertraum. Ist V mit dieser Norm vollständig, so heißt V Hilbertraum.

Beispiel 2.7

1) Sei $V = \mathbb{C}^n$. Dann ist

$$(u, v) = u^t \overline{v},\ u, v \in V$$

ein Skalarprodukt.

2) Sei $V = C^0(I)$. Dann ist

$$(f, g) = \int_I f(x) \overline{g(x)} dx,\ f, g \in V$$

ein Skalarprodukt.

Wir werden beide stillschweigend als Standard–Skalarprodukte auf den jeweiligen Räumen verwenden. Die induzierte Norm ist jeweils $||\cdot||_2$.

Satz 2.8 *(Cauchy–Schwarz)*
Sei V *ein Vektorraum mit Skalarprodukt. Dann gilt*

$$|(u,v)|^2 \leq ||u||^2||v||^2 \, \forall u, v \in V$$

und Gleichheit genau dann, wenn u *und* v *linear abhängig sind.*

Beweis: Falls $v = 0$, so ist der Satz richtig. Sei also $v \neq 0$. Es gilt

$$0 \leq || \, ||v||^2 u - (u,v)v||^2 = ||v||^4(u,u) - 2|(u,v)|^2||v||^2 + |(u,v)|^2(v,v)$$

und damit

$$|(u,v)|^2 \leq ||u||^2||v||^2$$

und Gleichheit genau dann, wenn $u = \lambda v$. □

Vorlesungsnotiz: Beim Auflösen muss der Skalar aus dem zweiten Argument geholt werden und wird komplex konjugiert. Evtl. $||u+v||^2 = (u+v, u+v) = ||u||^2 + 2\text{Re}(u,v) + ||v||^2$, und hier steht im gemischten Term $(u,v)(v,u)$.

Die wichtigste Folgerung ist

Satz 2.9 *Sei* V *ein Vektorraum mit Skalarprodukt. Dann ist*

$$||v|| = (v,v)^{1/2}, \; v \in V$$

eine Norm.

Beweis:

$$||u+v||^2 = ||u|| + 2\text{Re}(u,v) + ||v||^2 \leq ||u||^2 + 2||u|| \, ||v|| + ||v||^2 = (||u|| + ||v||)^2.$$

□

Satz 2.10 *Sei* V *endlichdimensional und seien* $||\cdot||$ *und* $|||\cdot|||$ *zwei Normen auf* V. *Dann sind* $||\cdot||$ *und* $|||\cdot|||$ *äquivalent, d.h.* $\exists \, C_1, C_2 > 0$*:*

$$C_1|||v||| \leq ||v|| \leq C_2|||v|||.$$

Beweis: Sei $(v_1, \ldots, v_n)$ eine Basis von V, $v \in V$, $v = \sum_k \alpha_k v_k$. Œsei

$$|||v||| = ||| \sum_k \alpha_k v_k ||| = ||(\alpha_1, \ldots, \alpha_n)||_\infty = \max_k |\alpha_k|$$

(die Äquivalenz ist transitiv). Sei

$$C_2 = n \max_k ||v_k||.$$

Dann gilt

$$||v|| \leq \sum_k |\alpha_k| \, ||v_k|| \leq n \max_k |\alpha_k| \max_k ||v_k|| \leq C_2 ||v||_\infty.$$

Insbesondere gilt:

$$v_k \to_{||\cdot||_\infty} v \Longrightarrow v_k \to_{||\cdot||} v.$$

Sei

$$C_1 = \inf_{||v||_\infty = 1} ||v||.$$

Angenommen, $C_1 = 0$. Dann ex. Folge (v_k) mit

$$||v_k||_\infty = 1, \; ||v_k|| \leq 1/k \Rightarrow v_k \mapsto_{||\cdot||} 0.$$

Die Einheitskugel bezüglich der Unendlichnorm ist kompakt, besitzt also eine gegen ein v aus V konvergente Teilfolge. Nach der Vorbemerkung ist $v = 0$ und damit $|||v||| = 0$, aber $|||v_n||| = 1$. Das ist ein Widerspruch, also galt $C_1 > 0$, und C_1 leistet das Verlangte. □

2.1.2 Lineare Operatoren

Wir werden in dieser Vorlesung im wesentlichen Matrizen als Spezialfall linearer Operatoren untersuchen.

Definition 2.11 *(lineare Operatoren) Seien U, V Vektorräume. $T : U \mapsto V$ heißt linear genau dann, wenn*

$$T(\alpha x + y) = \alpha T x + T y, \; \forall \alpha \in K, \; x, y \in U.$$

Sind U und V endlichdimensional, so kann T durch eine Matrix bezüglich vorgegebener Basen dargestellt werden. Falls $U = V$ und T in zwei verschiedenen Basen durch die Matrizen A und B dargestellt wird, so heißen A und B ähnlich, und es gibt eine Matrix X mit

$$A = XBX^{-1}.$$

Die Menge aller linearen Operatoren $L(U,V)$ bildet auf natürliche Weise selbst wieder einen Vektorraum.

Satz 2.12 *(Stetigkeit beschränkter linearer Operatoren) Seien* $(U, ||\cdot||_U)$ *und* $(V, ||\cdot||_V)$ *normierte Vektorräume. Sei* $T \in L(U,V)$. T *ist stetig genau dann, wenn*

$$||T|| := \sup_{u\in U, u\neq 0} \frac{||Tu||_V}{||u||_U} = \sup_{u\in U, u\neq 0} ||T\frac{u}{||u||_U}||_V = \sup_{u\in U, ||u||_U=1} ||Tu||_V$$

beschränkt ist. Es gilt

$$||Tu||_V \leq ||T||\,||u||_U \,\forall u \in U$$

und

$$||T_1T_2|| \leq ||T_1||\,||T_2|| \,\forall T_2 \in L(U,V),\, T_1 \in L(V,W).$$

Beweis:

$$||Tu||_V = ||T\frac{u}{||u||_U}||_V||u||_U \leq ||T||\,||u||_U \,(u \neq 0).$$

$$||T_1T_2|| = \sup_{||u||_U=1} ||T_1T_2u|| \leq \sup_{||u||_U=1} ||T_1||\,||T_2u||_V = ||T_1||\,||T_2||.$$

Wegen der Linearität reicht es, die Stetigkeit in 0 nachzuweisen. Sei u_k eine Nullfolge und $||T||$ beschränkt. Dann ist

$$||Tu_k|| \leq ||T||\,||u_k||$$

und damit ebenfalls Nullfolge. Sei nun $||T||$ unbeschränkt. Dann gibt es eine Folge u_k von Vektoren mit Norm 1 in U, so dass $||Tu_k||_V \geq k$. Dann ist $w_k = u_k/||Tu_k||_V$ eine Nullfolge, aber $||Tw_k||_V = 1$, also ist Tw_k keine Nullfolge und T nicht stetig. □

Satz 2.13 *(Norm von Operatoren) Seien* $(U, ||\cdot||_U)$ *und* $(V, ||\cdot||_V)$ *normierte Vektorräume,* $B(U,V)$ *der Vektorraum der stetigen linearen Operatoren von* U *nach* V*, also*

$$B(U,V) = \{T \in L(U,V) : ||T|| < \infty\}.$$

Dann ist $||T||$ *Norm auf* $B(U,V)$.

$||\cdot||$ heißt Operatornorm und ist die Standardnorm auf $B(U,V)$.

Korollar 2.14 *Sei* A_k *eine Folge von Matrizen.* A_k *konvergiert gegen* A *in einer Norm* $||\cdot||$ *genau dann, wenn alle Matrixelemente gegeneinander konvergieren.*

Beweis: Äquivalenz zur Unendlichnorm der Koeffizienten. □

Satz 2.15 *Sei $(U, ||\cdot||_U)$ endlichdimensional. Dann ist jede Abbildung $T \in L(U,V)$ stetig.*

Beweis: Sei $u = \sum_{k=0}^{n} \alpha_k u_k$, u_k Basis von U. Es gilt

$$||Tu||_V \leq \sum_{k=0}^{n} |\alpha_k| \, ||Tu_k||_V \leq ||u||_\infty n \max_k ||Tu_k||_V \leq (nC_2 \max_k ||Tu_k||_V)||u||_U$$

wobei C_2 der Äquivalenzfaktor zwischen $||\cdot||_\infty$ und $||\cdot||_U$ und $||\cdot||_\infty$ wie in 2.10 ist. □

Definition 2.16 *(Eigenwerte und Eigenvektoren)*
Sei $T \in L(U,U)$. $v \in U$, $v \neq 0$. v heißt Eigenvektor zum Eigenwert $\lambda \in \mathbb{C}$, falls $Av = \lambda v$. T heißt diagonalisierbar, falls U eine Basis aus Eigenvektoren v_k von T besitzt. Falls U endlichdimensional ist, so wird T in den v_k durch eine Diagonalmatrix D dargestellt, auf deren Hauptdiagonale die Eigenwerte stehen. Es gilt also

$$D = W^{-1}TW, \; W = (v_1 v_2 \cdots v_n).$$

Definition 2.17 *(Adjungierte Abbildung)*
Seien $(U, (\cdot,\cdot)_U)$ und $(V, (\cdot,\cdot)_V)$ Vektorräume mit Skalarprodukt. Sei $T \in L(U,V)$, $T^* \in L(V,U)$. Falls

$$(Tu, v)_V = (u, T^*v)_U \forall u \in U, \; v \in V,$$

so heißt T^* die zu T adjungierte Abbildung.
Falls $U = V$ und $T = T^*$, so heißt T selbstadjungiert.

Beispiel 2.18 *Sei $U = \mathbb{C}^n$, $V = \mathbb{C}^m$, $T \in L(U,V)$ (also T $(n \times m)$–Matrix, wobei wir immer unzulässigerweise die Matrizen mit den Abbildungen identifizieren, die sie darstellen). Dann gilt für $u \in U$, $v \in V$*

$$(Tu, v) = u^t T^t \overline{v} = u^t \overline{(\overline{T^t} v)} = (u, \overline{T^t} v)$$

und damit $T^* = \overline{T^t}$, über $\mathbb{R}$ natürlich $T^* = T^t$. Matrizen mit der Eigenschaft $T = T^* = \overline{T^t}$ heißen hermitesch, reelle Matrizen mit der Eigenschaft $T = T^* = T^t$ heißen symmetrisch.

Satz 2.19 *(Rechenregeln für adjungierte Operatoren)*

1. $(T_1 T_2)^* = T_2^* T_1^*$.
2. $(T^*)^* = T$.

3. TT^* und T^*T sind selbstadjungiert.

Satz 2.20 *Selbstadjungierte Operatoren haben reelle Eigenwerte. Eigenvektoren zu unterschiedlichen Eigenwerten stehen senkrecht aufeinander.*

Beweis: Sei T selbstadjungiert. Sei $Tx = \lambda x$, $x \neq 0$. Dann gilt

$$\lambda(x,x) = (\lambda x, x) = (Tx, x) = (x, Tx) = (x, \lambda x) = \overline{\lambda}(x,x)$$

und wegen $(x,x) \neq 0$ gilt $\lambda = \overline{\lambda}$.
Sei $Tx = \lambda_1 x$, $Ty = \lambda_2 y$, $\lambda_1 \neq \lambda_2$, $x \neq 0$, $y \neq 0$. Dann gilt

$$\lambda_1(x,y) = (Tx, y) = (x, Ty) = \overline{\lambda_2}(x,y) = \lambda_2(x,y)$$

und damit wegen $\lambda_1 \neq \lambda_2$: $(x,y) = 0$. □

Definition 2.21 *(Positiv definite Operatoren)*
Sei U Vektorraum mit Skalarprodukt, $T \in L(U,U)$. T heißt (symmetrisch) positiv definit, wenn T selbstadjungiert ist und

$$(Tu, u) > 0 \, \forall u \in U, \, u \neq 0.$$

Gilt nur $\geq$, so heißt T positiv semidefinit.

Satz 2.22 *Sei U Vektorraum mit Skalarprodukt, $T \in L(U,U)$ symmetrisch positiv definit. Dann ist*

$$(u,v)_T := (Tu, v), \, u \in U, \, v \in U$$

ein Skalarprodukt auf U.

Satz 2.23 *Sei $T \in L(U,V)$. T^*T ist positiv semidefinit. Falls T injektiv ist, so ist T positiv definit.*

Beweis: T^*T ist selbstadjungiert, und $(T^*Tx, x) = (Tx, Tx) \geq 0$. □

Den Satz über die Jordan–Normalform kennen Sie aus der Linearen Algebra I. Bitte machen Sie sich klar, dass Ihre Formulierung der folgenden entspricht.

Satz 2.24 *(Jordan–Normalform)*
Sei A eine $(n \times n)$–Matrix. v heißt Hauptvektor k. Stufe zum Eigenwert λ von A, falls

$$(A - \lambda I)^k v = 0, (A - \lambda I)^{k-1} v \neq 0.$$

Hauptvektoren erster Stufe sind Eigenvektoren.

1. *Jede Matrix besitzt eine Basis aus Hauptvektoren v_j.*

2. *Sei J die Darstellung von A in dieser Basis, also*

$$J = B^{-1}AB,\ B = (v_1 v_2 \cdots v_n).$$

Dann ist J (fast) eine Diagonalmatrix, möglicherweise mit einigen Einsen oberhalb der Hauptdiagonalen, auf der die Eigenwerte von A stehen.

Satz 2.25 *Sei A hermitesche $(n \times n)$–Matrix. Dann ist A diagonalisierbar. U besitzt eine Orthonormalbasis aus Eigenvektoren von A.*

Beweis: Zu zeigen ist: Alle Hauptvektoren sind Eigenvektoren, also Hauptvektoren erster Stufe. Sei $(A - \lambda I)^2 v = 0$. Dann gilt

$$0 = ((A - \lambda I)^2 v, v) = ((A - \lambda I)v, (A - \lambda I)v) = ||(A - \lambda I)v||^2$$

und damit schon $(A - \lambda I)v = 0$, es gibt also keine Hauptvektoren höherer Stufe, und die Jordan–Normalform ist eine Diagonalmatrix. Die Eigenvektoren bilden eine Basis und können orthogonal zueinander gewählt werden nach 2.20. □

Korollar 2.26 *Die Matrix A sei hermitesch. A ist positiv definit (semidefinit) genau dann, wenn alle Eigenwerte von A positiv (nichtnegativ) sind.*

Wir zitieren noch ein Kriterium zur Definitheit:

Satz 2.27 *Eine hermitesche Matrix ist genau dann positiv definit (semidefinit), wenn alle ihre Hauptminoren positiv (nichtnegativ) sind.*

Mit diesen Vorbemerkungen können wir nun leicht die 2–Norm einer Matrix berechnen. Sei $A \in \mathbb{C}^{m \times n}$ und $B = A^* A$.

Definition 2.28 *Sei $A \in \mathbb{C}^{n \times n}$. Dann heißt*

$$\rho(A) = \max\{|\lambda_k| :\ \lambda_k \textit{ Eigenwert von } A\}$$

Spektralradius von A.

Satz 2.29 *Sei $A \in \mathbb{C}^{m \times n}$. Dann gilt*

$$||A||_2 = \rho(A^t A)^{1/2}.$$

Beweis: $B = A^tA$ ist symmetrisch positiv semidefinit, also besitzt $\mathbb{C}^n$ eine Orthonormalbasis aus Eigenvektoren v_k zu Eigenwerten λ_k von B. Sei $v = \sum_k \mu_k v_k \in \mathbb{C}^m$, also $||v||^2 = \sum_k |\mu_k|^2$. Dann gilt

$$\begin{aligned} ||Av||_2^2 = (Av, Av) &= (A^*Av, v) \\ &= (\sum_k \mu_k \lambda_k v_k, \sum_j \mu_j v_j) \\ &= \sum_k \lambda_k |\mu_k|^2 \\ &\leq \rho(B) \sum_k |\mu_k|^2 \\ &= \rho(B)||v||_2^2 \end{aligned}$$

und "=", falls v als Eigenvektor zum maximalen Eigenwert von B gewählt wird. Also gilt

$$||A||_2 = \sup_{v \neq 0} \frac{||Av||_2}{||v||_2} = \rho(A^*A)^{1/2}.$$

□

Satz 2.30 *Sei $A \in \mathbb{R}^{m \times n}$. Dann gilt*

$$||A||_\infty = \max_{1 \leq i \leq m} \sum_{k=1}^{n} |A_{ik}|.$$

Beweis: Übungen. □

Zum Abschluss zitieren wir noch einen letzten Satz, der bei der Fehlerrechnung eine große Rolle spielt.

Satz 2.31 *(Neumannsche Reihe)*
Sei $(V, ||\cdot||)$ ein Banachraum, $T : V \mapsto V$ linear mit $||T|| < 1$ (induzierte Norm). Dann ist $(I - T)$ invertierbar, und

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

Beweis: Wegen $||T|| < 1$ sind für jedes $v \in V$ die Partialsummen von $\sum_{k=0}^{\infty} T^k v$ eine Cauchyfolge, also konvergiert die Summe für jedes v gegen ein $Bv \in V$. Es gilt

$$(I - T)Bv = \lim_{n \mapsto \infty} (I - T) \sum_{k=0}^{n} T^k v = \lim_{n \mapsto \infty} (v - T^{n+1} v) = v.$$

□

Korollar 2.32 *Seien V Banachraum, $T \in L(V,V)$ invertierbar, $\Delta T \in L(V,V)$ und T^{-1} sei stetig. Weiter sei $q = ||T^{-1}||\,||\Delta T|| < 1$. Dann ist $(T + \Delta T)$ invertierbar und*

$$||(T + \Delta T)^{-1}|| \leq \frac{||T^{-1}||}{1-q}.$$

Beweis:

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

ist invertierbar nach 2.31.

$$\begin{aligned} ||(T + \Delta T)^{-1}|| &= ||\sum_{k=0}^{\infty}(-T^{-1}\Delta T)^k T^{-1}|| \\ &\leq ||T^{-1}|| \sum_{k=0}^{\infty} q^k \\ &= ||T^{-1}|| \frac{1}{1-q} \end{aligned}$$

□

Dieser Satz lässt sich so interpretieren: Die Matrix T sei invertierbar. Bekannt ist eine Approximation T' mit einem Fehler $||T - T'|| < \epsilon$. Falls ϵ klein genug ist, so ist auch T' invertierbar.

Korollar 2.33 *Die Menge der invertierbaren $(n \times n)$–Matrizen ist offen.*

2.2 Fehler beim numerischen Rechnen

Fehler können beim numerischen Rechnen an mindestens vier Stellen entstehen:

1. Der **Modellierungsfehler** entsteht dadurch, dass wir ein (womöglich vereinfachtes) mathematisches Modell zugrunde legen, das nicht die gesamte Anwendung umsetzt. Beispiel: Im CT–Beispiel haben wir keine Streuung berücksichtigt.

2. Der **Diskretisierungsfehler** entsteht dadurch, dass wir nicht die exakte mathematische Formel implementieren. Beispiel: Approximation des Differentialquotienten durch einen Differenzenquotienten wie in 1.1.

3. Der **Messfehler** bewirkt, dass unsere Eingangsdaten nur eine endliche Genauigkeit haben.

4. Der **Rechenfehler** entsteht durch Rundung bei der Durchführung der Rechnung.

Im Rahmen dieser Vorlesung werden wir uns nur mit den Punkten drei und vier beschäftigen. Der Messfehler dominiert dabei üblicherweise den Rechenfehler.

Definition 2.34 *(absoluter und relativer Fehler)*
Sei $x \in V$, $(V, ||\cdot||)$ normierter Vektorraum, $\tilde{x}$ eine Näherung für x. Dann heißt

$$||\Delta x||,\ \Delta x = (x - \tilde{x})$$

absoluter Fehler *von $\tilde{x}$. Falls $x \neq 0$, so heißt*

$$\frac{||\Delta x||}{||x||}$$

relativer Fehler *von $\tilde{x}$.*

Natürlich hängen alle diese Definitionen von der verwendeten Norm ab. Im Allgemeinen gibt die Anwendung eine Norm vor.
Üblicherweise spielt der absolute Fehler eine untergeordnete Rolle, wir werden immer den relativen Fehler betrachten.
Auf einem Rechner kann immer nur eine Teilmenge M der reellen bzw. komplexen Zahlen darstellen. Üblicherweise wird dabei die Zahlendarstellung nach dem Standard IEEE 754 (1985) benutzt, der auch Regeln für die Rundung, Rechnung und Fehlerbehandlung festsetzt. Selbst neueste Prozessoren setzen den Standard um oder besitzen zumindest einen Schalter, mit dem man IEEE–Kompatibilität erzwingen kann (etwa im Intel–Compiler: “fp-model precise”).

Definition 2.35 *(Gleitkommazahlendarstellung und Runden nach IEEE 754)*
Seien $b >= 2$ (Basis), $p >= 1$ (Mantissenlänge), $r >= 1$ (Exponentlänge) für ein Format fest gewählte ganze Zahlen. Dann ist die Menge M der **Maschinenzahlen** *definiert durch*

$$M := \left\{ \pm \left(\sum_{k=1}^{p} m_k b^{-k} \right) b^{\pm e},\ m_k, e \in \mathbb{Z},\ 0 \leq m_k \leq b-1,\ |e| < b^r \right\}.$$

Die Zahlen $m \in M$ haben die b-adische **normalisierte Darstellung**

$$m = \pm 0.m_1 m_2 m_3 \ldots m_p b^{\pm e}$$

mit $m_1 \neq 0$ *(oder* $m = 0$*).*

$$\text{eps} = \frac{b^{-p+1}}{2}$$

heißt **Maschinengenauigkeit** *(und ist eine Matlab–Funktion).*
Eine Funktion rd $: \mathbb{R} \mapsto M$ *heißt* **Rundungsfunktion,** *falls*

$$|rd(x) - x| = \min_{y \in M} |y - x|.$$

$rd(x)$ ist nicht eindeutig, zu jedem Format gehört also auch immer eine Rundungsfunktion (IEEE definiert round-to-zero, round-to-infinity, ...).

Beispiel 2.36

$b = 10$, $p = 2$, $r = 1$ *(human format).*
$0.12 \cdot 10^{-9}$ *ist Maschinenzahl.*
$-0.99 \cdot 10^{8}$ *ist Maschinenzahl.*
$-0.234 \cdot 10^{10}$ *ist keine Maschinenzahl (Exponent und Mantisse zu lang).*
Üblicherweise schreibt man E10 für 10^{10} *usw.*

$b = 2$, $p = 23$, $r = 7$.
Dies ist der single precision–Standard, er belegt im Rechner $1 + 23 + 1 + 7 = 32$ *Bits. Es gilt* $\text{eps} = 2^{-23} \sim 10^{-7}$.

$b = 2$, $p = 52$, $r = 10$.
Dies ist der double precision–Standard, er belegt im Rechner $1 + 52 + 1 + 10 = 64$ *Bits. Es gilt* $\text{eps} = 2^{-52} \sim 10^{-16}$.

Wir vernachlässigen in unseren Betrachtungen den Einfluss des Exponenten, setzen also $r = \infty$. Falls die Beschränkung etwa in double precision auf 10^{308} als größte Zahl problematisch ist, kann dieses Problem durch Skalierung aller Größen gelöst werden. Nach IEEE–Standard wird ein Programm abgebrochen, wenn der Exponent zu groß (Overflow) oder zu klein (Underflow, zu nah an 0) wird. Dies wird allerdings von aktuellen Compilern im Allgemeinen nicht mehr beachtet.
Mit dieser Einschränkung wird nur die 0 auf 0 gerundet, und es gilt für $x \neq 0$

Satz 2.37 *(Abschätzung des Rundungsfehlers)*

$$\left|\frac{\text{rd}(x) - x}{\text{rd}(x)}\right| \leq \text{eps}$$

und

$$\left|\frac{\text{rd}(x) - x}{x}\right| \leq \text{eps}\,.$$

Beweis: Sei m eine Maschinenzahl in normalisierter Darstellung, $m \neq 0$. Wenn $x \in \mathbb{R}$ zu m gerundet wird, darf sich x in der b–adischen Darstellung maximal um $b/2$ an der $p+1$. Stelle hinter dem Komma von m unterscheiden,also

$$|x - m| \leq b^e b^{-p-1} b/2.$$

Da $m_1 \neq 0$ (m ist normalisiert), gilt

$$|m| \geq b^{-1} b^e$$

und damit

$$\left| \frac{m-x}{m} \right| \leq \frac{b^e b^{-p-1} b/2}{b^{-1} b^e} = \text{eps}\,.$$

Der zweite Teil folgt genauso, mit einer Fallunterscheidung für $m = 0.1b^e$ (Übungen). □

M ist nicht abgeschlossen bezüglich der arithmetischen Operationen. Im human format etwa gilt:

$$0.1 \in M,\ 0.1\,10^{-4} \in M, 0.1 + 0.1\,10^{-4} = 0.1001 \notin M.$$

Wir müssen also nach jeder Operation runden.

Definition 2.38 *(**Maschinenoperationen**)*
Auf $M \times M$ sind die Abbildungen $\oplus$, $\ominus$, $\odot$ und $\oslash$ nach M definiert durch

$$m_1 \oplus m_2 = \text{rd}(m_1 + m_2)$$

usw. Offensichtlich gilt

$$\left| \frac{(m_1 \oplus m_2) - (m_1 + m_2)}{m_1 + m_2} \right| \leq \text{eps}$$

für $m_1 + m_2 \neq 0$ usw.

Bemerkung: M ist nicht assoziativ bezüglich der Maschinenoperationen. Im human format gilt:

$$(10^{-30} \oplus 1) \ominus 1 = 0$$

aber

$$10^{-30} \oplus (1 \ominus 1) = 10^{-30}$$

2.3 Fehlerverstärkung

Uns interessiert besonders, wie stark ein Eingangsfehler (Messfehler oder Rundungsfehler) das Ergebnis beeinflusst. Sei also f eine stetig differenzierbare Funktion auf $I \subset \mathbb{R}$, die im Punkt x ausgewertet werden soll. Statt x sei nur eine Näherung $\tilde{x}$ mit relativem Fehler ϵ bekannt. Wir berechnen den relativen Fehler von $\tilde{y} = f(\tilde{x})$ zu $y = f(x)$. Mit Taylorentwicklung und Lagrange–Restglied oder auch einfacher mit dem Mittelwertsatz gilt

$$f(\tilde{x}) = f(x) + f'(\xi)(\tilde{x} - x)$$

und damit für $x \neq 0$ und $f(x) \neq 0$

$$\left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| \leq \left| \frac{f'(\xi)}{f(x)} \, \frac{\tilde{x} - x}{x} \, x \right| \leq M\epsilon$$

mit

$$M = \max_{\xi \in [\tilde{x}, x]} |f'(\xi)| \left| \frac{x}{f(x)} \right|.$$

Der Fehler in der Ausgangsvariablen wird also höchstens um den Faktor M vergrößert. M heißt **Verstärkungsfaktor** oder **Konditionszahl** und wird, falls der Fehler $|\tilde{x} - x|$ klein ist, häufig durch

$$\tilde{M} = \left| \frac{f'(x)}{f(x)} \, x \right|$$

abgeschätzt. Allgemein gilt für $f : \mathbb{R}^n \mapsto \mathbb{R}$

Satz 2.39 *Sei f auf einer konvexen und kompakten Menge $I \subset \mathbb{R}^n$ stetig differenzierbar. Seien $x, \, \tilde{x} \in I$, $x \neq 0$, $f(x) \neq 0$. Dann gilt*

$$\left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| \leq \sum_{j=1}^{n} \left| \frac{(\partial f / \partial x_j)(\xi)}{f(x)} \, \frac{\tilde{x}_j - x_j}{x_j} \, x_j \right| \leq \sum_{j=1}^{n} M_j \frac{|\tilde{x}_j - x_j|}{|x_j|}$$

mit

$$M_j = \max_{\xi \in [\tilde{x}, x]} \left| \frac{\partial f}{\partial x_j}(\xi) \right| \cdot \left| \frac{x_j}{f(x)} \right|.$$

Beweis: Betrachte die Funktion $g(t) = f(x + t(\tilde{x} - x))$ und wende die Vorbemerkung an. □

Beispiel 2.40

1. $f(x,y) = x + y$

$$M_x = \left| \frac{x}{x+y} \right|.$$

Dieser Term kann sehr groß werden, wenn der Nenner fast verschwindet, der Zähler aber nicht, also für $x \sim -y$.

2. $f(x,y) = xy$

$$M_x = \max_{\xi} \left| \frac{\xi y}{xy} \right| \sim 1$$

für $x \sim \tilde{x}$.

Wir folgern daraus: Die Multiplikation ist problemlos, bei der Addition zweier Zahlen x und y kann der relative Fehler explodieren, falls $x \sim -y$. Dieses Phänomen heißt Auslöschung.

Definition 2.41 *(Kondition und Stabilität)*

1. *Ein Problem heißt* **gut gestellt** *(***gut konditioniert***), wenn kleine Änderungen in den Parametern zu kleinen Änderungen im Ergebnis führen. Ein Problem heißt* **schlecht gestellt** *(***schlecht konditioniert***), wenn kleine Änderungen in den Parametern zu großen Änderungen im Ergebnis führen. Dieser Fehler ist rein analytisch und* **unvermeidbar**.

2. *Ein Algorithmus zur Lösung eines Problems heißt* **(vorwärts–)stabil**, *falls er bei kleinen Änderungen der Eingangsdaten ein Ergebnis liefert, dessen Fehler (***Algorithmusfehler***) in der Größenordnung des analytischen Fehlers liegt. Ansonsten heißt er* **instabil**.

3. *Für das Problem $y = f(x)$ betrachten wir den implementierten Algorithmus $\tilde{y} = g(x)$. Falls $\tilde{y} = f(\tilde{x})$ für ein $\tilde{x}$ mit $||x - \tilde{x}||/||x||$ klein, so heißt g* **rückwärtsstabil**. *Es gilt: Rückwärtsstabile Algorithmen sind (vorwärts–) stabil.*

Was im Einzelfall klein oder groß heißt, wird durch die Anwendung vorgegeben.
Beweis zur Bemerkung: Es sei x exakt bekannt, die Näherung $\tilde{y}$ für $y = f(x)$ werde auf einem Rechner ausgerechnet, es sei $\tilde{y} = f(\tilde{x})$ und $f \in C^1$. Dann ist der unvermeidbare Fehler beschränkt durch

$$\text{eps} \sup_{\xi} \left| \frac{f'(\xi)}{f(x)} x \right|.$$

Für den tatsächlichen relativen Fehler gilt

$$\left|\frac{\tilde{y}-y}{y}\right| = \left|\frac{f(\tilde{x})-f(x)}{f(x)}\right| = \left|\frac{\tilde{x}-x}{x}\right| \cdot \left|\frac{xf'(\xi)}{f(x)}\right|$$

Liegt also der relative Fehler von $\tilde{x}$ zu x in der Größenordnung von eps, so ist der Algorithmus auch vorwärtsstabil.

Korollar 2.42

1. *Falls die Konditionszahlen einer Funktion f klein sind, so ist die Auswertung von f gut gestellt.*
2. *Die Auswertung der Multiplikation ist ein gut gestelltes Problem.*
3. *Die Auswertung der Addition ist ein schlecht gestelltes Problem, falls die Argumente unterschiedliches Vorzeichen und (fast) gleichen Betrag haben.*

Zur Illustration der Stabilität betrachten wir $f(x) = x$. Offensichtlich hat f den Verstärkungsfaktor 1, der unvermeidbare Fehler ist gleich dem Eingangsfehler. Zur Auswertung von f benutzen wir den Algorithmus

$$y = (x \oplus 1) \ominus 1.$$

Oben haben wir bereits gesehen, dass für $x = 10^{-30}$ der Fehler 100% beträgt, unabhängig vom Fehler in x. Dieser Algorithmus ist also sicherlich nicht stabil, wir haben durch die Addition der 1 eine künstliche Auslöschung erzeugt.
Häufig ist die Auslöschung aber nicht so offensichtlich. So gilt etwa

Satz 2.43 *Die direkte Implementation der pq–Formel zur Lösung quadratischer Gleichungen ist instabil.*

Beweis: Es lassen sich leicht p, q angeben, so dass der Algorithmusfehler beliebig viel größer ist als der unvermeidbare Fehler, siehe Übungen. □

Ein häufig angewandter Trick zur Auswertung schwieriger Funktionen ist die Reihenentwicklung. In der numerischen Behandlung gewöhnlicher Differentialgleichungen tauchen häufig Terme der Form

$$\sqrt{1+\epsilon^2} - 1$$

für kleine $\epsilon > 0$ auf. Würde man diesen Term so ausrechnen, wie er dort steht, würde ϵ^2 komplett in der 1 aufgehen, und das Ergebnis wäre 0 unabhängig von ϵ.

Andererseits ist die Konditionszahl der Funktion fast 1, der Fehler ist also nicht unvermeidbar. Wir nutzen in diesen Fällen die Taylorreihe der Wurzel und erhalten

$$\sqrt{1+\epsilon^2} - 1 = 1 + \frac{1}{2}\epsilon^2 - \frac{1}{8}\tilde{\epsilon}^4 - 1 \sim \frac{1}{2}\epsilon^2$$

was tatsächlich für kleine ϵ eine gute Näherung ist.
Als letzte Anwendung werden wir den unvermeidbaren Fehler bei der Lösung eines linearen Gleichungssystems

$$Ax = b$$

mit einer invertierbaren $n \times n$–Matrix A und $b \in \mathbb{R}^n$ berechnen. Wir bestimmen also eine Abschätzung für den Fehler, der entsteht, wenn die Koeffizienten der invertierbaren Matrix A oder des Vektors b nicht genau bekannt sind, sondern statt dessen nur Näherungen $A + \Delta A$ und $b + \Delta b$ zur Verfügung stehen und wir ersatzweise die Lösung des Gleichungssystems

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

berechnen.

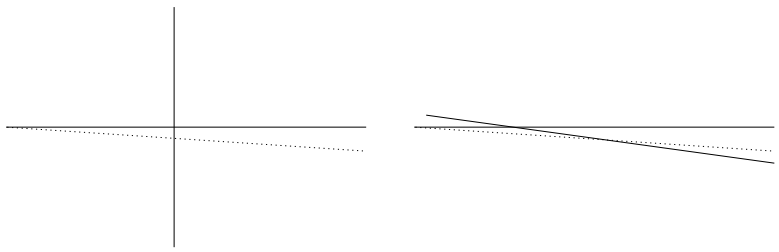

Abbildung 2.1: Graphische Lösung von Gleichungssystemen: Links gut gestellt, rechts schlecht gestellt, kleine Änderungen (gestrichelte Linie) in den Koeffizienten führen zu großer Änderung des Schnittpunkts.

Sei zunächst $n = 2$. Dann können wir die Lösung des Gleichungssystems als Schnittpunkt zweier Geraden im $\mathbb{R}^2$ graphisch bestimmen. Kleine Änderungen in den Koeffizienten führen zu kleinen Änderungen in der Lage der Linien. **Aber:** Falls die Linien fast parallel liegen, führt eine kleine Änderung in der Lage der Linien zu großen Änderungen beim Schnittpunkt. Die Verstärkung des Eingangsfehlers muss also von der Richtung der Linien, also von A, abhängen.

Satz 2.44 *Sei $A \in \mathbb{R}^{n\times n}$ invertierbar. Sei $x \in \mathbb{R}^n$ und $Ax = b$. Sei weiter $\Delta A \in \mathbb{R}^{n\times n}$ und $\Delta b \in \mathbb{R}^n$. Es sei*

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

die **Kondition** *von A und es gelte*

$$q = k(A)\frac{||\Delta A||}{||A||} < 1.$$

Dann ist $A + \Delta A$ invertierbar. Sei $\tilde{x} = x + \Delta x$ die Lösung von

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

Dann gilt für den relativen Fehler in der Lösung

$$\frac{||\Delta x||}{||x||} \leq \frac{k(A)}{1-q}\left(\underbrace{\frac{||\Delta b||}{||b||}}_{\text{rel.Fehler in b}} + \underbrace{\frac{||\Delta A||}{||A||}}_{\text{rel. Fehler in A}}\right)$$

Die relativen Fehler in A und b werden also (höchstens) um den Faktor

$$M = k(A)/(1-q)$$

verstärkt.

Für sinnvolle Anwendungen ist $||\Delta A||$ klein gegen $||A||$, also $q \sim 0$ und damit $M \sim k(A)$.
Beweis: Nach 2.32 ist $A + \Delta A$ invertierbar, und es gilt

$$||(A + \Delta A)^{-1}|| \leq \frac{||A^{-1}||}{1-q}.$$

Es gilt

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

und damit wegen $Ax = b$

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

und

$$\Delta x = (A + \Delta A)^{-1}(\Delta b - \Delta A x),$$

also insbesondere

$$||\Delta x|| \leq ||(A + \Delta A)^{-1}||(||\Delta b|| + ||\Delta A||\,||x||).$$

Für den relativen Fehler für $x \neq 0$

$$\begin{aligned}
\frac{||\Delta x||}{||x||} &\leq \frac{||A^{-1}||}{1-q}\left(\frac{||\Delta b||}{||x||} + ||\Delta A||\right) \\
&= \frac{k(A)}{1-q}\left(\frac{||\Delta b||}{||A||\,||x||} + \frac{||\Delta A||}{||A||}\right) \\
&\leq \frac{k(A)}{1-q}\left(\frac{||\Delta b||}{||b||} + \frac{||\Delta A||}{||A||}\right)
\end{aligned}$$

wegen $||b|| = ||Ax|| \leq ||A||\,||x||$. □

Kapitel 3

Direkte Verfahren zur Lösung linearer Gleichungssysteme

Fast jedes praktische Problem führt am Ende nach langer Modellierung auf ein lineares Gleichungssystem. Deshalb ist ihre Lösung von fundamentaler Bedeutung für die Angewandte Mathematik. Wir betrachten zunächst direkte Verfahren, die in endlicher Zeit eine Lösung liefern, gegenüber iterativen Verfahren, bei denen eine Folge ausgerechnet wird, die gegen die Lösung konvergiert. Direkte Verfahren sind dabei typischerweise langsam für große Matrizen und spielen heute eine untergeordnete Rolle.
Eine gute Quelle für klassische Algorithmen und Analysen zu diesem Bereich ist das Buch von Golub und van Loan, Matrix Computations.

3.1 Gauß–Elimination und LR–Zerlegung

Die **Gauß–Elimination** sollte bereits aus der Schule bekannt sein. Wir rechnen trotzdem zur Einführung ein Mikro–Beispiel.

$$
\begin{array}{rcrcrcrcrcl}
3 & x_1 & + & 2 & x_2 & + & & x_3 & = & 8 & \\
6 & x_1 & + & 5 & x_2 & - & 4 & x_3 & = & 12 & \equiv \quad \mathbf{A^{(1)}x = b^{(1)}} \\
-3 & x_1 & + & & x_2 & - & 2 & x_3 & = & -3 &
\end{array}
$$

$$
\begin{array}{rcrcrcrcrcl}
3 & x_1 & + & 2 & x_2 & + & & x_3 & = & 8 & \\
 & & & & x_2 & - & 6 & x_3 & = & -4 & \equiv \quad \mathbf{A^{(2)}x = b^{(2)}} \\
 & & & 3 & x_2 & - & & x_3 & = & 5 &
\end{array}
$$

$$
\begin{array}{rcrcrcrcrcl}
3 & x_1 & + & 2 & x_2 & + & & x_3 & = & 8 & \\
 & & & & x_2 & - & 6 & x_3 & = & -4 & \equiv \quad \mathbf{A^{(3)}x = b^{(3)}} \\
 & & & & & & 17 & x_3 & = & 17 &
\end{array}
$$

Durch **Rückwärtseinsetzen** ergibt sich damit

$$x_3 = 17/17 = 1,\ x_2 = (-4+6)/1 = 2,\ x_3 = (8-1-2\cdot 2)/3 = 1.$$

Wir werden Algorithmen immer in einem Pseudocode formulieren.
Zu lösen sei $Ax = b$, $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$. Setze $A^{(1)} = A$ und $b^{(1)} = b$. Es sei $A^{(k)} = (a_{jl}^{(k)})$ usw.

Für $i = 1 \dots n-1$
    Zur Konstruktion des Gleichungssystems $A^{(i+1)}x = b^{(i+1)}$
    Übernehme die ersten i Gleichungen, d.h. die ersten i Zeilen.
    Für $j = i+1 \dots n$
        $l_{ji} = \frac{a_{ji}^{(i)}}{a_{ii}^{(i)}}$ falls $a_{ii}^{(i)} \neq 0$.
        Für $k = i+1 \dots n$
            $a_{jk}^{(i+1)} = a_{jk}^{(i)} - l_{ji} \cdot a_{ik}^{(i)}$
        $b_j^{(i+1)} = b_j^{(i)} - l_{ji} b_i^{(i)}$
    Setze die restlichen Einträge auf 0.
Für $i = n \dots 1$
    $x_i = \left(b_i^{(n)} - \sum_{j=i+1}^{n} a_{ij}^{(n)} x_j \right) / a_{ii}^{(i)}.$

Hierbei benötigen wir die Matrizen $A^{(k)}$ zur Berechnung der Lösung nicht, es liegt also nahe, jeweils $A^{(k)}$ mit $A^{(k+1)}$ zu überschreiben. Es wird also im Laufe des Algorithmus kein zusätzlicher Speicherplatz benötigt.
Wir bestimmen den Aufwand zur Lösung des Systems. Wir vereinbaren zunächst: Da Addition und Multiplikation fast immer zusammen auftreten, zählen wir sie als eine **Rechenoperation**. Tatsächlich sind moderne Rechnerarchitekturen in der Lage, diese beiden Operationen gleichzeitig durchzuführen (fused multiply add), was, wie man sich schnell überlegt, den IEEE–Standard verletzt.

Für das Auflösen des Gleichungssystems werden dann

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(2+\sum_{k=i+1}^{n}1\right)=\frac{1}{6}(2n^3+3n^2-5n)=\frac{1}{6}n(n-1)(2n+5)$$

Rechenoperationen und n Divisionen benötigt, wobei wir für die einzelnen Divisionen jeweils einmal den Kehrwert der $a_{ii}^{(i)}$ ausrechnen und dann mit ihm multiplizieren. Die Division ist nämlich tatsächlich recht aufwändig, einen Algorithmus zu ihrer schnellen Berechnung (mit einigen Rechenoperationen) werden wir im Kapitel über die Newton–Iteration herleiten.
Zur Durchführung des Rückwärtseinsetzens erhalten wir

$$\sum_{i=1}^{n}(2+\sum_{j=i+1}^{n}1)=n^2/2+7/2\,n.$$

Alle Berechnungen dieser Art interessieren uns immer nur für große n. Dann dominieren aber sofort die Terme mit hoher Potenz die mit kleiner, und nur der Leitterm mit der höchsten Potenz ist interessant. Es würde also reichen, den größten Term (mit einer Abschätzung für den Rest) zu kennen. Wir definieren daher die Landau-Symbole:

Definition 3.1 *(**Landau–Symbole**)*

1. *Seien* $f,\ g:\mathbb{N}\mapsto\mathbb{N}$.

   $$f(n)=O(g(n))\ \textit{für große}\ n\Leftrightarrow\exists C>0,\ n_0>0:\ |f(n)|\le C|g(n)|\,\forall n>n_0.$$

2. *Seien* $f,\ g:\mathbb{R}\mapsto\mathbb{R}$.

   $$f(h)=O(g(h))\ \textit{für kleine}\ h\Leftrightarrow\exists C>0,\ h_0>0:\ |f(h)|\le C|g(h)|\,\forall 0<h<h_0.$$

Wenn der Zusammenhang klar ist, werden wir den Zusatz weglassen.

Beispiel 3.2

1. $O(n^\alpha)=O(n^\beta)$ *für* $0<\alpha\le\beta$.
   $\sin(x)=O(1)$ *für große* x.
2. $O(h^\alpha)=O(h^\beta)$ *für* $\alpha\ge\beta>0$.
   $\sin(x)=O(1)$ *für kleine* x.
   $\sin(x)=O(x)$ *für kleine* x.

Mit dieser Konvention gilt

Satz 3.3 *Die Auflösung einer Gleichung mit n Unbekannten mit dem Gauß–Algorithmus benötigt $n^3/3 + O(n^2)$ Rechenoperationen und n Divisionen.*

Bemerkung:

1. Die Cramersche Regel rechnet die Determinanten der Matrix aus, was bei direkter Berechnung die Komplexität $O(n!)$ hat (und damit völlig unbrauchbar ist).

2. Die Gauss–Elimination ist durchführbar genau dann, wenn alle $a_{ii}^{(i)} \neq 0$.

3. Falls $a_{ii}^{(i)} = 0$, aber $a_{ki}^{(i)} \neq 0$ für ein $k > i$, so vertausche die $k.$ und die $i.$ Zeile des Gleichungssystems (was die Lösung natürlich nicht ändert).

4. Falls $a_{ki}^{(i)} = 0$ für alle $k \geq i$, so ist x_i aus den Gleichungen i bis n bereits eliminiert. In diesem Fall hat $A^{(i)}$ die Form

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

   Entwicklung der Determinante nach der ersten Spalte zeigt sofort: Dann ist $A^{(i)}$ singulär, und damit auch A. Falls A invertierbar ist, kann dieser Fall also nicht auftreten.

   Die Elimination ist auf einer Permutation des Systems aber immer ausführbar.

5. Setze $R := A^{(n)}$. Dann gilt

$$R_{ik} = 0 \text{ für } i > k.$$

   R mit dieser Eigenschaft (alle Elemente unterhalb der Hauptdiagonalen verschwinden) heißt **rechte obere Dreiecksmatrix**. Entsprechend heißt eine Matrix L **linke untere Dreiecksmatrix**, falls alle Elemente oberhalb der Hauptdiagonalen verschwinden, d.h.

$$L_{ik} = 0 \text{ für } i < k.$$

Falls alle $L_{ii} = 1$, so heißt L normiert. Produkte von (normierten) linken unteren Dreiecksmatrizen sind (normierte) linke untere Dreiecksmatrizen usw., die Dreiecksmatrizen bilden algebraisch jeweils einen Ring.
Wir lassen in der allgemeinen Definition zu, dass Dreiecksmatrizen nicht quadratisch sind, dann sind sie natürlich nicht miteinander multiplizierbar.

6. Fehleranalyse: Eine genaue Fehleranalyse ist für den Gauß–Algorithmus schwierig. Wir betrachten nur die Berechnung von x_1. $\tilde{x_1}$ wird berechnet durch

$$\tilde{x_1} = \frac{1}{a_{11}} \underbrace{(b_1 - \sum_{k=2}^{n} a_{1k}\tilde{x_k})}_{a_{11}x_1}.$$

Ist nun $|a_{11}x_1|$ klein gegenüber b_1, so kann dies nur dadurch entstanden sein, dass in der Differenz Auslöschung aufgetreten ist. Fehler werden also stark verstärkt, wenn $|a_{11}|$ klein ist. Wir ordnen deshalb im $i.$ Schritt die Gleichungen so an, dass das Diagonalelement in der $i.$ Spalte unterhalb der Hauptdiagonalen betragsmaximal ist, dass also gilt

$$|a_{ii}^{(i)}| = \max_{k \geq i} |a_{ki}^{(i)}|.$$

Diese Strategie heißt **Spaltenpivotsuche** und macht den Gaußalgorithmus bereits zu einem stabilen (in praktischen Fällen). Alternativ kann man in jedem Schritt zusätzlich auch die Variablen umbenennen (also die Spalten von A umordnen), so dass auf der Diagonalen das betragsmäßig größte Element der rechten unteren Teilmatrix erscheint. Diese Strategie heißt **totale Pivotsuche**.

Wir rechnen dazu ein kurzes Beispiel, zunächst ohne Pivotsuche. Wir benutzen zur Rechnung das Human Format.

$$\begin{array}{llll} 10^{-4} & x_1 + & x_2 = & 1 + 10^{-4} \\ & x_1 + & x_2 = & 2 \end{array}$$

$$\begin{array}{llll} 10^{-4} & x_1 + & x_2 = & 1 \\ & \underbrace{(-10^4 \oplus 1)}_{=-10^4} & x_2 = & \underbrace{-10^4 \oplus 2}_{=-10^4} \end{array}$$

und damit $\tilde{x_2} = -10^4 \oslash -10^4 = 1$ und $\tilde{x_1} = (1 \ominus 1)/10^{-4} = 0$. Da $x = (1, 1)$ die korrekte Lösung ist, hat x_1 einen Fehler von 100% (genau wie oben vorausgesagt sorgt die Auslöschung beim Rückwärtseinsetzen für einen großen Fehler). Die Kondition der Matrix ist kleiner als drei, dieser Fehler ist also nicht

unvermeidbar. Nun dasselbe mit Pivotsuche:

$$\begin{array}{rcrcl} & x_1 & + & x_2 & = & 2 \\ 10^{-4} & x_1 & + & x_2 & = & 1+10^{-4} \end{array}$$

$$\begin{array}{rcrcl} & x_1 & + & x_2 & = & 2 \\ & & \underbrace{(1 \ominus 10^{-4})}_{=1} & x_2 & = & \underbrace{1 \ominus 2 \cdot 10^{-4}}_{=1} \end{array}$$

und wir erhalten die korrekte Lösung $x_2 = 1$, $x_1 = 1$.

7. **Platzbedarf**: Üblicherweise wird die Matrix A durch $A^{(n)}$ überschrieben. Die dabei nicht mehr genutzten Einträge unterhalb der Hauptdiagonalen nutzt man, um sich die Zahlen l_{ji} zu merken, also im i. Schritt

$$A_{ji} = l_{ji},\ j > i$$

und

$$A_{jk} = A_{jk}^{(i+1)},\ k > i,\ j > i.$$

Damit ist es möglich, sofort ein weiteres Gleichungssystem $Ax = b'$ zu lösen, ohne die Elimination erneut durchführen zu müssen. Tatsächlich wird üblicherweise zunächst der Eliminationsschritt durchgeführt (mit Aufwand $n^3 + O(n^2)$) und dann die rechte Seite eingesetzt (mit Aufwand $n^2 + O(n)$). Falls das Gleichungssystem permutiert wird, werden die l_{ik} mitpermutiert. Die Permutation muss ebenfalls gespeichert werden.

8. Viele Gleichungssysteme können schneller als angegeben aufgelöst werden. Für die **Inversion** einer Matrix etwa löst man die Gleichungssysteme

$$Ax_k = e_k$$

für die Spalten x_k von A^{-1}. Durch Nutzen der Nullen auf der rechten Seite lässt sich die Inverse in $n^3 + O(n^2)$ Rechenoperationen berechnen.
Andererseits: Sei A eine Matrix, bei der nur die b Nebendiagonalen besetzt sind, also

$$A_{ik} = 0 \text{ für } |i-k| > b.$$

Dann hat A die **Bandbreite** b. Die LR–Zerlegung lässt sich für kleine b in $nb^2 + O(nb)$ Rechenoperationen berechnen. (Übungen)
Insbesondere lässt sich die LR–Zerlegung von **Tridiagonalmatrizen** (Matrizen mit Bandbreite 1) mit $O(n)$ Rechenoperationen berechnen.

9. Der Rechenaufwand zur Lösung eines Gleichungssystems lässt sich durch völlig andere Methoden sogar im Exponenten der Komplexität reduzieren. Schon 1969 zeigte Strassen in seiner damals sensationellen, nur drei Seiten langen Arbeit **"Gaussian Elimination is not Optimal"**, dass sich der Aufwand zur Inversion bzw. Multiplikation von Matrizen von n^3 auf $n^{\log_2 7}$ drücken lässt (Beispielimplementation). Inzwischen sind weitere Algorithmen dieser Art bekannt, allgemein wird vermutet, dass die Untergrenze für die Komplexität tatsächlich $O(n^2)$ ist.
   Leider sind die so entstehenden Algorithmen alle instabil und bieten wegen der großen Konstanten im $O(\ldots)$ nur für extrem große Matrizen einen theoretischen Vorteil.

Wir werden die Gauß–Elimination nun mit Hilfe von Elementar– und Permutationsmatrizen beschreiben.

Definition 3.4

1. *Eine normierte linke untere Dreiecksmatrix $L = (l_{kj})$ heißt* **Elementarmatrix,** *wenn nur in einer Spalte unterhalb der Hauptdiagonalen Einträge ungleich 0 sind, d.h.*
$$\exists i : \; l_{kj} = 0,\; k > j,\; j \neq i.$$
2. *Eine Matrix $P \in \mathbb{R}^{n\times n}$ heißt* **Permutationsmatrix,** *falls in jeder Zeile und Spalte genau eine 1 auftaucht und alle anderen Einträge 0 sind, d.h.*
$$\exists\, \sigma \in \{1 \ldots n\}^n : \; \sigma_k \neq \sigma_j \textit{ für } k \neq j,\; a_{i,k} = \begin{cases} 1, & k = \sigma_i \\ 0, & k \neq \sigma_i \end{cases}.$$

Beispiel 3.5

$$L = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{i+1,i} & 1 & & \\ & & \vdots & & \ddots & \\ & & l_{n,i} & & & 1 \end{pmatrix}$$

ist Elementarmatrix.

$$P = \begin{pmatrix} & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & & & \end{pmatrix},\; P^t = \begin{pmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{pmatrix}$$

sind Permutationsmatrizen zu $\sigma = (2, 3, 4, 1)$ *bzw.* $(4, 1, 2, 3)$. *Offensichtlich gilt*

$$PP^t = P^t P = I.$$

Bemerkung:

1. LA lässt die ersten i Zeilen von A konstant und addiert jeweils auf die Zeilen $i + 1$ bis n das $l_{j,i}$–fache der i. Zeile auf.

2. Die Inverse von L ergibt sich durch Multiplizieren der Elemente unterhalb der Hauptdiagonalen mit -1, denn um die Operation wieder rückgängig zu machen, muss die $i.$ Zeile entsprechend wieder abgezogen werden.

3. Das Produkt zweier Elementarmatrizen $L^{(i)}L^{(i+1)}$ zu den Spalten i und $i+1$ ist eine normierte linke untere Dreiecksmatrix, die durch Überlagerung von $L^{(i)}$ und $L^{(i+1)}$ entsteht.
   Begründung: Statt erst ein Vielfaches der $(i + 1).$ und dann ein Vielfaches der i. Zeile zu addieren, kann man beides gleichzeitig tun.

4. PA bringt die Zeilen von A in die Reihenfolge σ. AP^t bringt die Spalten in die Reihenfolge σ. PAP^t bringt Zeilen und Spalten in die vorgegebene Reihenfolge, insbesondere bleiben Diagonalelemente Diagonalelemente.

5. Produkte von Permutationsmatrizen sind Permutationsmatrizen.

6. Sei L_i eine Elementarmatrix zur Spalte i und P eine Permutation mit $Pe_k = e_k$ für $k \leq i$. Dann ist $L'_i = PL_iP^t$ wieder Elementarmatrix, es gilt $PL_i = L'_iP$.
   Beweis: Sei $L_i = (I + F)$. F hat nur Einträge in der $i.$ Spalte ab Zeile $i + 1$.

   $$\begin{aligned} PL_iP^t &= P(I + F)P^t \\ &= I + PFP^t. \end{aligned}$$

   Da Rechtsmultiplikation mit P^t nur die Spalten k mit $k > i$ betrifft, dort aber nur Nullspalten stehen, ist $FP^t = F$. Da Linksmultiplikation nur die Zeilen j mit $j > i$ vertauscht, ändert F seine Form nicht und $(I + PF)$ ist wieder Elementarmatrix. □

Offensichtlich sind das genau die Matrizen, die wir zur exakten Beschreibung des Gauß–Algorithmus benötigen. Wir formulieren dies als

Satz 3.6 *(**LR-Zerlegung**, engl. LU–Zerlegung)*
Sei A eine $n \times n$–Matrix. Dann gibt es eine Permutationsmatrix P zur Permutation σ, eine normierte linke untere Dreiecksmatrix L und eine rechte obere Dreiecksmatrix R (alles $(n \times n)$), so dass

$$PA = LR.$$

L und R heißen LR–Zerlegung von PA.

Beweis: Wir führen die Gauss–Elimination an $A = A^{(1)}$ durch. Es sei $A^{(i)}$ bereits berechnet. Falls $a_{ki}^{(i)} = 0$ für alle $k \geq i$, so ist x_i bereits eliminiert und wir setzen $L_i = I$, $P_i = I$.
Falls mindestens ein Element der i. Spalte unter oder auf der Hauptdiagonalen nicht Null ist, wählen wir eine Permutationsmatrix $P^{(i)}$, so dass die Zeilenvertauschung $P^{(i)}A^{(i)}$ ein solches Element an die Stelle (i, i) auf der Hauptdiagonalen bringt (z.B. ein Element mit maximalem Betrag in der Spaltenpivotsuche). $P^{(i)}$ lässt dabei natürlich die ersten i Zeilen fest.
Wir wählen die l_{ji}, $j > i$, wie in der Gauss–Elimination als

$$l_{ji} = \frac{(PA^{(i)})_{ji}}{(PA^{(i)})_{ii}}$$

und

$$L^{(i)} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{i+1,i} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{n,i} & & 1 \end{pmatrix}.$$

Die Matrix $A^{(i+1)}$ aus der Gauss–Elimination ergibt sich dann durch

$$A^{(i+1)} = L^{(i)}P^{(i)}A^{(i)}.$$

Nach $n - 1$ Schritten erhält man so aus $A^{(1)} = A$ die rechte obere Dreiecksmatrix $A^{(n)} = R$. Es gilt nach den Vorbemerkungen

$$\begin{aligned} R = A^{(n)} &= L^{(n-1)}P^{(n-1)}L^{(n-2)} \cdots L^{(1)}P^{(1)}A \\ &= L'^{(n-1)} \cdots L'^{(1)}P^{(n-1)} \cdots P^{(1)}A \end{aligned}$$

und damit

$$\underbrace{P^{(n-1)} \cdots P^{(1)}}_{=P} A = \underbrace{(L'^{(1)})^{-1} \cdots (L'^{(n-1)})^{-1}}_{=L} R$$

wobei $L_i'^{-1}$ aus L_i durch Permutieren und Multiplizieren der Elemente unterhalb der Hauptdiagonalen mit -1 entsteht. L ist eine unitäre linke untere Dreiecksmatrix, ihre Einträge unterhalb der Hauptdiagonalen ist die Überlagerung der Elementarmatrizen, und das sind gerade die Einträge $l_{\sigma_j,k}$ mit der Permutation σ zu P. □

Bemerkung:

1. Sei $PA = LR$. Dann kann $Ax = b$ gelöst werden mittels

$$\begin{aligned} Ax = b &\iff PAx = Pb \\ &\iff LRx = Pb \\ &\iff Ly = Pb,\ Rx = y \end{aligned}$$

   Dabei wird y bestimmt durch **Vorwärtseinsetzen** (Auflösung der Gleichungen von vorn nach hinten) in $Ly = Pb$ und x durch **Rückwärtseinsetzen** (Auflösung der Gleichungen von hinten nach vorn) in $Rx = y$.

2. Nicht jede invertierbare Matrix besitzt eine LR–Zerlegung. Sei etwa

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \cdot \begin{pmatrix} b & c \\ 0 & d \end{pmatrix} = \begin{pmatrix} b & c \\ ab & ac+d \end{pmatrix}.$$

   Dann gilt offensichtlich $b = 0$, also $ab = 0$ ↯.

3. Es gibt singuläre Matrizen, die eine LR–Zerlegung besitzen.

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

4. Die LR–Zerlegung mit Spaltenpivotsuche und Vorwärts–Rückwärtseinsetzen ist für praktische Zwecke ein gutartiger Algorithmus zur Bestimmung der Lösung eines linearen Gleichungssystems.

5. Der Aufwand zur Berechnung der LR–Zerlegung beträgt $n^3/3 + O(n^2)$ Rechenoperationen ($+n$ Divisionen). Der Aufwand für das Einsetzen beträgt $n^2 + O(n)$.

Satz 3.7 *(***Eindeutigkeit** *der LR–Zerlegung) Sei A eine invertierbare $n \times n$–Matrix. Falls A eine LR–Zerlegung besitzt, so ist diese Zerlegung eindeutig.*

Beweis: A besitze die LR–Zerlegungen (L, R) und (L', R'). Da A invertierbar ist, sind auch die Dreiecksmatrizen invertierbar. Es gilt

$$A = LR = L'R' \Longrightarrow (L')^{-1}L = R'R^{-1} =: Z.$$

Z ist Produkt linker unterer normierter Dreiecksmatrizen, also selbst wieder normierte linke untere Dreiecksmatrix. Andererseits ist Z Produkt rechter oberer Dreiecksmatrizen, also selbst wieder rechte obere Dreiecksmatrix. Damit ist Z Diagonalmatrix und hat, weil sie normiert ist, 1 auf der Hauptdiagonalen, ist also die Einheitsmatrix. Damit gilt

$$L = L' \text{ und } R = R'.$$

□

Satz 3.8 *(Existenz der LR–Zerlegung)*

1. *Sei A eine $n \times n$–Matrix. Alle Hauptminoren von A, d.h. die Determinanten aller quadratischen Teilmatrizen, die in der linken oberen Ecke beginnen, seien ungleich 0. Dann besitzt A eine LR–Zerlegung.*

2. *Sei A symmetrisch positiv definit. Dann besitzt A eine LR–Zerlegung.*

Beweis:
zu 1.: In den Übungen.
Zu 2.: Die Hauptminoren positiv definiter Matrizen sind positiv. □

```
function [ A,P,Q ] = LR( A, pivot )
%LR Compute LU decomposition. Accept symbolic input for pivot=0
% A - return R on upper right, return L on lower left
% diagonal of L is 1
% pivot=0: No pivoting, throw error when 0 appears on diagonal
% pivot=1: Column pivoting, if 0 appears on diagonal
```

Listing 3.1: LR–Zerlegung (LR-Zerlegung/LR.m)

Klicken für den Quellcode von LR-Zerlegung/LR.m

```
function B = checkLR( A,P,Q )
%CHECKLR check output of LR decomposition
%input arguments as in LR output arguments
%B is LR
[n m]=size(A);
B=zeros(n,m);
```

Listing 3.2: Berechnung von A aus der LR–Zerlegung (LR-Zerlegung/checkLR.m)

Klicken für den Quellcode von LR-Zerlegung/checkLR.m

```
function B = LRSolve( A,B,P,Q )
%LRSOLVE Solve AX=B. A,P,Q are from LR.
%B can be a matrix of vectors.
%Permute right hand side vectors.
[n m]=size(A);
if (size(B,1)~=m)
```

Listing 3.3: Lösung eines LGS mit der LR–Zerlegung (LR-Zerlegung/LRSolve.m)

Klicken für den Quellcode von LR-Zerlegung/LRSolve.m

```
function [ output_args ] = doit( N, pivot )
%DOIT Solve random linear equation of size N
%Remember: Pivoting effect does not show with random matrices.

if (nargin<1)
    N=128;
```

Listing 3.4: Lösung eines zufälligen LGS mit der LR–Zerlegung (LR-Zerlegung/doit.m)

Klicken für den Quellcode von LR-Zerlegung/doit.m

3.2 Cholesky–Zerlegung

Für symmetrisch positiv definite Matrizen (die in der Praxis häufig auftauchen) lässt sich der Aufwand für die LR–Zerlegung halbieren. Sei A also eine reelle $n \times n$–Matrix und s.p.d. Man könnte erwarten, dass symmetrische Matrizen eine Zerlegung der Form

$$A = LL^t$$

haben für eine Dreiecksmatrix L. Für die übliche LR–Zerlegung ist das im allgemeinen falsch, denn L ist dabei normiert. Für die Zerlegung von symmetrischen Matrizen verzichten wir daher auf die Normierung.
Sei $A = LR$ die LR–Zerlegung von A (A ist positiv definit, also besitzt sie eine LR–Zerlegung, und diese ist eindeutig, denn A ist als positiv definite Matrix invertierbar). Wegen $0 \neq \det A = \det L \det R$ sind die R_{ii} invertierbar. Sei D die Diagonalmatrix mit $D_{ii} = R_{ii}$. Dann ist $D^{-1}R$ eine normierte rechte obere Dreiecksmatrix und

$$A = A^t = (LDD^{-1}R)^t = (D^{-1}R)^t(LD)^t$$

eine LR–Zerlegung. Wegen der Eindeutigkeit gilt also $L = (D^{-1}R)^t$, und insbesondere

$$0 < (Ax, x) = (LRx, x) = (Rx, L^t x) = (Rx, D^{-1}Rx).$$

Einsetzen von $R^{-1}e_k$ liefert $D_{kk} > 0$, wir können also die Wurzel aus den Diagonalelementen ziehen. Sei D' die Diagonalmatrix mit $D'_{ii} = 1/\sqrt{R_{ii}}$. Dann gilt

$$A = LR = (D^{-1}R)^t R = R^t D^{-1} R = R^t D' D' R = (D'R)^t (D'R),$$

A lässt sich also tatsächlich als Produkt von zueinander transponierten Dreiecksmatrizen schreiben. Wir halten dieses Ergebnis fest in

Satz 3.9 *(**Cholesky–Zerlegung**)*
Sei A eine symmetrisch positiv definite $(n\times n)$–Matrix. Dann gibt es eine linke untere (nicht notwendig normierte) $(n \times n)$–Dreiecksmatrix L, so dass

$$A = LL^t.$$

Bei der Berechnung der Cholesky–Zerlegung kann also die ganz normale LR–Zerlegung berechnet werden. Die Elemente oberhalb der Hauptdiagonalen müssen aber wegen der Symmetrie nicht mitgezogen werden, es ergibt sich insgesamt also der halbe Aufwand. Da Zeilenvertauschungen die Symmetrie zerstören würden, muss dabei ohne Pivotsuche gearbeitet werden, sie ist aber bei s.p.d.–Matrizen tatsächlich auch nicht notwendig.
Der Algorithmus lässt sich leicht angeben. Seien a_k die Spalten der zu zerlegenden Matrix A und l_k die Spalten von L. L ist linke untere Dreiecksmatrix, d.h. $(l_k)_j = 0$ für $j > k$. Wegen

$$\begin{pmatrix} a_1 & a_2 & \dots a_n \end{pmatrix} = A = LL^t = \begin{pmatrix} l_1 & l_2 & \cdots & l_n \end{pmatrix} \begin{pmatrix} l_1^t \\ l_2^t \\ \vdots \\ l_n^t \end{pmatrix}$$

$$= \begin{pmatrix} \ddots & & \\ \vdots & \ddots & \\ & \cdots & \ddots \end{pmatrix} \begin{pmatrix} \ddots & \cdots & \\ & \ddots & \vdots \\ & & \ddots \end{pmatrix}$$

gilt in der ersten Spalte

$$a_1 = (l_1)_1 l_1$$

und damit

$$(l_1)_1 = \sqrt{(a_1)_1}$$

und

$$l_1 = \frac{1}{(l_1)_1} a_1.$$

Entsprechend für die k. Spalte

$$a_k = \sum_{i=1}^{k} (l_k)_i l_i$$

und damit

$$(l_k)_k l_k = a_k - \sum_{i=1}^{k-1} (l_k)_i l_i,$$

also

$$(l_k)_k = \sqrt{(a_k)_k - \sum_{i=1}^{k-1} (l_k)_i (l_i)_k}$$

und

$$l_k = \frac{1}{(l_k)_k} \left(a_k - \sum_{i=1}^{k-1} (l_k)_i l_i \right).$$

Satz 3.10 *(**Aufwand** der Cholesky–Zerlegung)*
Die Cholesky–Zerlegung lässt sich mit $n^3/6 + O(n^2)$ Rechenoperationen, n Divisionen und n Wurzelberechnungen berechnen.

Beweis: Übungen. □

```
function L = cholesky( A )
%CHOLESKY Compute Cholesky decomposition of A.
%A is assumed to be symmetric, only the lower left is used.
%We make sure that the function can be used on symbolic input.
L=zeros(size(A));
if (~isnumeric(A))
```

Listing 3.5: Cholesky–Zerlegung (Cholesky/cholesky.m)

Klicken für den Quellcode von Cholesky/cholesky.m

```
function A = testcholesky( L )
%TESTCHOLESKY Compute L L^t
%Should compute this directly.
%A=L*L';
[n n]=size(L);
A=zeros(size(L));
```

Listing 3.6: Berechnung von A aus der Choleskyzerlegung (cholesky/testcholesky.m)

Klicken für den Quellcode von cholesky/testcholesky.m

```
function B=solvecholesky( L,B )
%SOLVECHOLESKY Solve linear equation using Cholesky
%decomposition.
[n n]=size(L);
%Forward
for i=1:n
```

Listing 3.7: Berechnung der Lösung eines LGS aus der Choleskyzerlegung (cholesky/solvecholesky.m)

Klicken für den Quellcode von cholesky/solvecholesky.m

```
function doit( n )
%DOIT Generate random nxn s.p.d. matrix, compute its Cholesky
% decomposition, solve a random system.
if (nargin<1)
    n=3;
end
```

Listing 3.8: Lösung eines zufälligen LGS mit der Choleskyzerlegung (cholesky/doit.m)

Klicken für den Quellcode von cholesky/doit.m

3.3 QR–Zerlegung

Bei der LR–Zerlegung schreibt man eine Matrix als Produkt von Matrizen, für die das Gleichungssystem $Qx = b$ leicht gelöst werden kann. Statt Dreiecksmatrizen könnte man dazu auch unitäre Matrizen Q mit der Eigenschaft

$$Q^{-1} = Q^t$$

nutzen. Dies ist die Idee hinter der QR–Zerlegung.
Im gesamten Kapitel beschäftigen wir uns der Übersichtlichkeit wegen nur mit reellen Matrizen im $\mathbb{R}^n$ und dem Standard–Skalarprodukt. Alle Sätze sind geeignet sofort auch auf komplexe Vektorräume übertragbar.

Satz 3.11 *(**unitäre Matrizen**)*

1. *Sei $Q \in \mathbb{R}^{n\times n}$. Q ist genau dann unitär mit $Q^tQ = QQ^t = I$, wenn die Spalten und Zeilen von Q eine Orthonormalbasis bilden.*

2. *Sei $v \in \mathbb{R}^n$, $v \neq 0$. Dann beschreibt die Spiegelmatrix*

$$Q(v) = (I - 2\frac{vv^t}{v^tv})$$

eine Spiegelung an der Hyperebene $x \cdot v = 0$. $Q(v)$ ist unitär. $Q(v)$ heißt Householder–Spiegelung.

3. *Sei $\varphi \in \mathbb{R}$, $1 \leq i < k \leq n$. Dann beschreibt die Matrix*

$$R(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & & & \\ \sin\varphi & \cos\varphi & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

eine Rotation um φ in der (x_1, x_2)–Ebene. Sei nun P eine Permutationsmatrix mit Permutation σ, $i = \sigma_1$, $k = \sigma_2$. Dann beschreibt

$$R_{ik}(\varphi) = PR(\varphi)P^t$$

eine Rotation um φ in der (x_i, x_k)–Ebene und heißt Givens–Rotation. $R_{ik}(\varphi)$ ist unitär.

Beweis:

1. Seien q_k die Spalten von Q. Dann ist

$$(q_i, q_j) = q_i^t q_j = (Q^t Q)_{ij} = (I)_{ij} = \delta_{ij}$$

mit dem Kronecker–δ, also ist $\{q_k\}$ Orthonormalsystem (und umgekehrt).

2. Sei w Element der Spiegelebene, also $w^t v = 0$. Dann gilt

$$Q(v)v = (I - 2\frac{vv^t}{v^t v})v = v - 2v\frac{v^t v}{v^t v} = -v$$

$$Q(v)w = (I - 2\frac{vv^t}{v^t v})w = w - 2v\frac{v^t w}{v^t v} = w$$

und Q ist Spiegelmatrix an der Ebene. $Q(v)$ ist symmetrisch, und es gilt

$$(Q(v)Q(v))v = v,\ (Q(v)Q(v))w = w$$

nach der Vorbemerkung, also ist Q unitär. Wir halten auch noch gleich fest, dass die Multiplikation $Q(v)y$ schnell ausgeführt werden kann:

$$Q(v)y = (I - 2\frac{vv^t}{v^t v})y = y - 2v\frac{v^t y}{v^t v}$$

und dies ist mit $2n$ Rechenoperationen (gegenüber n^2 für die normale Matrix–Vektor–Multiplikation) berechenbar, wenn $v^t v$ berechnet ist.

3. Durch Nachrechnen.

□

Satz 3.12 *(**Gram–Schmidtsches Orthonormalisierungsverfahren, kurz Schmidtsches Orthonormalisierungsverfahren**)*
Seien $a_1, \ldots, a_n \in \mathbb{R}^m$ linear unabhängig, also insbesodere $m \geq n$. Seien q_k und $\tilde{q}_k$ im $\mathbb{R}^m$ mit

$$\tilde{q_1} = a_1,\ q_1 = \frac{\tilde{q_1}}{||\tilde{q_1}||}$$

und

$$\tilde{q_k} = a_k - \sum_{j=1}^{k-1}(q_j, a_k)q_j,\ q_k = \frac{\tilde{q_k}}{||\tilde{q_k}||},\ k = 2 \ldots n.$$

Dann bilden die q_k ein Orthonormalsystem, und es gilt für die lineare Hülle

$$< q_1, \ldots, q_k > = < a_1, \ldots, a_k >,\ k = 1 \ldots n.$$

Weiter sei Q die Matrix mit den Spalten q_k, A die Matrix mit den Spalten a_k. Dann gibt es eine rechte obere $(n \times n)$–Dreiecksmatrix R mit $A = QR$. Die Spalten von Q bilden ein Orthonormalsystem. Falls $m = n$, so ist Q unitär.

Beweis: Der Satz sei für $a_1 \dots a_{k-1}$ bereits bewiesen, $q_1 \dots q_{k-1}$ stehen also senkrecht aufeinander. Sei $l < k$. Dann gilt

$$\begin{aligned}(\tilde{q}_k, q_l) &= (a_k - \sum_{j=1}^{k-1}(q_j, a_k)q_j, q_l) \\ &= (a_k, q_l) - (q_l, a_k)(q_l, q_l) \\ &= 0.\end{aligned}$$

$\tilde{q}_k$ und damit auch q_k stehen also senkrecht auf allen q_l mit $l < k$.
q_k liegt nach Definition und Induktionsvoraussetzung in der linearen Hülle $< a_1, \dots, a_k >$. Andererseits gilt

$$a_k = \tilde{q}_k + \sum_{j=1}^{k-1}(q_j, a_k)q_j$$

und damit gilt auch

$$< q_1, \dots, q_k > = < a_1, \dots, a_k > .$$

Es gibt also für jedes q_k Koeffizienten $r_{l,k}$, $k = 1 \dots m$, $l = 1 \dots k$, so dass

$$a_k = \sum_{l=1}^{k} r_{lk} q_l$$

oder

$$A = QR$$

mit der rechten oberen Dreiecksmatrix $R_{lk} = r_{l,k}$ für $k < l$ und 0 sonst. □
Mit dieser Methode lassen sich also zumindest invertierbare Matrizen A in ein Produkt aus einer unitären und einer rechten oberen Dreiecksmatrix zerlegen. Das Gleichungssystem $Ax = b$ löst man dann durch

$$Ax = b \Longleftrightarrow QRx = b \Longleftrightarrow Qy = b,\ y = Rx$$

und damit $y = Q^t b$ und x kann durch Rückwärtseinsetzen bestimmt werden.
Wir erweitern die Aussage zu

Satz 3.13 *(**Definition und Existenz der** QR**–Zerlegung**)*
Sei A eine $(m \times n)$–Matrix.

1. *Falls $m = n$, so gibt es eine unitäre $n \times n$–Matrix Q und eine $(n \times n)$ rechte obere Dreiecksmatrix R, so dass $A = QR$.*
2. *Falls $m > n$, so gibt es eine $m \times n$–Matrix Q mit $Q^tQ = I_n$ und eine $(n \times n)$ rechte obere Dreiecksmatrix R, so dass $A = QR$.*

3. *Falls $m < n$, so gibt es eine unitäre $m \times m$–Matrix Q und eine $(m \times n)$–Matrix R, deren Elemente unterhalb der Hauptdiagonalen verschwinden, so dass $A = QR$.*

Diese Zerlegung heißt QR–Zerlegung von A.

Beweis: Sei zunächst $m \geq n$ und seien a_k die Spalten von A.
Das Schmidtsche Orthogonalisierungsverfahren auf den a_k versagt, falls $\tilde{q}_k = 0$. In diesem Fall wählen wir für q_k irgendeinen normierten Vektor, der zu $q_1 \ldots, q_{k-1}$ orthogonal ist. Dies ist möglich, da $m \geq n$. Die Spalten der Matrix $Q = (q_1, \ldots, q_n)$ stehen also senkrecht aufeinander, es gilt

$$Q^t Q = I_n.$$

a_k liegt in der linearen Hülle $< q_1, \ldots, q_k >$, also gibt es wie beim Schmidtschen Orthogonalisierungsverfahren ein R mit $A = QR$. Es gilt $R_{kk} = 0$ falls $\tilde{q}_k = 0$.
Für $m < n$ wenden wir diese Methode auf die Vektoren $a_1, \ldots, a_m$ an und erhalten wieder $(a_1, \ldots, a_m) = QR$. $q_1, \ldots, q_m$ ist eine Basis, damit liegen die Vektoren $a_{m+1} \ldots a_n$ in ihrer linearen Hülle, es gilt $(a_{m+1}, \ldots, a_n) = QB$ für eine $(n \times (n-m))$–Matrix B und damit

$$(a_1, \ldots, a_n) = Q[R; B].$$

□

Anders als bei der LR–Zerlegung existiert die QR–Zerlegung also auch ohne Permutation, und mit der obigen Erweiterung sogar für nicht–quadratische Matrizen.

```
function [Q R] = ONV( A )
%ONV Compute QR decomposition of A based on Schmidt
%orthonormalization. Currently works on real input only!
%Fails in the case of singular A.
[n m]=size(A);
Q=zeros(n,m);
```

Listing 3.9: QR–Zerlegung nach Schmidt (QR/ONV.m)

Klicken für den Quellcode von QR/ONV.m

```
function A = testONV( Q,R )
%TESTONV compute A from its Schmidt decomposition
A=Q*R;
end
```

Listing 3.10: Berechnung von A aus der QR–Zerlegung nach Schmidt (QR/testONV.m)

Klicken für den Quellcode von QR/testONV.m

```
function B = solveONV( Q,R,B )
%SOLVEONV Use ONV decomposition for solution of linear equations
[n n]=size(Q);
B=Q'*B;
for i=n:-1:1
    for k=i+1:n
```

Listing 3.11: Berechnung der Lösung eines LGS aus der QR–Zerlegung nach Schmidt (QR/solveONV.m)

Klicken für den Quellcode von QR/solveONV.m

```
function [ output_args ] = doitONV( n )
%DOITONV solve random equation of size n
if (nargin<1)
    n=3;
end
A=rand(n);
```

Listing 3.12: Lösung eines zufälligen LGS nach Schmidt (QR/doitONV.m)

Klicken für den Quellcode von QR/doitONV.m

Die ganze Rechnung hat aber einen Haken. Nehmen wir an, die Dimension von A sei sehr groß. Bei der Berechnung von $\tilde{q}_k$ ziehen wir von a_k die Anteile in den orthogonalen Richtungen q_j, $j < k$, ab. Dabei wird die Norm von a_k jedes Mal kleiner. Die Norm von $\tilde{q}_k$ ist also kleiner, häufig viel kleiner, als die von a_k (es sei denn, die Matrix A war schon unitär). Dies kann nur durch die Subtraktionen entstanden sein, also durch Auslöschung. Auslöschung ist für das Schmidtsche Orthogonalisierungsverfahren also unvermeidlich, es ist instabil.
Wir brauchen also ein alternatives Verfahren und nutzen dazu die Householder–Spiegelungen. Gegeben sei eine $(m \times n)$–Matrix A. Wie bei der LR–Zerlegung wollen wir im ersten Schritt die Matrix mit einer (jetzt unitären) Matrix von links multiplizieren, so dass die Elemente unterhalb der Hauptdiagonalen in der ersten Spalte eliminiert werden. Wir suchen also eine unitäre Matrix Q_1, so dass

$$Q_1 A = \begin{pmatrix} \alpha_1 & * \\ 0 & * \\ \vdots & \vdots \\ 0 & * \end{pmatrix}$$

Seien $a_1, \ldots, a_n$ die Spalten von A. Da unitäre Abbildungen längenerhaltend sind, gilt schon mal $||a_1||_2 = ||Q_1 a_1||_2 = |\alpha_1|$. Wir wollen Q als Spiegelungsmatrix wählen und betrachten das Problem zunächst im $\mathbb{R}^2$ (Abbildung 3.1). Offensichtlich gibt es zwei Möglichkeiten, einen gegebenen Vektor auf ein Vielfaches des Einheitsvektor abzubilden, nämlich die Spiegelung an den Geraden

$$a_1 \pm ||a_1|| e_1$$

mit den Normalvektoren

$$a_1 \mp ||a_1|| e_1.$$

Wir lassen die Wahl des Vorzeichens zunächst offen und setzen

$$\sigma \in \{1, -1\},\ \alpha_1 = \sigma ||a_1||_2 = \sqrt{a_1^t a_1},\ v_1 = a_1 - \alpha_1 e_1.$$

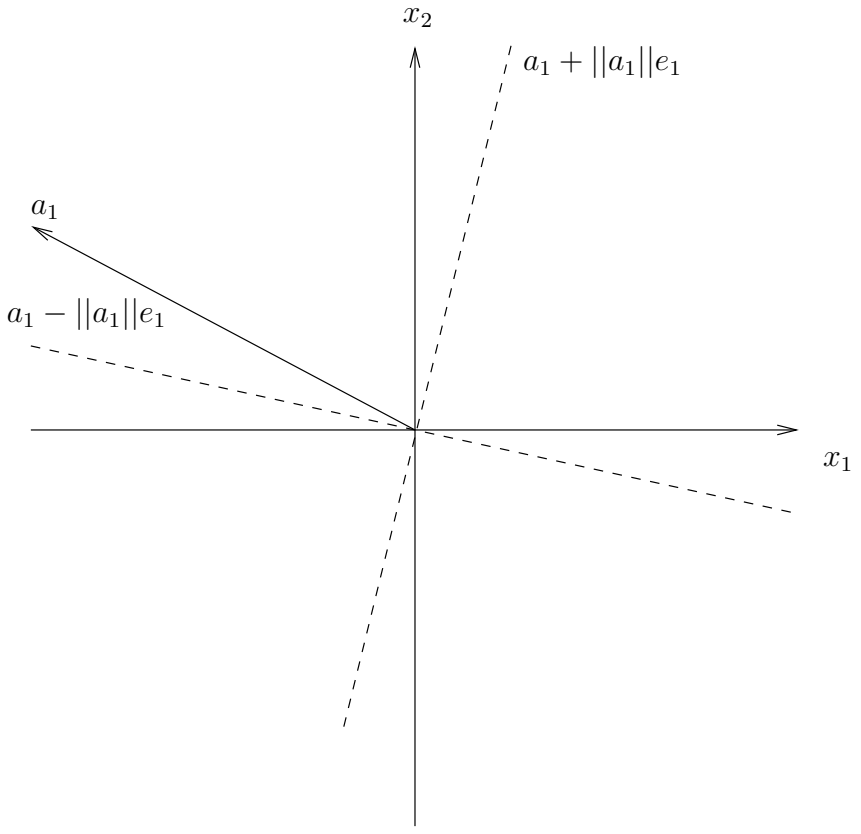

Abbildung 3.1: a_1 kann auf $||a_1|| e_1$ und $-||a_1|| e_1$ gespiegelt werden.

Wie erwartet gilt

$$\begin{aligned} Q(v_1)a_1 &= (I - 2\frac{v_1 v_1^t}{v_1^t v_1})a_1 \\ &= a_1 - 2\frac{(a_1 - \alpha_1 e_1)^t a_1}{(a_1 - \alpha_1 e_1)^t (a_1 - \alpha_1 e_1)}(a_1 - \alpha_1 e_1) \\ &= a_1 - 2\frac{\alpha_1^2 - \alpha_1 (a_1)_1}{\alpha_1^2 - 2\alpha_1 (a_1)_1 + \alpha_1^2}(a_1 - \alpha_1 e_1) \\ &= \alpha_1 e_1. \end{aligned}$$

Der Spaltenvektor a_1 wird durch die Linksmultiplikation mit $Q(v_1)$ also auf ein Vielfaches des Einheitsvektors abgebildet. Mit $A^{(1)} = A$ steht in $Q(v_1)A^{(1)}$ in der ersten Spalte unterhalb der Hauptdiagonalen 0. Wir streichen nun die erste Zeile und Spalte von $Q(v_1)A^{(1)}$ und erhalten eine neue Matrix $A^{(2)}$. Falls noch Zeilen und Spalten übrigbleiben, führen wir dieselbe Rechnung auf der neuen Matrix $A^{(2)}$ durch. Wir erhalten eine Matrix $Q(v_2)'$, so dass $Q(v_2)A^{(2)}$ in der ersten Spalte unterhalb der Hauptdiagonalen verschwindet.
Wir ergänzen $Q(v_2)'$ in der linken oberen Ecke mit einer Einheitsmatrix zu einer $(m \times m)$–Matrix $Q(v_2)$ und erhalten damit

$$\begin{aligned} Q(v_2)Q(v_1)A &= \begin{pmatrix} 1 & 0 \\ 0 & Q(v_2)' \end{pmatrix} Q(v_1)A^{(1)} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & Q(v_2)' \end{pmatrix} \begin{pmatrix} \alpha_1 & * \\ 0 & A^{(2)} \end{pmatrix} \\ &= \begin{pmatrix} \alpha_1 & * & * \\ 0 & \alpha_2 & * \\ 0 & 0 & * \end{pmatrix} \end{aligned}$$

Sei $k = \min(n, m)$. Wir führen diese Schritte insgesamt k-mal durch und streichen dabei in Schritt i $(i-1)$ Zeilen und Spalten. Für $i = k$ bleibt nichts übrig, und es gilt

$$Q(v_k) \cdots Q(v_2)Q(v_1)A =: R$$

und $(m \times n)$–Matrix mit $R_{ij} = 0$ für $j < i$.
Wir multiplizieren noch von links mit den Transponierten und erhalten insgesamt

$$A = \underbrace{Q(v_1)^t \cdots Q(v_k)^t}_{=:Q} R.$$

Dies ist eine Zerlegung von A in eine unitäre Matrix Q und eine Matrix mit 0 unterhalb der Hauptdiagonalen. Häufig wird auch diese QR–Zerlegung von A genannt.
Um mit der obigen Definition konsistent zu bleiben: Für $m > n$ sind die Zeilen $n+1$

bis m von R leer. In diesem Fall können wir wir die letzten $m-n$ Spalten von Q und die letzten $m-n$ Zeilen von R ändern, ohne dass sich QR ändert. Dann erhalten wir in jedem Fall eine QR–Zerlegung in der oben definierten Form.
Dieser Algorithmus versagt, falls $v_k = 0$ für $\sigma = 1$ **und** $\sigma = -1$. Dann ist aber die erste Spalte von $A^{(k)}$ komplett 0, und die gewünschte Dreiecksform ist bereits hergestellt. Wir können also für $Q(v_k)$ eine beliebige unitäre Matrix wählen, üblicherweise $Q(v_k) = I$. Der Householder–Algorithmus liefert also immer zwei Matrizen Q und R mit $A = QR$.
Zur Wahl von σ: Natürlich wählen wir σ so, dass bei der Berechnung von $a_1 - \alpha_1 e_1$ keine Auslöschung auftritt. Dies ist dann der Fall, wenn $\sigma = -\operatorname{sgn}(a_{11})$.
Dieser Algorithmus zur Herstellung der QR–Zerlegung heißt **Householder**–Algorithmus.
Es gilt

Satz 3.14 *(***Stabilität des Householder–Algorithmus***)*
Der Householder–Algorithmus ist stabil.

(ohne Beweis).
Bemerkung:

1. Der Aufwand zur Berechnung der QR–Zerlegung mit Hilfe der Householder–Matrizen beträgt $mn^2 - n^3/3 + O((n+m)^2)$ für $m \geq n$ und ist damit für $n = m$ doppelt so hoch wie bei der LR–Zerlegung ($n^3/3$ vs. $2n^3/3$) (Übungen).

2. Sei $A = QR$ und A invertierbar. Dann kann die Lösung von $Ax = b$ berechnet werden mit
$$Ax = b \Longleftrightarrow QRx = b \Longleftrightarrow Rx = Q^t b$$
und Rückwärtseinsetzen.

3. Üblicherweise wird die Matrix Q weder abgespeichert noch überhaupt ausgerechnet, sondern nur die Vektoren v_k. Diese werden zusammen mit der rechten oberen Dreiecksmatrix R in die Originalmatrix A geschrieben. Die Hauptdiagonale mit den α_k kommt dabei in einen Extra–Vektor. Qw für einen Vektor w wird dann über die Definition von Q mit Hilfe der v_k berechnet, was nach den Vorbemerkungen sehr schnell geht.

4. Sei $A = QR$. Dann ist
$$A^t A = R^t Q^t Q R = R^t R.$$
$R^t R$ ist also Cholesky–Zerlegung von $A^t A$.

```
function [A, alphavec] = householder( A )
%HOUSEHOLDER compute QR decomposition of A by householder
%operations. R always has A size (different from script).
[m n]=size(A);
mi=min(n,m);
alphavec=zeros(mi,1);
```

Listing 3.13: QR–Zerlegung nach Householder (QR/householder.m)

Klicken für den Quellcode von QR/householder.m

```
function A = householdertest( QR, alphavec )
%HOUSEHOLDERTEST
%compute A from its QR decomposition
%Unefficient. Use computeQR to compute Q and R,
%Then compute its product.
[m n]=size(QR);
```

Listing 3.14: Berechnung von A aus seiner Householderzerlegung (QR/householdertest.m)

Klicken für den Quellcode von QR/householdertest.m

```
function B = householdersolve( QR, alphavec, B )
%HOSUEHOLDERSOLVE
[m n]=size(QR);
B=applyQt(QR,B);
for i=n:-1:1
    for k=i+1:n
```

Listing 3.15: Berechnung der Lösung eines LGS aus der Householder–Zerlegung (QR/householdersolve.m)

Klicken für den Quellcode von QR/householdersolve.m

```
function [ output_args ] = doithouseholder( m,n )
%DOITHOUSEHOLDER Solve random linear equation with householder
if (nargin<1)
    m=3;
end
```

```
if (nargin<2)
```

Listing 3.16: Lösung eines zufälligen LGS mit der Householderzerlegung (QR/doithouseholder.m)

Klicken für den Quellcode von QR/doithouseholder.m

Satz 3.15 *(Eindeutigkeit der QR–Zerlegung Sei $A \in \mathbb{R}^{(n,n)}$ invertierbar. Seien $A = QR = Q'R'$ zwei QR–Zerlegungen von A. Dann gibt es eine $(n \times n)$–Diagonalmatrix D mit $|D_{ii}| = 1$, $i = 1 \dots n$, $Q = Q'D$, $R = D^{-1}R'$. Falls $R_{ii} > 0$ und $R'_{ii} > 0$, $i = 1 \dots n$, so ist $R = R'$ und $Q = Q'$.*

Beweis: Übungen. □

Zum Abschluss sollte man einmal eine größere QR–Zerlegung von Hand rechnen, um sich wirklich im Klaren zu sein, wie das funktioniert. Im Anhang findet sich ein durchgerechnetes Beispiel für eine symbolische 7x5–Matrix.
7x5–QR–Zerlegung

Satz 3.16 *(Hessenbergform)*
Sei A eine $n \times n$–Matrix. Dann gibt es eine unitäre $(n \times n)$–Matrix Q, so dass für $H := QAQ^t$ gilt

$$H_{ik} = 0$$

für $i > k + 1$. H heißt dann **Hessenbergmatrix**. *Falls A symmetrisch ist, so ist H sogar eine Bandmatrix der Bandbreite 1.*
H ist ähnlich zu A, besitzt also dieselben Eigenwerte. Die QR–Zerlegung von H kann schnell berechnet werden.

Beweis: Wir streichen die erste Zeile von A und erhalten eine Matrix $A^{(1)}$. Wir führen darauf einen Householderschritt aus. Dies liefert eine unitäre $(n-1, n-1)$–Matrix Q'_1. Wir ergänzen wie bei Householder diese Matrix zu einer unitären (n, n)–Matrix Q_1. Nach Konstruktion ist dann $Q_1 A$ eine Matrix mit (möglicherweise) von Null verschiedenen Elementen in der ersten und zweiten Zeile der ersten Spalte, darunter stehen nur Nullen. Das ist gerade die geforderte Form. Rechtsmultiplikation mit Q_1^t zerstört dies nicht, d.h. $Q_1 A Q_1^t$ hat in der ersten Spalte die richtige Form.
Wir streichen nun die erste Spalte und Zeile von A und führen den Algorithmus wie bei Householder rekursiv durch. Dies liefert eine Folge von Matrizen $Q_1 \dots Q_{n-1}$, und es gilt

$$H = \underbrace{Q_{n-1} Q_{n-2} \dots Q_1}_{=:Q} A Q_1^t \dots Q_{n-1}^t$$

ist Hessenbergmatrix.
Beweis zum Aufwand: In den Übungen. □

Bemerkung: Falls man auf die Idee kommt, einfach den normalen Householder–Algorithmus anzuwenden und QAQ^t bildet, erhält man zwar mit QA sogar eine rechte obere Dreiecksmatrix, die Rechtsmultiplikation mit Q^t zerstört diese Form aber wieder.

```
function [ A, Q ] = hessenberg( A )
%HESSENBERG Compute Hessenberg form of a matrix.
%Assume A is quadratic.
%Q is the true matrix Q, not in vector form.
[n n]=size(A);
Q=eye(n);
```

Listing 3.17: Hessenbergform einer Matrix (QR/hessenberg.m)

Klicken für den Quellcode von QR/hessenberg.m

```
function A = hessenbergtest( H,Q )
%HESSENBERGTEST Compute A from its Hessenberg representation
A=Q'*H*Q;
end
```

Listing 3.18: Berechnung einer Matrix aus ihrer Hessenbergform (QR/hessenbergtest.m)

Klicken für den Quellcode von QR/hessenbergtest.m

```
function  [A alphavec] = QRhessenberg( A )
%QRHESSENBERG compute QR decomposition of Hessenberg matrix.
%QR is stored as in householder.
%v should be computed directly.
[m n]=size(A);
mi=min(n,m)-1;
```

Listing 3.19: Berechnung der QR–Zerlegung einer Hessenbergmatrix (QR/QRhessenberg.m)

Klicken für den Quellcode von QR/QRhessenberg.m

```
function [ output_args ] = QRhessenbergtest( input_args )
%QRHESSENBERGTEST Summary of this function goes here
%   Detailed explanation goes here


end
```

Listing 3.20: Berechnung einer Hessenbergmatrix aus ihrer QR–Zerlegung (QR/QRhessenbergtest.m)

Klicken für den Quellcode von QR/QRhessenbergtest.m

```
function doithessenberg( n )
%DOITHESSENBERG Compute Hessenberg form of random matrix
if (nargin<1)
    n=3;
end
A=rand(n);
```

Listing 3.21: Test der Hessenbergzerlegungen (QR/doithessenberg.m)

Klicken für den Quellcode von QR/doithessenberg.m

3.4 Übersicht: Direkte Lösung von LGS

	LR–Zerlegung	Cholesky–Zerlegung	QR–Zerlegung
Existenz	$\exists P : PA = LR$	As.p.d $\Rightarrow A = LL^t$	$A = QR$
Eindeutig	Ja	Ja, falls $L_{ii} > 0$	$Q' = QD$
(A inv.)			$R' = D^{-1}R$
Lsg. LGS	Vorw/Rückw.	Vorw./Rückw.	$Rx = Q^t b$
Aufwand	$n^3/3$	$n^3/6$	$2n^3/3$
Stabil	Ja (Pivot)	Ja	Ja
$n \neq m$	Nein	Nein	Ja

Kapitel 4

Über- und unterbestimmte Gleichungssysteme

Wir haben bereits in der Einleitung gesehen, dass im allgemeinen die Anzahl der Gleichungen und Variablen in einem linearen Gleichungssystem nicht übereinstimmen. Sollen etwa bei einer Landvermessung Positionen bestimmt werden, so macht man typischerweise mehr Messungen als notwendig, um Messfehler ausgleichen zu können. Das bekannteste Beispiel findet sich bei Gauss, der die Theorie dazu in seinem Buch “Theoria combinationis observationum erroribus minimis obnoxiae” veröffentlichte (die Society for Industrial and Applied Mathematics hat freundlicherweise für die Nicht–Lateiner eine englische Übersetzung veröffentlicht). Der alte 10 DM–Schein erinnerte an diese Arbeit von Gauss.

Abbildung 4.1: 10 DM–Schein (Quelle: Bundesbank)

Klick für Bild scheinvorn
Klick für Bild schein
Klick für Bild scheincrop

Der Hintergrund dazu ist eigentlich ein statistischer, wir werden uns hier ausschließlich auf die numerische Sichtweise konzentrieren (das Ergebnis ist natürlich genau dasselbe, wie in den Übungen gezeigt wird).

4.1 Die Methode der kleinsten Quadrate

Als Hauptbeispiel betrachten wir die **polynomiale Regression**: In einem Experiment werden Messwerte y_i zu Zeiten t_i gemessen, $i = 1 \dots m$. Es wird vermutet, dass die Ergebnisse durch ein Polynom $p \in \mathcal{P}_{n-1}$ beschrieben werden können, also $y_i = p(t_i)$, $i = 1 \dots m$. P_{n-1} ist dabei die Menge der Polynome vom Grad $\leq n-1$. Zu bestimmen ist das Polynom p (Interpolations–/Approximationsaufgabe). Es gilt zunächst

Satz 4.1 *(Polynominterpolation)*
Seien $t_i \in \mathbb{R}$ paarweise verschieden, und $y_i \in \mathbb{R}$, $i = 1 \dots m$. Dann gilt:

1. *Falls $n = m$, so gibt es genau ein Polynom p in $\mathcal{P}_{n-1}$ mit $p(t_i) = y_i$, $i = 1 \dots m$.*
2. *Falls $n > m$, so gibt es unendlich viele Polynome p in $\mathcal{P}_{n-1}$ mit $p(t_i) = y_i$, $i = 1 \dots m$.*
3. *Falls $n < m$, so gibt es höchstens ein, im Allgemeinen aber gar kein Polynom in $\mathcal{P}_{n-1}$ mit $p(t_i) = y_i$, $i = 1 \dots m$.*

Beweis:
Für alle

$$a \in \mathbb{R}^n,\ a = (a_0, \dots, a_{n-1})^t$$

sei $p_a(t)$ das Polynom mit den Koeffizienten a_k, also

$$p_a(t) = a_0 + a_1 t + \dots + a_{n-1} t^{n-1}.$$

Weiter sei $A \in \mathbb{R}^{m \times n}$ mit $A_{ik} = t_i^{k-1}$, $y = (y_1, \dots, y_m)^t$. Dann ist p_a genau dann Lösung der Aufgabe, wenn

$$Aa = \begin{pmatrix} t_1^0 & t_1^1 & \cdots & t_1^{n-1} \\ \vdots & \vdots & & \vdots \\ t_n^0 & t_n^1 & \cdots & t_m^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = y.$$

A heißt **Vandermondematrix**. Sei $n = m$. Wir zeigen, dass dann A injektiv und damit invertierbar ist. Sei also $a \in \mathbb{R}^n$ mit $Aa = 0$. Wegen $0 = (Aa)_i = p_a(t_i)$ hat p_a damit die m Nullstellen $t_1, \dots, t_m$. Da p_a ein Polynom vom Grad $\leq m-1$ ist, ist nach dem Fundamentalsatz der Algebra p_a das Nullpolynom, und damit $a = 0$, also besteht der Kern von A nur aus 0. Damit ist A invertierbar, und die Interpolationsaufgabe mit $n = m$ hat genau eine Lösung.
Falls $n > m$, so gibt es zu wenige Messungen. Der Rang von A ist m, daraus folgt der Satz.

Falls $m > n$, so gibt es zu viele Gleichungen (Messungen). Der Rang von A ist n. Falls die Messungen exakt sind, so kann man einfach ein Subset von m Gleichungen wählen und die eindeutige Lösung bestimmen. Falls die Messungen oder der polynomiale Zusammenhang nicht exakt sind, gibt es gar keine Lösung. □

Vorlesungsnotiz: 4. November 2012

Die hier implizierte Idee, falls man zu viele Messungen hat, diese einfach wegzuwerfen, ist natürlich nicht praxistauglich. Wir müssen davon ausgehen, dass unsere Werte nicht exakt sind, und wollen mit Hilfe der zusätzlichen Messungen diese Fehler korrigieren.
Für eine numerische Lösung müssen wir also im einen Fall den Lösungsbegriff erweitern, damit überhaupt eine Lösung existiert, im anderen Fall aus den vielen vorhandenen Lösungen eine auswählen.
Wir betrachten den Fall $n = 2$, wir vermuten also einen linearen Zusammenhang. Dann unterscheiden wir

1. $m < n$, also $m = 1$: Jede Gerade, die durch den Punkt (t_1, y_1) geht, ist Lösung der Aufgabe (**unterbestimmter Fall**).

2. $m = n$, also $m = 2$: Die durch die Punkte (t_1, y_1) und (t_2, y_2) definierte Gerade ist Lösung der Aufgabe (**eindeutig bestimmter Fall**).

3. $m > n$, also $m > 2$: Es gibt keine Lösung der Aufgabe (es sei denn, die Messpunkte liegen zufällig tatsächlich auf einer gemeinsamen Geraden). Wir modifizieren unsere Aufgabe und suchen statt dessen eine Gerade, die die gegebenen Punkte möglichst gut approximiert (**überbestimmter Fall**).

```
function beispielpolyregr
%BEISPIELPOLYREGR Beispiele zur Ausgleichsgeraden
for i=1:11
    x1(i)=i-6;
    y1(i)=x1(i)+(rand-0.5)*2;
end
```

Listing 4.1: Unterbestimmtes / eindeutig bestimmtes / überbestimmtes System (ueberbest/beispielpolyregr.m)

Klicken für den Quellcode von ueberbest/beispielpolyregr.m

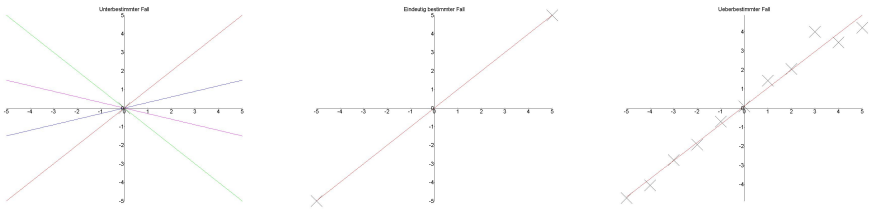

Abbildung 4.2: Beispiel: Polynomiale Regression. Unterbestimmt, bestimmt, überbestimmt.

Klick für Bild unterbestimmt
Klick für Matlab Figure unterbestimmt
Klick für Bild bestimmt
Klick für Matlab Figure bestimmt
Klick für Bild ueberbestimmt
Klick für Matlab Figure ueberbestimmt

Im dritten Fall suchen wir ein Polynom $p_a \in \mathcal{P}_{n-1}$, so dass der Abstand der Punkte (t_i, y_i) und $(t_i, p_a(t_i)) = (t_i, (Aa)_i)$, also $||y - Aa||$, insgesamt möglichst klein ist. Wir haben also das Problem der Lösung von

$$Aa = y \iff ||Aa - y|| = 0$$

ersetzt durch die Bestimmung von

$$\arg\min_a ||Aa - f||.$$

Wenn $||Aa - f|| = 0$ nicht möglich ist, dann soll es also zumindest möglichst klein sein. Für invertierbares A sind diese Lösungen natürlich gleich. Wir erhalten eine echte Erweiterung des Lösungsbegriffs für den Fall, dass eigentlich keine Lösung existiert.
Wir wählen nun zur Abstandsmessung die euklidische Norm, im Kapitel über Approximation werden wir auch andere Normen betrachten.

Definition 4.2 *(**kleinste Quadrate–Lösung, least squares solution**) Sei $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. $x \in \mathbb{R}^n$ heißt kleinste Quadrate–Lösung von $Ax = b$ genau dann, wenn*

$$||Ax - b||_2^2 \leq \min_{y \in \mathbb{R}^n} ||Ay - b||_2^2.$$

Bemerkung: Wir machen keine Voraussetzungen an m, n oder den Rang von A.

Bemerkung: Falls $Ax = b$ Lösungen besitzt, so sind genau diese auch die kleinste Quadrate–Lösungen.
Bemerkung: Für $m \geq n$ und das Interpolationsproblem heißen die durch die kleinste Quadrate–Lösungen von $Ax = b$ definierten Polynome Ausgleichspolynome bzw. Ausgleichsgeraden ($n = 2$).
Leider hilft uns diese Definition nicht bei der Berechnung der kleinsten Quadrate–Lösung. Wir geben daher eine zur Definition äquivalente Bedingung an. Sei im Folgenden immer $A \in \mathbb{R}^{m \times n}$. Wir beschränken uns hier der Einfachheit halber wieder auf reelle Matrizen, obwohl alles sofort auch ins Komplexe übertragbar ist. Wir erinnern noch einmal an

1. die Definitionen

$$\text{Ker}(A) = \{x \in \mathbb{R}^n | Ax = 0\}\,,\ \text{Bild}(A) = \{Ax | x \in \mathbb{R}^n\}$$

und

$$\text{rang}(A) = \dim Bild(A).$$

2. den Rangsatz

$$\text{rang}(A) = \text{rang}(A^t).$$

3. den Dimensionssatz

$$\dim \text{Ker}(A) + \dim \text{Bild}(A) = n.$$

Lemma 4.3 *Sei $A \in \mathbb{R}^{m \times n}$.*

1. $\mathbb{R}^m = \textit{Bild}(A) \oplus \textit{Ker}(A^t)$ *(orthogonale Summe).*
2. $\textit{Ker}(A^t A) = \textit{Ker}(A)$.

Beweis:

1. Sei $x \in \text{Ker}(A^t) \subset \mathbb{R}^m$, $y \in \mathbb{R}^n$ beliebig.

$$A^t x = 0 \Rightarrow 0 = (A^t x, y) = (x, Ay)$$

also $\text{Ker}(A^t) \subset \text{Bild}(A)^\perp$. Sei nun $x \in \text{Bild}(A)^\perp$. Dann gilt

$$0 = (x, AA^t x) = (A^t x, A^t x)$$

und damit $x \in \text{Ker}(A^t)$, insgesamt also $\text{Ker}(A^t) = \text{Bild}(A)^\perp$.

2. Sei $A^tAx = 0$, dann gilt

$$0 = (A^tAx, x) = (Ax, Ax)$$

und damit schon $Ax = 0$.

□

Satz 4.4 *(**Gauss–Normalgleichung**)*
Sei $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. $x \in \mathbb{R}^n$ ist genau dann kleinste Quadrate–Lösung von $Ax = b$, falls

$$A^tAx = A^tb.$$

Diese Gleichung heißt Gausssche Normalgleichung. Die Menge der kleinste Quadrate–Lösungen ist nicht leer.

Beweis: Nach dem Lemma gibt es $b_1, b_2 \in \mathbb{R}^m$, $b_1 = Az$, $z \in \mathbb{R}^n$, $A^tb_2 = 0$ mit $b = b_1 + b_2$. Sei $x \in \mathbb{R}^m$. Dann gilt

$$||Ax - b||_2^2 = ||\underbrace{Ax - b_1}_{\in \text{Bild}(A)} - \underbrace{b_2}_{\in \text{Ker}(A^t)}||_2^2 = ||Ax - b_1||_2^2 + ||b_2||_2^2 \geq \underbrace{||Az - b_1||_2^2}_{=0} + ||b_2||_2^2.$$

z ist also kleinste Quadrate–Lösung. Weiter ist ein $x \in \mathbb{R}^n$ genau dann kleinste Quadrate–Lösung, wenn $Ax = b_1 = Az$, also

$$A(x - z) = 0 \Longleftrightarrow 0 = A^tA(x - z) = A^tAx - A^tAz = A^tAx - A^t(b_1 + \underbrace{b_2}_{\in \text{Ker}(A^t)}),$$

also $A^tAx = A^tb$.
Wir skizzieren noch die Idee zu einem zweiten Beweis (nur eine Richtung). Sei x eine kleinste Quadrate–Lösung, und $y \in \mathbb{R}^n$ beliebig. Dann hat

$$g(\lambda) = ||A(x + \lambda y) - b||_2^2$$

ein lokales Minimum bei $\lambda = 0$, es gilt also

$$\begin{aligned} 0 = g'(0) &= ((Ax - b + \lambda Ay)^t(Ax - b + \lambda Ay))'(0) \\ &= 2(Ax - b, Ay) + 2\lambda(Ay, Ay)|_{\lambda=0} \\ &= 2(A^t(Ax - b), y). \end{aligned}$$

Wir wählen nun $y = A^t(Ax - b)$ und damit gilt

$$A^tAx = A^tb.$$

□

Bemerkung: Seien x_1, x_2 zwei kleinste Quadrate–Lösungen. Dann gilt

$$x_1 - x_2 \in \mathsf{Ker}(A^t A) = \mathsf{Ker}(A).$$

Die kleinste Quadrate–Lösung ist also genau dann **eindeutig**, wenn A den Rang n hat. Im Allgemeinen ist sie es **nicht**.

Beispiel 4.5

1. *Eine feste Länge L wird m–mal gemessen mit Ergebnissen l_1 bis l_m. Das zugehörige überbestimmte Gleichungssystem lautet*

$$\begin{array}{rcl} L & = & l_1 \\ L & = & l_2 \\ & \vdots & \\ L & = & l_m \end{array} \quad \Rightarrow \quad \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} L = \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_m \end{pmatrix}.$$

   Für die kleinste Quadrate–Lösung L erhalten wir

$$mL = (1, \ldots, 1) \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = (1, \ldots, 1) \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_m \end{pmatrix} = l_1 + l_2 + \ldots + l_m$$

   und damit

$$L = \frac{\sum_{i=1}^{m} l_i}{m},$$

   also, nicht sehr überraschend, den Mittelwert der l_i.

2. *Zu Zeitpunkten t_i werden die Messwerte y_i gemessen, $i = 1 \ldots 4$.*

t_i	-2	0	1	1
y_i	-2	-4	4	6

   Es wird ein linearer Zusammenhang der Form $y(t) = at + b$ vermutet. Wir bestimmen die Ausgleichsgerade. Das überbestimmte Gleichungssystem lautet

$$\begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} -2 \\ -4 \\ 4 \\ 6 \end{pmatrix}.$$

Die Normalgleichung lautet

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -2 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -2 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} -2 \\ -4 \\ 4 \\ 6 \end{pmatrix}$$

also

$$\begin{pmatrix} 4 & 0 \\ 0 & 6 \end{pmatrix} \begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} 4 \\ 14 \end{pmatrix}$$

und damit erhalten wir die Ausgleichsgerade $(7/3)x + 1$.

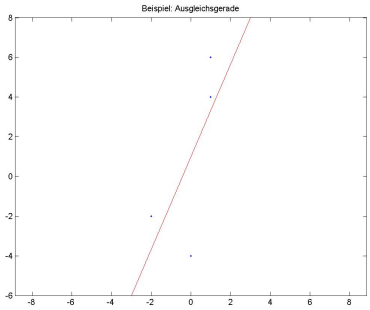

Abbildung 4.3: Beispiel zur Ausgleichsgeraden

Klick für Bild ausgleich
Klick für Matlab Figure ausgleich

```
function beispielaus
%BEISPIELAUS Beispiel Ausgleichsgerade aus Skript
addpath('../Cholesky');
A=[1 -2; 1 0; 1 1; 1 1];
b=[-2;-4;4; 6];
b1=A'*b;
```

Listing 4.2: Lösung eines überbest. LGS (ueberbest/beispielaus.m)

Klicken für den Quellcode von ueberbest/beispielaus.m

Sei A *die* $m \times n$*–Nullmatrix. Dann ist jedes* $x \in \mathbb{R}^n$ *kleinste–Quadrate–Lösung von* $Ax = b$*, denn*

$$A^t Ax = 0 = A^t b.$$

4.2 Die Minimum Norm–Lösung

Im allgemeinen ist die kleinste Quadrate–Lösung nicht eindeutig. Wir wählen in diesen Fällen eine spezielle aus, nämlich die mit kleinster Norm. Dies führt zur Definition der Minimum–Norm–Lösung.

Definition 4.6 *(**Minimum Norm–Lösung**, **verallgemeinerte Lösung**, **Moore–Penrose–Lösung**)*
Sei $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. $x^+ \in \mathbb{R}^n$ heißt Minimum Norm–Lösung von $Ax = b$ genau dann, wenn gilt

1. *x^+ ist kleinste Quadrate–Lösung von $Ax = b$.*
2. *x^+ hat unter allen kleinste Quadrate–Lösungen von $Ax = b$ die kleinste Norm, d.h.*

$$||x^+|| \leq ||x|| \forall x : x \text{ ist kleinste Quadrate–Lösung von } Ax = b$$

.

Bemerkung: Wir machen keine Voraussetzungen an m, n oder den Rang von A.
Bemerkung: Für die kleinste Quadrate–Lösung und die Minimum Norm–Lösung gibt es eine einfache geometrische Deutung. Zunächst ist klar:

$$Ax = b$$

ist lösbar genau dann, wenn $b \in$ Bild(A). Falls dies nicht gilt, so projizieren wir b orthogonal auf Bild(A), erhalten b_1 und lösen statt dessen

$$Ax = b_1.$$

Die so erhaltenen Lösungen sind die kleinste Quadrate–Lösungen.
Falls es mehrere Lösungen gibt: Offensichtlich trägt der Nullraum von A nichts zur Lösung von $Ax = b$ bei. Also projizieren wir die Lösungen auf den Ker$(A)^\perp (=$ Bild(A^t) und diese Projektion ist eindeutig. Die so erhaltene Lösung ist die Minimum Norm–Lösung.
Diese Definition erlaubt uns noch nicht die direkte Berechnung von x^+. Die geometrische Überlegung legt aber die folgende Beziehung nahe. Wir geben wieder eine zum Optimierungsproblem äquivalente nachrechenbare Bedingung an.

Satz 4.7 *(Berechnung und Eindeutigkeit der Minimum Norm–Lösung)*
Sei $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. x^+ ist genau dann Minimum Norm–Lösung von $Ax = b$, falls gilt

1. $A^tAx^+ = A^tb$.

2. $x^+ \in$ *Bild*(A^t).

Die Minimum Norm–Lösung ist eindeutig bestimmt.

Beweis: Sei x eine kleinste Quadrate–Lösung von $Ax = b$. Nach Nach Lemma 4.3, angewandt auf A^t, ist $x = u + v$, $u \in$ Bild(A^t), $v \in$ Ker(A). Mit $x^+ := u$ (wir wählen also wie in der Vorüberlegung x^+ als Projektion auf Ker$(A)^\perp$) gilt

$$A(x - x^+) = Av = 0$$

und damit

$$A^tAx^+ = A^tA(x^+ + (x - x^+)) = A^tAx = b,$$

denn x ist kleinste Quadrate–Lösung, also ist auch x^+ kleinste Quadrate–Lösung. Sei nun $\overline{x}$ eine weitere kleinste Quadrate–Lösung. Mit der Bemerkung zu Satz 4.4 gilt

$$\overline{x} = x^+ + w,\ w \in \text{Ker}(A)$$

und

$$||\overline{x}||_2^2 = ||x^+ + w||_2^2 = || \underbrace{x^+}_{\in \text{Bild}_{(A^t)}} + \underbrace{w}_{\in \text{Ker}_{(A)}} ||_2^2 = ||x^+||_2^2 + ||w||_2^2 \geq ||x^+||_2^2.$$

Gleichheit bekommen wir genau dann, wenn $w = 0$, also $\overline{x} = x^+ + w = x^+$, die Minimum Norm–Lösung ist also eindeutig. □

Beispiel 4.8

1. *Sei A die $m \times n$–Nullmatrix, $b \in \mathbb{R}^m$ beliebig. Dann erfüllt jedes $x \in \mathbb{R}^n$ die Normalengleichung*

$$A^tAx = A^tb$$

und ist damit kleinste Quadrate–Lösung. Die Minimum Norm–Lösung ist unter diesen die mit kleinster Norm, also $x^+ = 0$. Offensichtlich ist x^+ auch eindeutig bestimmt durch die Bedingung

$$x^+ \in \textit{Bild}(A^t) = \{0\}\,.$$

2. *Wir suchen die Minimum Norm–Lösung des Ausgleichsproblems für eine einzelne Messung (t_1, y_1) und den linearen Ansatz $at + b$. (a, b) ist kleinste Quadrate–Lösung, also muss gelten*

$$\begin{pmatrix} 1 \\ t_1 \end{pmatrix} \begin{pmatrix} 1 & t_1 \end{pmatrix} \begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} 1 \\ t_1 \end{pmatrix} \begin{pmatrix} y_1 \end{pmatrix}$$

also

$$\begin{pmatrix} 1 & t_1 \\ t_1 & t_1^2 \end{pmatrix} \begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} y_1 \\ ty_1 \end{pmatrix}$$

oder

$$b + at_1 = y_1.$$

Da in diesem Fall Lösungen von $Ax = b$ existieren, sind genau diese natürlich auch die kleinste Quadrate–Lösungen, den Ansatz über die Normalengleichung hätten wir uns also sparen können.
Wegen

$$x^+ \in \textit{Bild}(A^t) = \left\{A^t u | u \in \mathbb{R}^m\right\} = \left\{(u, t_1 u)^t | u \in \mathbb{R}\right\}$$

gilt für ein u: $b = u$, $a = t_1 u$ und

$$y_1 = b + at_1 = u + t_1^2 u$$

und damit

$$u = \frac{y_1}{1 + t_1^2}$$

und die gesuchte Gerade ist

$$y(t) = \frac{y_1 t_1}{1 + t_1^2} t + \frac{y_1}{1 + t_1^2}.$$

Insbesondere erfüllt diese Gerade natürlich $y(t_1) = y_1$. Tatsächlich hat diese Gerade eine kleinere 2–Norm auf den Koeffizienten als die eigentlich viel naheliegendere Lösung $y(t) = y_1$.

4.3 Die Pseudoinverse

Falls A maximalen Rang hat (also Rang$(A) = \min(n, m)$). so lässt sich die Minimum Norm–Lösung durch Matrixinversion berechnen.

Satz 4.9 Pseudoinverse, Moore–Penrose–Inverse, verallgemeinerte Inverse
Die Abbildung $A^+ : \mathbb{R}^m \mapsto \mathbb{R}^n$, $A^+ b = x^+$, ist linear. A^+ heißt Pseudoinverse (Moore–Penrose–Inverse, verallgemeinerte Inverse) von A.

1. *Falls $n = m$ und A invertierbar, so gilt $A^+ = A^{-1}$.*

2. *Falls $m > n$ und A injektiv (Rang$(A) = n$), so ist A^tA invertierbar und*

$$A^+ = (A^tA)^{-1}A^t.$$

3. *Falls $m < n$ und A surjektiv (Rang$(A) = m$), so ist AA^t invertierbar und*

$$A^+ = A^t(AA^t)^{-1}.$$

Beweis: Sei $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$.

1. Falls A invertierbar ist (Rang$(A) = m = n$), so ist die einzige kleinste Quadrate–Lösung die eindeutige Lösung von $Ax = b$, also gilt

$$A^+ = A^{-1}.$$

2. Für $m > n$ ist der Zielraum in der Dimension größer als der Urbildraum. A kann also nicht surjektiv sein, aber injektiv. Sei A injektiv, d.h. Rang$(A) = n$. Wegen Ker$(A) =$ Ker(A^tA) ist auch A^tA injektiv, also invertierbar. x^+ erfüllt die Normalengleichung
$$A^tAx^+ = A^tb$$
also
$$x^+ = (A^tA)^{-1}A^tb.$$

3. Für $m < n$ ist der Urbildraum in der Dimension größer als der Zielraum. A kann also nicht injektiv sein, aber surjektiv. Sei A surjektiv, d.h. Rang$(A) = m$. Dann gibt es Lösungen von $Az = b$, und genau diese sind die kleinste Quadrate–Lösungen.
Wegen
$$\text{Rang}(A^t) = \text{Rang}(A) = \dim \text{Bild}(A) = m$$
ist A^t injektiv, also ist AA^t invertierbar. Wegen $x^+ \in$ Bild(A^t) gilt $x^+ = A^tu$ für ein $u \in \mathbb{R}^m$, also
$$b = Ax^+ = AA^tu$$
und damit
$$x^+ = A^tu = A^t(AA^t)^{-1}b.$$

□

Falls die Inverse von A existiert, so gilt $A^+ = A^{-1}$. Die Pseudoinverse hat aber auch für die anderen Fälle einige Gemeinsamkeiten mit der Inversen.

Satz 4.10 *(Rechenregeln der Pseudoinversen)*

1. $(A^+)^+ = A$.
2. $AA^+A = A$.
3. $A^+AA^+ = A^+$.
4. AA^+ *und* A^+A *sind selbstadjungiert.*

Beweis: Übungen. □

Dagegen gilt im allgemeinen nicht $(AB)^* = B^+A^+$. Sei als Beispiel

$$A = (1,0),\ B = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ (AB) = (AB)^+ = (1):\ B^+A^+ = (\frac{1}{2}, \frac{1}{2}) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = (\frac{1}{2}).$$

Leider ist es ungünstig, die Minimum Norm–Lösungen mit 4.9 direkt auszurechnen. Sei als Beispiel A invertierbar und selbstadjungiert. Angenommen, wir berechnen die Minimum Norm–Lösung (in diesem Fall natürlich die Lösung) von $Ax = b$ durch

$$A^tAx = A^tb.$$

Dann müssen wir in der euklidischen Norm mit Fehlerverstärkung in der Größenordnung

$$k_2(A^tA) = k_2(A^2) = k_2(A)^2$$

rechnen (siehe 2.44 und 2.29). Für Matrizen mit hoher Kondition ist aber

$$k_2(A)^2 >> k_2(A),$$

dieser Algorithmus ist also nicht stabil. Generell sollte man die Berechnung von A^tA vermeiden.
Wir nutzen daher zur Berechnung die QR–Zerlegung.

1. Sei $m \geq n$, $A \in \mathbb{R}^{m\times n}$, Rang$(A) = n$ und

   $$A = QR$$

   eine QR–Zerlegung von A, also

   $$Q \in \mathbb{R}^{m\times n} \text{ mit } Q^tQ = I,\ R \in \mathbb{R}^{n\times n} \text{rechte obere Dreiecksmatrix.}$$

A hat maximalen Rang, also auch R, und damit sind R und R^t invertierbar. x^+ ist eindeutig bestimmt durch die Normalengleichung

$$R^tQ^tb = A^tb = A^tAx^+ = R^tQ^tQRx = R^tRx^+.$$

R^t ist invertierbar, also gilt

$$Rx^+ = Q^tb$$

und wir erhalten x^+ durch Rückwärtseinsetzen.

2. Sei $m \leq n$, $A \in \mathbb{R}^{m\times n}$, Rang$(A) = m$. In diesem Fall nutzen wir die QR–Zerlegung von A^t. Sei also

$$A^t = QR$$

eine QR–Zerlegung von A^t,

$$Q \in \mathbb{R}^{n\times m} \text{ mit } Q^tQ = I,\ R \in \mathbb{R}^{m\times m} \text{rechte obere Dreiecksmatrix.}$$

Wie oben sind R, R^t invertierbar. Nach Satz 4.9 gilt

$$x^+ = A^t(AA^t)^{-1}b = QR(R^tQ^tQR)^{-1}b = Q(R^t)^{-1}b = Qy,\ R^ty = b.$$

Wir erhalten y durch Vorwärtseinsetzen und daraus x^+.

Bei dieser Berechnung ändert sich die Kondition in der euklidischen Norm für invertierbare Matrizen nicht. Die Kondition von A berechnet sich aus den Eigenwerten von A^tA, die für R aus den Eigenwerten von R^tR. Es gilt z.B. für Fall 1

$$A^tA = R^tQ^tQR = R^tR,$$

insbesondere sind die Eigenwerte dieselben und damit

$$k_2(A) = k_2(R)$$

und die Fehlerverstärkung bei der Lösung von

$$Ax = b \text{ oder } Rx = Q^tb$$

ist dieselbe.

4.4 Die Singulärwertzerlegung

Eine einfache, wenn auch schwierig zu berechnende, Darstellung der kleinsten Quadrate–Lösungen bekommt man über die Singulärwertzerlegung.
Wir nutzen den Satz, dass eine hermitesche Matrix A eine Orthonormalbasis aus Eigenvektoren besitzt (Satz 2.25), also mit einer unitären Matrix diagonalisierbar ist. Für allgemeine Matrizen gilt das natürlich nicht, aber wir zeigen die Existenz einer abgeschwächten Form, die sehr ähnliche Eigenschaften besitzt.

Satz 4.11 *(**Singulärwertzerlegung, Singular Value Decomposition, SVD**)*
Sei $A \in \mathbb{R}^{m\times n}$. Sei r der Rang von A.

1. *Es gibt Matrizen $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$ mit*

   $$U^tU = I_r,\ V^tV = I_r,\ \Sigma \text{ invertierbare Diagonalmatrix}$$

   und

   $$A = U\Sigma V^t.$$

   Die Spalten von U sind Eigenvektoren von AA^t, die Spalten von V sind Eigenvektoren von A^tA, auf der Hauptdiagonalen von Σ stehen die Wurzeln der von Null verschiedenen Eigenwerte von AA^t und A^tA.

   U, Σ, V heißen Singulärwertzerlegung von A. Die Zahlen σ_k auf der Hauptdiagonalen von Σ heißen Singulärwerte, die Spalten u_k, v_k von U bzw. V heißen Singulärvektoren.

   Bezüglich des Standard–Skalarprodukts gilt

   $$Ax = \sum_{k=1}^{r} \sigma_k(x, v_k)u_k.$$

2. *Für die Pseudoinverse gilt*

   $$A^+ = V\Sigma^{-1}U^t$$

   bzw.

   $$A^+b = \sum_{k=1}^{r} \frac{1}{\sigma_k}(b, u_k)v_k.$$

Beweis:

1. A^tA ist hermitesch und positiv semidefinit, besitzt also eine Orthonormalbasis aus Eigenvektoren $v_1, \ldots, v_n$ zu den der Größe nach geordneten Eigenwerten $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$. Es gilt

   $$\dim \mathsf{Ker}(A^tA) = \dim \mathsf{Ker}(A) = n - \mathsf{Rang}(A) = n - r,$$

also

$$\lambda_1, \ldots, \lambda_r > 0,\ \lambda_{r+1}, \ldots, \lambda_n = 0.$$

Sei

$$\sigma_k = \sqrt{\lambda_k} > 0,\ k = 1 \ldots r,\ \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{pmatrix}.$$

Sei $V \in \mathbb{R}^{n \times r}$ die Matrix mit den Spalten v_k, $k = 1 \ldots r$. Dann gilt

$$A^t A V = V \Sigma^2.$$

Wir setzen nun $U = AV\Sigma^{-1}$. Die Spalten von V sind Eigenvektoren, also nicht 0. Wegen $V\Sigma^2 = A^t AV$ sind also auch die Spalten von AV und damit U nicht 0, und

$$AA^t U = AA^t AV\Sigma^{-1} = AV\Sigma = U\Sigma^2.$$

Die Spalten von U sind also Eigenvektoren von AA^t, wieder zu den Eigenwerten σ_k^2.

$\mathrm{Ker}(A^t A) = \mathrm{Ker(A)} \Rightarrow Av_k = 0$ und $Vv_k = 0, k = r+1 \ldots n$ (v_j ist ONB).

$$\begin{aligned} (U\Sigma V^t - A)(\underbrace{v_1 \ldots v_r}_{=V} v_{r+1} \ldots v_n) &= (AVV^t - A)(Vv_{r+1} \ldots v_n) \\ &= (AV - AV, 0, \ldots, 0) \\ &= 0. \end{aligned}$$

Da $(v_1, \ldots, v_r, v_{r+1}, \ldots, v_n)$ unitär, also insbesondere invertierbar, ist, gilt bereits

$$U\Sigma V^t - A = 0 \Rightarrow A = U\Sigma V^t.$$

Die Darstellung von Ax folgt durch Einsetzen von

$$x = \sum_{k=1}^{n} (x, v_k) v_k.$$

2. Sei $b \in \mathbb{R}^m$ und $y = V\Sigma^{-1}U^t b$, also

$$y = \sum_{k=1}^{r} \frac{1}{\sigma_k} (b, u_k) v_k.$$

Wir müssen zeigen, dass y Minimum–Norm–Lösung ist, also y kleinste Quadrate–Lösung und $y \in \mathrm{Bild}(A^t)$. Wegen

$$v_k = \sigma_k^{-2} A^t A v_k$$

liegen die v_k im Bild von A^t, also auch y.
Es gilt

$$A^t A y = V\Sigma U^t U\Sigma V^t V\Sigma^{-1}U^t b = V\Sigma U^t b = A^t b,$$

also ist y auch kleinste Quadrate–Lösung von $Ax = b$ und damit Minimum Norm–Lösung.

□

Beispiel 4.12

1. *Zu zeigen: AA^+ und A^+A sind symmetrisch.*
   Beweis: Sei $U\Sigma V^t$ die SVD von A. Dann ist z.B.

$$AA^+ = U\Sigma V^t V\Sigma^{-1}U^t = UU^t.$$

2. *Bestimme das Polynom vom Grad kleiner oder gleich $n-1$, das durch (t_1, y_1) geht und die kleinste 2–Norm auf den Koeffizienten aufweist.*
   Lösung: Wir suchen die Minimum Norm–Lösung für die lineare Gleichung

$$\sum_{k=0}^{n-1} a_k t_1^k = y_1.$$

   Sei $L = \sqrt{\sum_{k=0}^{n-1} t_1^{2k}}$. Es gilt

$$A = (t_1^0 \dots t_1^{n-1}) = \underbrace{1}_{=:U} \cdot \underbrace{L}_{=:\Sigma} \cdot \underbrace{(t_1^0 \dots t_1^{n-1})/L}_{=:V^t} = U\Sigma V^t,$$

   wobei U in seiner einzigen Spalte einen Vektor der Norm 1 enthält, Σ eine Diagonalmatrix ist mit Eintrag L und V in der einzigen Spalte einen Vektor der Norm 1 enthält. U, Σ und V erfüllen also die Anforderungen an die Singulärwertzerlegung, und es gilt

$$A^+ = V\Sigma^{-1}U^t = \frac{1}{L^2}\begin{pmatrix} t_1^0 \\ \vdots \\ t_1^{n-1} \end{pmatrix}$$

   im Einklang mit Beispiel 4.8. Die Lösung ist

$$p(t) = \frac{y_1}{L^2}(1 + t_1 t + t_1^2 t^2 + \dots + t_1^{n-1} t^{n-1}).$$

Dieser Satz liefert also eine einfache explizite Darstellung für die Pseudoinverse. Leider ist sie zum Nachweis analytischer Eigenschaften sehr nützlich, für das praktische Rechnen aber unbrauchbar. Um sie zu nutzen, muss zunächst die Singulärwertzerlegung berechnet werden, es müssen also die Eigenwerte und Eigenvektoren von A^tA und AA^t ausgerechnet werden. Aktuell haben wir dazu noch gar keinen Algorithmus. Wir werden einen kennenlernen, der aber eine Serie von QR–Zerlegungen berechnet, also in jedem Fall aufwändiger ist als die direkte Berechnung der kleinste Quadrate–Lösung mit Hilfe der QR–Zerlegung. Aus diesem Grund warnt die klassische Linpack–Bibliothek im Users Guide: We warn the user that although the pseudo–inverse occurs frequently in the literature of various fields, there is seldom any need to compute it explicitly.
Zusätzlich erlaubt die Singulärwertzerlegung, aus der linearen Algebra bekannte Begriffe numerisch zu interpretieren. Der numerische Rang einer Matrix etwa ist definiert als die Anzahl der Singulärwerte, die größer sind als eine Schranke ϵ. Wir betrachten dazu die Matrizen

$$A = \begin{pmatrix} a & a \\ a & a \end{pmatrix}, \tilde{A} = \begin{pmatrix} a+\epsilon & a \\ a & a \end{pmatrix}$$

Diese Matrizen sind numerisch für kleine ϵ nicht unterscheidbar, die eine kann durch Rundung der anderen entstehen, sie haben aber für $a \neq 0$ die Ränge 1 und 2. Der Rang ist also numerisch nicht berechenbar. A hat aber wie oben berechnet einen Singulärwert 0, $\tilde{A}$ hat zwar keinen Singulärwert 0, aber einen sehr kleinen, beide hätten also den gleichen numerischen Rang 1.
Wir halten fest, dass die Berechnung der Eigenwerte positiv (semi–)definiter Matrizen numerisch eine besonders große Rolle spielt (nämlich zur Berechnung der Singulärwertzerlegung).
Zum Abschluss betrachten wir noch eine erweiterte Form der Pseudoinversen, die Tikhonov–Inverse. Nehmen wir an, die Kondition einer Matrix A in der 2–Norm, also der Quotient aus größtem und kleinstem Singulärwert, sei sehr groß (etwa wie in der Matrix $\tilde{A}$ aus dem letzten Beispiel). Mit der Darstellung der Pseudoinversen nach 4.11 ist auch klar, woher dieser Fehler stammt: $\frac{1}{\sigma_k}$ wird groß, also werden Fehler in b dann stark verstärkt, und die Pseudoinverse wird unbrauchbar. In diesen Fällen wird 4.2 leicht modifiziert.

Definition 4.13 *(**Tikhonov–Inverse**)*
Sei $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\gamma > 0$ fest. $x_\gamma^+ \in \mathbb{R}^n$ heißt **Tikhonov–regularisierte Lösung** *von $Ax = b$ genau dann, wenn x_γ^+ das Funktional*

$$g(x) = ||Ax - b||_2^2 + \gamma^2 ||x||_2^2$$

für $x \in \mathbb{R}^n$ minimiert.
Die Abbildung A_γ^+ von b auf x_γ^+ heißt **Tikhonov–Inverse**.

In der Definition wird bereits vorausgesetzt, dass die Tikhonov–regularisierte Lösung eindeutig ist.

Satz 4.14 *(Darstellung der Tikhonov–Inversen)*
Seien A, b, γ wie in 4.13. Dann ist die Tikhonov–regularisierte Lösung von $Ax = b$ eindeutig, die Tikhonov–Inverse also wohldefiniert. Es gilt

$$(A^t A + \gamma^2 I)x_\gamma^+ = A^t b$$

und mit den Bezeichnungen der Singulärwertzerlegung

$$A_\gamma^+ b = V(\Sigma^2 + \gamma^2 I)^{-1}\Sigma U^t b$$

oder

$$A_\gamma^+ b = \sum_{k=1}^{r} \frac{\sigma_k}{\sigma_k^2 + \gamma^2}(b, u_k)v_k.$$

Beweis: Übungen. □

Die Reihendarstellung der Tikhonov–regularisierten Lösung zeigt ihre Auswirkung bei Fehlern in b: Der Parameter $\gamma > 0$ verhindert, dass der fehlerverstärkende Quotient vor dem Skalarprodukt zu groß wird, er begrenzt also den Einfluß von Fehlern auf das Ergebnis. Der Nachteil ist, dass selbst bei korrektem b nicht mehr die exakte Lösung ausgerechnet wird, sondern nur eine Approximation. Die Wahl des Regularisierungsparameters γ ist also entscheidend: Ist γ zu klein oder Null, so werden Fehler zu stark verstärkt. Ist γ zu groß, wird der Approximationsfehler groß.

Korollar 4.15 *(Grenzwert der Tikhonov–Inversen)*
Es gilt

$$A^+ b = \lim_{\gamma \mapsto 0} A_\gamma^+ b.$$

Dies ist eine alternative Definition der Pseudoinversen, die ohne die (sehr heuristische) Idee, Eindeutigkeit durch Minimierung der euklidischen Norm zu erzwingen, auskommt.

Kapitel 5

Iterative Lösung von Gleichungssystemen mit Fixpunktiterationen

Wie bereits erwähnt, spielen die bisher hergeleiteten direkten Verfahren zur Lösung von linearen Gleichungssystemen inzwischen eine untergeordnete Rolle. Der Grund ist, dass typische große Gleichungssysteme heute Milliarden von Unbekannten und Gleichungen beinhalten (z.B. beschrieben in einem zufälligen technischen Report der Universität Cambridge). Eine Lösung mit Hilfe der hergeleiteten Zerlegungen in 10^{27} Rechenoperationen wäre viel zu langsam. Trotzdem ist eine (approximative) Lösung möglich. Der Grund ist, dass die Gleichungssysteme typischerweise eine spezielle Struktur aufweisen, die es auszunutzen gilt. Wir kennen diese Vorgehensweise von Bandmatrizen (Band–LR) und symmetrischen Matrizen (Cholesky–Zerlegung).
Die am häufigsten auftretende Struktur ist die Sparsity oder Dünne Besetzung von Matrizen. Wird etwa bei der Wettervorhersage ein Deutschlandmodell aufgestellt, so wird die aktuelle Wetterlage mit kurzen Zeitschritten fortgeschrieben. Unbekannte sind dann jeweils die Wetterverhältnisse kurze Zeit nach einem bekannten Zustand. Die sind aber notwendig lokal, d.h. das Wetter in Münster wird nicht vom Wetter in Berlin abhängen. Es werden von der großen Zahl an Variablen also nur sehr wenige in einer Gleichung auftreten, fast alle Einträge der Matrix verschwinden. Im oben angegebenen Report wird eine Matrix der Größe $(1.2 \cdot 10^9)^2$ beschrieben, die aber nur $16 \cdot 10^9$ Einträge hat. Dies ist typischerweise bei der Behandlung von partiellen Differentialgleichungen der Fall.
Leider kann die Gauss–Elimination diesen Vorteil nicht angemessen nutzen. Aus A dünn besetzt folgt nämlich leider nicht, dass z.B. L und R aus der Gauss–Elimination dünn besetzt sind.
Stellen wir uns etwa vor, dass in einer Matrix A nur die erste Spalte und erste Zeile sowie die Diagonale ungleich 0 sind. Im ersten Schritt werden dann Vielfache der ersten Zeile auf alle anderen Zeilen addiert, d.h. die Matrix $A^{(2)}$ aus der Gauss–

Elimination ist voll besetzt, d.h. wir haben unseren strukturellen Vorteil bereits im ersten Schritt komplett verloren.

$$A = \begin{pmatrix} * & * & * & \cdots & * \\ * & \ddots & 0 & \dots & 0 \\ * & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ * & 0 & \dots & 0 & * \end{pmatrix} \quad A^{(2)} = \begin{pmatrix} * & * & \cdots & * \\ * & * & \dots & * \\ \vdots & \vdots & & \vdots \\ * & * & \dots & * \end{pmatrix}$$

Die zusätzlich entstehenden Elemente werden als Fill–in bezeichnet. Zur Vermeidung des Fill–ins wurden viele Strategien zur günstigen Anordnung der Zeilen und Spalten einer Matrix entwickelt, etwa die klassischen Algorithmen zur Bandbreitenreduktion von Rosen, Cuthill und McKee (Skript TU Ilmenau), im obigen Beispiel kann man durch Umkehren der Reihenfolge von Zeilen und Spalten das Problem natürlich komplett lösen. Typischerweise versagen diese Verfahren allerdings spätestens bei der Behandlung dreidimensionaler Probleme.
In diesen Fällen verwenden wir iterative Methoden. Sie berechnen eine Folge von Vektoren $x^{(k)}$, die gegen die Lösung x von $Ax = b$ konvergieren. Dies erscheint zunächst unattraktiv: Wir ersetzen einen Algorithmus, der sicher nach bestimmter Zeit die exakte Lösung liefert (im Rahmen der unvermeidbaren Fehler), durch einen, bei dem viele Zwischenlösungen berechnet werden müssen, der an einer Stelle mit einem Restfehler zur Konvergenz abgebrochen werden muss usw. Für solche Algorithmen benötigen wir mindestens drei wichtige Eigenschaften:

1. $x^{(k)}$ sollte einfach berechenbar sein (meist rekursiv aus $x^{(k-1)}$).
2. Die Konvergenz sollte möglichst schnell sein, damit nicht zu viele Zwischenschritte berechnet werden müssen.
3. Wir benötigen eine Abschätzung dafür, wie nah ein Folgeglied $x^{(k)}$ schon an $\overline{x}$ liegt, damit wir wissen, wann unsere Näherung ausreichend ist.

Wir müssen uns für lineare Gleichungen also fragen, welche Operationen an einer dünn besetzten Matrix besonders einfach auszuführen sind. Dies ist vor allem die Matrix–Vektor–Multiplikation, die gerade so viele Rechenoperationen benötigt, wie die Matrix an Einträgen hat. Tatsächlich werden die meisten von uns hergeleiteten Algorithmen nur Matrix–Vektor–Multiplikationen benötigen.
Wir werden als Grundlage den Banachschen Fixpunktsatz beweisen, und daraus erste iterative Methoden für lineare und nichtlineare Probleme herleiten. Im ganzen Kapitel sind die Matrizen A immer quadratisch und invertierbar, wir betrachten zunächst keine über– oder unterbestimmten Gleichungssysteme.

5.1 Der Banachsche Fixpunktsatz

Definition 5.1 *(**kontrahierend, Fixpunkt**)*
Seien X, Y normierte Räume, $D \subset X$.

1. *Eine Funktion*
$$g : D \mapsto Y$$
heißt kontrahierend in D genau dann, wenn eine Konstante $0 < q < 1$ existiert mit
$$||g(x) - g(y)|| \leq q||x - y|| \ \forall x, y \in D.$$
q heißt Kontraktionskonstante (und ist Lipschitzkonstante).

2. *Sei $g : D \mapsto X$. $\overline{x} \in D$ heißt Fixpunkt von g genau dann, wenn*
$$g(\overline{x}) = \overline{x}.$$

Bemerkung: Sei g kontrahierend. Dann ist g (Lipschitz–) stetig.
Beweis: Sei x_n eine gegen x konvergente Folge, dann gilt

$$||g(x_n) - g(x)|| \leq q||x_n - x|| \mapsto 0.$$

□

Satz 5.2 *(**Banachscher Fixpunktsatz**)*
Sei X ein vollständiger normierter Raum (Banachraum). Sei $\emptyset \neq D \subset X$ abgeschlossen, d.h. jede Cauchyfolge in D konvergiert in D.
Sei $g : D \mapsto D$ kontrahierend in D. Dann hat g genau einen Fixpunkt.

Beweis: Sei $q < 1$ Kontraktionskonstante von g. Seien zunächst x und y zwei Fixpunkte von g. Dann gilt

$$||x - y|| = ||g(x) - g(y)|| \leq q||x - y||,$$

also $x = y$ wegen $q < 1$. Damit ist der Fixpunkt eindeutig.
Die Existenz zeigen wir konstruktiv und geben eine konvergente Folge an, deren Grenzwert der Fixpunkt ist. Sei $x^{(0)} \in D$ beliebig. Wir definieren in D die Folge $x^{(k)}$ durch
$$x^{(k+1)} = g(x^{(k)}).$$

$x^{(k)}$ heißt **Fixpunktiteration**.
g ist kontrahierend, also gilt mit der Definition von $x^{(k)}$

$$\begin{aligned} ||x^{(k+1)} - x^{(k)}|| &= ||g(x^{(k)}) - g(x^{(k-1)})|| \\ &\leq q||x^{(k)} - x^{(k-1)}|| \\ &\leq q^2||x^{(k-1)} - x^{(k-2)}|| \\ &\vdots \\ &\leq q^k||x^{(1)} - x^{(0)}||. \end{aligned}$$

Sei $\epsilon > 0$ beliebig und M so groß, dass

$$\frac{q^M}{1-q}||x^{(1)} - x^{(0)}|| \leq \epsilon.$$

Seien $l, k > M$ und ohne Einschränkung $l \geq k$. Dann gilt

$$\begin{aligned} ||x^{(l)} - x^{(k)}|| &\leq \underbrace{||x^{(l)} - x^{(l-1)}||}_{\leq q^{l-1}||x_1 - x_0||} + \underbrace{||x^{(l-1)} - x^{(l-2)}||}_{\leq q^{l-2}||x_1 - x_0||} + \ldots + \underbrace{||x^{(k+1)} - x^{(k)}||}_{\leq q^k||x_1 - x_0||} \\ &\leq q^k \sum_{j=0}^{l-k-1} q^j ||x^{(1)} - x^{(0)}|| \\ &\leq \frac{q^k}{1-q}||x^{(1)} - x^{(0)}|| \qquad (5.1) \\ &\leq \epsilon \end{aligned}$$

nach Wahl von k und M. Also ist $x^{(k)}$ eine Cauchyfolge in D und hat einen Grenzwert $\overline{x} \in D$. Es gilt

$$||\overline{x} - g(\overline{x})|| = \lim ||x^{(k+1)} - g(\overline{x})|| = \lim ||g(x^{(k)}) - g(\overline{x})|| \leq \lim q||x^{(k)} - \overline{x}|| = 0,$$

also ist $\overline{x}$ der einzige Fixpunkt von g. □

Korollar 5.3 *(Konvergenz der Fixpunktiteration)*
Seien für g die Voraussetzungen aus 5.2 erfüllt, insbesondere g kontrahierend. Dann konvergiert die Fixpunktiteration

$$x^{(k+1)} = g(x^{(k)}),\ x^{(0)} \in D$$

gegen einen Fixpunkt von g.

Wenn wir die Fixpunktiteration zur approximativen Berechnung des Fixpunkts nutzen wollen, brauchen wir Abschätzungen, wie nah ein Folgeglied bereits am Grenzwert liegt.

Korollar 5.4 Fehlerabschätzung
Seien für g die Voraussetzungen aus 5.2 erfüllt, und q sei die Kontraktionskonstante von g. Es gilt

$$||\overline{x} - x^{(k)}|| = \lim_{l \mapsto \infty} ||x^{(l)} - x^{(k)}|| \leq \frac{q^k}{1-q} ||x^{(1)} - x^{(0)}||$$

mit (5.1).
Sei $y^{(j)}$ eine zweite Fixpunktiteration mit Startwert $x^{(k)}$. Dann lautet die Abschätzung, angewandt auf $y^{(0)}$

$$||\overline{x} - x^{(k)}|| = ||\overline{x} - y^{(0)}|| \leq \frac{1}{1-q} ||y^{(1)} - y^{(0)}|| = \frac{1}{1-q} ||x^{(k+1)} - x^{(k)}||$$

oder angewandt auf $y^{(1)}$

$$||\overline{x} - x^{(k+1)}|| = ||\overline{x} - y^{(1)}|| \leq \frac{q}{1-q} ||y^{(1)} - y^{(0)}|| = \frac{q}{1-q} ||x^{(k+1)} - x^{(k)}||.$$

Mit Hilfe der ersten Abschätzung können wir im Vorhinein (a priori) eine obere Schranke für den Konvergenzfehler angeben. Mit Hilfe der zweiten Abschätzung können wir im Nachhinein (a posteriori), wenn wir also bereits das $k+1$. Folgeglied berechnet haben, ebenfalls eine obere Schranke angeben. Notwendigerweise ist die a priori–Abschätzung eine worst case–Abschätzung, während die a posteriori–Abschätzung auf dem tatsächlichen Folgenverlauf basiert. Deshalb ist normalerweise die a posteriori–Abschätzung deutlich schärfer.

Beispiel 5.5

1. $$g : \mathbb{R} \mapsto \mathbb{R} : \; g(x) := 0.9 \cos(x)$$

   ist kontrahierend: Seien $x, y \in \mathbb{R}$. Dann ist nach dem Mittelwertsatz

   $$|g(x) - g(y)| = |g'(\xi)||x - y| \leq 0.9 \, |x - y|.$$

2. $$g : \mathbb{R} \mapsto \mathbb{R} : \; g(x) := \cos(x)$$

   ist nicht kontrahierend, denn

   $$\lim_{\pi \neq x \mapsto \pi} \frac{|\cos(x) - \cos(\pi)|}{|x - \pi|} = 1$$

   und damit gibt es kein $q < 1$, das diesen Ausdruck nach oben begrenzt.

3.

$$g : [-0.1, 0.1] \mapsto [0.99, 1],\ g(x) = \cos(x)$$

ist kontrahierend, aber keine Selbstabbildung.

4.

$$g : [0.6, 0.9] \mapsto [0.6, 0.9],\ g(x) = \cos(x)$$

ist kontrahierend und Selbstabbildung.

5.

$$g : \mathbb{R}^2 \mapsto \mathbb{R}^2,\ g(a, b) := (a/2 - 2b, 0).$$

g *hat nur den einen Fixpunkt* $(0, 0)$*, ist aber nicht kontrahierend in der euklidischen Norm, denn*

$$||g(0, 1)||^2 = ||(-2, 0)||^2 > ||(0, 1)||^2.$$

Trotzdem ist die Fixpunktiteration mit beliebigen Startwert konvergent. Sei nämlich

$$x^{(0)} = (a, b),\ \textit{also}\ x^{(1)} = (a/2 - 2b, 0)$$

und damit

$$x^{(k+1)} = \frac{1}{2^k}(a/2 - 2b, 0).$$

Die Folge konvergiert also immer gegen den einzigen Fixpunkt $(0, 0)$*. Die Kontraktionseigenschaft ist also nur hinreichend, aber nicht notwendig für die Konvergenz.*
Tatsächlich können wir in diesem Fall einfach die Kontraktionseigenschaft durch Wahl einer geeigneten Norm erzwingen. Statt der euklidischen Norm wählen wir auf dem $\mathbb{R}^2$ *die Norm*

$$||(a, b)|| := \max(|a - 4b|, |b|).$$

$||\cdot||$ *ist tatsächlich eine Norm, und es gilt*

$$\begin{aligned} ||g(a, b)|| &= ||(a/2 - 2b, 0)|| \\ &= |\frac{1}{2}a - 2b| \\ &= \frac{1}{2}|a - 4b| \\ &\leq \frac{1}{2}\max(|a - 4b|, |b|) \\ &= \frac{1}{2}||(a, b)||. \end{aligned}$$

Anders als die Konvergenz hängt also die Kontraktionseigenschaft von der Normwahl ab. In 5.13 werden wir uns daher fragen, wann eine Norm existiert, so dass eine vorgelegte (lineare) Funktion kontrahierend ist.

Die Beispiele führen zu

Satz 5.6 *(Abschätzung der Kontraktionskonstante für differenzierbare Funktionen) Seien $g : D \mapsto \mathbb{R}^m$, $D \subset \mathbb{R}^n$ abgeschlossen, und g sei stetig differenzierbar. Es gelte*

$$||g'(x)|| \leq q < 1 \, \forall x \in D,$$

wobei $g'(x)$ die Jakobimatrix von g an der Stelle x ist. Weiter sei D konvex. Dann ist g kontrahierend mit der Kontraktionskonstante q.
Falls $||g'(x)|| \geq 1$ für ein x im Inneren von D, so ist g in einer Umgebung von x nicht kontrahierend.

Beweis: (Nur Hinrichtung) Seien $x, \, y \in D$. D ist konvex, also ist die Funktion

$$G(\lambda) = g(x + \lambda(y - x)), \, \lambda \in [0, 1]$$

wohldefiniert. G ist differenzierbar, es gilt

$$G'(\lambda) = g'(x + \lambda(y - x))(y - x).$$

$$\begin{aligned} ||g(x) - g(y)|| &= ||G(0) - G(1)|| \\ &= || \int_0^1 G'(\lambda) \mathrm{d}\lambda|| \\ &\leq \sup_{\lambda \in [0,1]} ||G'(\lambda)|| \\ &= \sup_{\lambda \in [0,1]} ||g'(x + \lambda(y - x))|| \cdot ||y - x|| \\ &\leq q||x - y|| \end{aligned}$$

also ist g kontrahierend mit Kontraktionskonstante q. □

Korollar 5.7 *Es sei der $\mathbb{R}^n$ versehen mit der Norm $|| \cdot ||$ und*

$$g : (\mathbb{R}^n, || \cdot ||) \mapsto (\mathbb{R}^n, || \cdot ||), \, g(x) := Bx + c, \, B \in \mathbb{R}^{n \times n}, \, c \in \mathbb{R}^n.$$

g ist genau dann kontrahierend, wenn $||B|| < 1$ in der induzierten Matrixnorm. Falls $||B|| < 1$, so konvergiert die Fixpunktiteration gegen einen Fixpunkt

$$\overline{x} = (I - B)^{-1} c.$$

Beweis:

$$g'(x) = B.$$

□

Hoffentlich kommt Ihnen dieser Satz bekannt vor: Für $x_0 = 0$ gilt

$$x^{(1)} = c,\ x^{(2)} = Bc + c,\ x^{(3)} = B^2c + Bc + c, \ldots$$

Dies ist gerade die Neumannsche Reihe, deren Konvergenz gegen $(I - B)^{-1}c$ wir bereits in 2.31 nachgewiesen haben.

Beispiel 5.8 *Häufig ist die zielführende Formulierung der Fixpunktgleichung nicht sofort klar. Wir suchen den Fixpunkt von* $\tan x$ *in* $[\pi/2, 3\pi/2]$*. Die Wahl*

$$g(x) = \tan x,\ g'(x) = \frac{1}{\cos^2 x} \geq 1$$

führt offensichtlich nicht zum Ziel, denn g *ist dann nicht einmal kontrahierend. Wir formen daher die Fixpunktgleichung um. Hier wählen wir*

$$\overline{x} = \tan(\overline{x}) \Longleftrightarrow \arctan \overline{x} = \overline{x} - \pi \Longleftrightarrow \overline{x} = \pi + \arctan \overline{x}.$$

Mit

$$D = [\frac{\pi}{2}, \frac{3\pi}{2}],\ g(x) = \pi + \arctan x,\ g'(x) = \frac{1}{1 + x^2}$$

bildet g D *auf* D *ab und ist nach 5.6 kontrahierend mit Kontraktionskonstante*

$$q = \frac{1}{1 + \pi^2/4} \sim 0.29.$$

D *enthält den gesuchten Fixpunkt. Wir führen die Fixpunktiteration durch mit dem Startwert* $x_0 = \pi$ *und erhalten*

k	$x^{(k)}$	*a priori*	*a posteriori*	$\lvert\overline{x} - x^{(k)}\rvert$	$\lvert\overline{x} - x^{(k)}\rvert / \lvert\overline{x} - x^{k-1}\rvert$
1	3.1416			$1.3518e+000$	
2	4.4042	$1.4758e-001$	$1.7744e+000$	$8.9190e-002$	$6.5978e-002$
3	4.4891	$4.2563e-002$	$1.1931e-001$	$4.2900e-003$	$4.8100e-002$.
4	4.4932	$1.2275e-002$	$5.7439e-003$	$2.0266e-004$	$4.7239e-002$
5	4.4934	$3.5401e-003$	$2.7132e-004$	$9.5871e-006$	$4.7307e-002$
6	4.4934	$1.0210e-003$	$1.2804e-005$	$4.7571e-007$	$4.9619e-002$

Offensichtlich ist die a priori–Abschätzung viel zu pessimistisch, weil die Kontraktionskonstante zu groß abgeschätzt wurde.

```
function [ output_args ] = tanbeisp( input_args )
%TANBEISP
N=200;
x=(0:N)/N*8;
plot(x,tan(x),x,x);
ylim([-8,8]);
```

Listing 5.1: Beispiel zum Banachschen Fixpunktsatz (Banach/tanbeisp.m)

Klicken für den Quellcode von Banach/tanbeisp.m

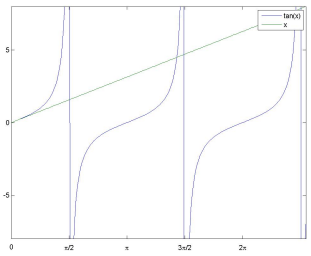

Abbildung 5.1: Bestimmung des Fixpunkts von $\tan(x) = x$

Klick für Bild tanbeisp
Klick für Matlab Figure tanbeisp

Im Allgemeinen kann man auf keine der Voraussetzungen des Banachschen Fixpunktsatzes verzichten, insbesondere nicht auf die Voraussetzung, dass g D in sich selbst abbildet (s. Übungen) – es sei denn, man setzt die Existenz eines Fixpunkts voraus. Im vierten Beispiel von 5.5 haben wir bereits gesehen, dass für $\cos(x)$ eine Umgebung des Fixpunkts im Intervall $[0, 2\pi]$ existiert, so dass $\cos(x)$ dort Selbstabbildung und kontrahierend ist. Solch ein Intervall lässt sich immer finden, wenn am Fixpunkt selbst die Funktion kontrahierend ist.

Satz 5.9 *(**Lokaler Konvergenzsatz**)*
Sei X vollständig, $g : U \mapsto X$, $U \subset X$, kontrahierend. Sei $\overline{x}$ ein Fixpunkt von g im Inneren von U. Dann gibt es eine Umgebung D von $\overline{x}$, so dass die Fixpunktiteration mit Startwerten in D gegen $\overline{x}$ konvergiert.

Beweis: Sei q die Kontraktionskonstante von g. Sei D eine abgeschlossene Kugel um $\overline{x}$ mit Radius $\epsilon > 0$, die ganz in U liegt. Wir zeigen: g bildet D in sich selbst ab. Sei $x \in D$.

$$\begin{aligned} ||g(x) - \overline{x}|| &= ||g(x) - g(\overline{x})|| \\ &\leq q||x - \overline{x}|| \\ &\leq q \cdot \epsilon < \epsilon. \end{aligned}$$

Also gilt $g(x) \in D$, und die Aussage folgt mit 5.2. □

Korollar 5.10 *Sei $g : U \mapsto \mathbb{R}^n$ stetig differenzierbar, $U \subset \mathbb{R}^n$, $\overline{x}$ ein Fixpunkt von g, und sei*

$$||g'(\overline{x})|| < 1.$$

Dann gibt es eine Umgebung D von $\overline{x}$, so dass die Fixpunktiteration mit Startwerten in D gegen $\overline{x}$ konvergiert.

Beweis: g' ist stetig, also gibt es eine Umgebung U' von $\overline{x}$ mit

$$||g'(x)|| \leq \frac{1 + ||g'(\overline{x})||}{2} < 1 \, \forall x \in U'.$$

□

Definition 5.11 *(**Asymptotische Konvergenzgeschwindigkeit**)*
Sei

$$\lim_{k \mapsto \infty} x^{(k)} = \overline{x}.$$

1. *Die Konvergenz heißt asymptotisch linear (von der Ordnung 1), falls es ein $q < 1$, $k_0 > 0$ gibt mit*

   $$||x^{(k+1)} - \overline{x}|| \leq q||x^{(k)} - \overline{x}|| \, \forall k > k_0.$$

   Falls nicht, so heißt die Konvergenz sublinear. Falls es für jedes q ein k_0 mit der Eigenschaft gibt, so heißt die Folge superlinear.

2. *Sei $p > 1$. Die Konvergenz heißt asymptotisch von der Ordnung p, falls es ein $C > 0$, $k_0 > 0$ gibt mit*

   $$||x^{(k+1)} - \overline{x}|| \leq C||x^{(k)} - \overline{x}||^p \, \forall k > k_0.$$

3. *Sei $\epsilon^{(k)}$ eine positive Nullfolge, die mit der Ordnung p konvergiert. Sei $x^{(k)}$ eine Folge, die gegen $\overline{x}$ konvergiert, und es gelte*

$$||x^{(k)} - \overline{x}|| \leq \epsilon^{(k)} \, \forall k.$$

Dann heißt auch $x^{(k)}$ konvergent von der Ordnung p.

Beispiel 5.12

1. *Fixpunktiterationen für kontrahierende Funktionen g sind asymptotisch konvergent von der Ordnung 1 (a priori–Abschätzung). Wir erhalten in jedem Iterationsschritt eine Verbesserung der Approximation des Grenzwerts um den Faktor q.*

2. *Das Newtonverfahren (siehe übernächstes Kapitel) ist konvergent von der Ordnung 2 (quadratische Konvergenz). Falls im $k.$ Schritt 1 Dezimalstelle korrekt ist, so sind es in den nächsten 2, 4, 8, 16, die Konvergenz ist also erheblich schneller als im linearen Fall.*

Von besonderem Interesse ist die lineare Schrittfunktion

$$g : \mathbb{R}^n \mapsto \mathbb{R}^n, \; g(x) = Bx + c, \; B \in \mathbb{R}^{n \times n}, \; c \in \mathbb{R}^n.$$

g ist kontrahierend genau dann, wenn $||B|| < 1$ in der induzierten Matrixnorm nach Korollar 5.7.
Hierbei hängt die Kontraktionseigenschaft wie schon in 5.5 bemerkt von der gewählten Vektorraumnorm ab, die Konvergenz der Fixpunktiteration aber nicht (nach 2.10). Zum Beispiel:

$$B = \begin{pmatrix} 0.9 & 0.2 \\ -0.2 & 0.9 \end{pmatrix},$$

$$||B||_\infty = 1.1 > 1, \; ||B||_2 = \sqrt{\rho(B^t B)} = \sqrt{\rho \begin{pmatrix} 0.85 & 0 \\ 0 & 0.85 \end{pmatrix}} \sim 0.92 < 1$$

mit der Definition von $\rho(B)$ als Betrag des betragsgrößten Eigenwerts von B (**Spektralradius**, s. 2.28). g ist also kontrahierend bezüglich der 2–Norm, aber nicht kontrahierend bezüglich der ∞–Norm. Trotzdem konvergiert die Fixpunktiteration für beide Normen. Dies ist kein Widerspruch, denn die Kontraktionseigenschaft ist hinreichend, aber nicht notwendig.
Offensichtlich gilt: Falls es irgendeine Norm auf dem $\mathbb{R}^n$ gibt, so dass $||B||$ in der induzierten Matrixnorm kleiner als 1 ist, so ist g in dieser Norm kontrahierend, und das Fixpunktverfahren konvergiert (bezüglich jeder Norm). Wann gibt es also für eine vorgelegte Matrix eine solche Norm? Dies beantwortet

Satz 5.13 *(Infimum der induzierten Matrixnormen)*
Sei $B \in \mathbb{C}^{n\times n}$. Dann ist

$$\rho(B) = \inf_{||\cdot|| \text{ induzierte Matrixnorm}} ||B||.$$

Insbesondere gibt es für alle $\epsilon > 0$ eine Vektorraumnorm $||\cdot||_{B,\epsilon}$ auf dem $\mathbb{C}^n$, so dass

$$\rho(B) \leq ||B||_{B,\epsilon} \leq \rho(B) + \epsilon$$

in der induzierten Matrixnorm.

Beweis: Sei zunächst x ein Eigenvektor von B zum Eigenwert λ mit $|\lambda| = \rho(B)$, $||\cdot||$ eine Norm auf dem $\mathbb{C}^n$. Dann gilt

$$||B|| \geq \frac{||Bx||}{||x||} = \frac{||\lambda x||}{||x||} = |\lambda| = \rho(B).$$

Für den Beweis der Existenz betrachten wir zunächst den einfachen Fall, dass B symmetrisch ist. Dann gilt für die euklidische Norm

$$||B||_2 = \sqrt{\rho(B^*B)} = \sqrt{\rho(B^2)} = \rho(B)$$

und die Aussage ist bereits für die 2–Norm und alle $\epsilon > 0$ erfüllt.
Für nicht–symmetrische Matrizen ist der Satz leider schwieriger. Sei $\epsilon > 0$ fest gewählt, $D \in \mathbb{R}^{n\times n}$ die Diagonalmatrix mit $D_{ii} = \epsilon^{i-1}$, und sei J die Jordan–Normalform von B. Es gibt also eine invertierbare Matrix $X \in \mathbb{C}^{n\times n}$ mit

$$D = \begin{pmatrix} 1 & & & \\ & \epsilon & & \\ & & \ddots & \\ & & & \epsilon^{n-1} \end{pmatrix}, \; J = XBX^{-1} = \begin{pmatrix} \lambda_1 & \sigma_1 & & & \\ & \lambda_2 & \sigma_2 & & \\ & & \ddots & \ddots & \\ & & & \lambda_{n-1} & \sigma_{n-1} \\ & & & & \lambda_n \end{pmatrix},$$

wobei $\sigma_k \in \{0, 1\}$ und die λ_k die Eigenwerte von B sind. Es gilt

$$C := D^{-1}JD = \begin{pmatrix} \lambda_1 & \epsilon\sigma_1 & & & \\ & \lambda_2 & \epsilon\sigma_2 & & \\ & & \ddots & \ddots & \\ & & & \lambda_{n-1} & \epsilon\sigma_{n-1} \\ & & & & \lambda_n \end{pmatrix}.$$

Mit 2.30 gilt

$$||C||_\infty \leq \rho(B) + \epsilon.$$

Für eine invertierbare lineare Abbildung L und jede Norm $||\cdot||$ ist

$$|||x||| := ||Lx||$$

ebenfalls eine Norm, also insbesondere auch

$$||x||_{B,\epsilon} := ||D^{-1}Xx||_\infty.$$

Für die induzierte Matrixnorm gilt

$$\begin{aligned}
||B||_{B,\epsilon} &= \sup_{x\neq 0} \frac{||Bx||_{B,\epsilon}}{||x||_{B,\epsilon}} \\
&= \sup_{x\neq 0} \frac{||D^{-1}XX^{-1}JDD^{-1}Xx||_\infty}{||D^{-1}Xx||_\infty} \\
&= \sup_{y\neq 0} \frac{||D^{-1}JDy||_\infty}{||y||_\infty},\ y = D^{-1}Xx \\
&= ||C||_\infty \\
&\leq \rho(B) + \epsilon.
\end{aligned}$$

□

Hieraus folgern wir den Hauptsatz über die Konvergenz von Fixpunktverfahren für lineare Gleichungssysteme.

Korollar 5.14 *Sei $A \in \mathbb{R}^{n\times n}$. $A = M - N$, M invertierbar, $b,\ x_0 \in \mathbb{R}^n$. Die Folge $x^{(k)} \in \mathbb{R}^n$ sei definiert durch*

$$x^{(k+1)} = M^{-1}(Nx^{(k)} + b),\ \textit{also}\ Mx^{(k+1)} = Nx^{(k)} + b.$$

Sei $B := M^{-1}N$. $x^{(k)}$ konvergiert genau dann für alle $x_0,\ b$ gegen die Lösung von $Ax = b$, wenn $\rho(B) < 1$.

Beweis:
Sei $x \neq 0$ im Kern von A, also $Mx = Nx$ oder $x = M^{-1}Nx = Bx$. Also ist x Eigenvektor von B zum Eigenwert 1 und damit $\rho(B) \geq 1$.
Sei nun $\rho(B) < 1$, dann besteht der Kern von A nur aus der 0 und A ist invertierbar. Die Iteration ist von der Form

$$x^{(k+1)} = Bx^{(k)} + M^{-1}b,$$

also konvergiert die Iteration nach 5.13 und 5.7 gegen einen Fixpunkt $\overline{x}$ mit

$$M\overline{x} = N\overline{x} + b$$

und damit

$$A\overline{x} = b.$$

Sei nun $\rho(B) \geq 1$. Dann gibt es einen Eigenvektor x zum Eigenwert λ mit $|\lambda| \geq 1$. Setze $b := 0$, $x^{(0)} := x$. Dann gilt

$$x^{(k)} = \lambda^k x^{(0)}.$$

Insbesondere konvergiert die Folge in diesem Fall nicht gegen die Lösung 0. □
Mit der Wahl $M = I$, $N = (I - B)$ erhalten wir wieder die Neumannsche Reihe zurück.
Mit diesem Wissen können wir auch die seltsame Normwahl im letzten Beispiel aus 5.5 aufklären.

Beispiel 5.15
Sei wie in 5.5

$$g : \mathbb{R}^2 \mapsto \mathbb{R}^2,\ g(x) := \begin{pmatrix} 1/2 & -2 \\ 0 & 0 \end{pmatrix} x =: Bx.$$

B ist diagonalisierbar und hat die Eigenvektoren $(1,0)$ zum Eigenwert $1/2$ und $(4,1)$ zum Eigenwert 0, es gilt also

$$J = XBX^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 0 \end{pmatrix},\ X^{-1} = \begin{pmatrix} 1 & 4 \\ 0 & 1 \end{pmatrix},\ X = \begin{pmatrix} 1 & -4 \\ 0 & 1 \end{pmatrix}.$$

Nach 5.13 gibt es eine Norm $||\cdot||$, so dass bezüglich der induzierten Matrixnorm gilt $||B|| < 1$.
Da B diagonalisierbar ist, können wir uns das Hantieren mit der Matrix D aus 5.13 sparen, die wurde nur benötigt, um auch mit Jordan–Matrizen umgehen zu können. Wir setzen also gleich

$$||(a,b)^t|| := ||X(a,b)^t||_\infty = \max(|a - 4b|, |b|).$$

Mit der Rechnung aus 5.5 gilt dann tatsächlich $||B|| = 1/2$ und die Fixpunktiteration zu g konvergiert gegen den einzigen Fixpunkt $(0,0)^t$.

Beispiel 5.16 *(Chaotisches Verhalten von Fixpunktiterationen)*
Abschließend betrachten wir noch Beispiel dafür, dass das Langzeitverhalten von Fixpunktiterationen für nicht–kontrahierende Funktionen schwierig vorherzusagen ist. Bekannt ist die Mandelbrotmenge, die aus der Konvergenzanalyse komplexer Fixpunktiterationen kommt. Wir betrachten die noch einfachere reelle Funktion

$$g_\lambda(x) := \lambda x(1 - x),$$

das logistische Modell für Bevölkerungsdynamik. Hier ist die Frage interessant, ob es stationäre Bevölkerungsdaten gibt, bei denen sich Geburten und Todeszahlen die Waage halten, und ob diese stationären Punkte erreicht werden. Hierbei steht λ für die Geburtenrate zwischen 0 und 4. Klarerweise konvergiert die Populationszahl gegen 0, wenn die Geburtenrate zu klein ist ($\lambda < 1$). Für λ zwischen 1 und 3 ist der einzige positive Fixpunkt attraktiv (d.h. die Ableitung am Fixpunkt ist vom Betrag kleiner als 1, also gibt es nach 5.10 eine kleine Umgebung, in der die Fixpunktiteration konvergiert). Ab 3 ist der Fixpunkt abstoßend, und die Konvergenzanalyse wird unübersichtlich.
Offensichtlich ist g_λ Selbstabbildung auf dem Intervall $[0,1]$ für $\lambda \in [0,4]$. Der Plot zeigt für $\lambda \in [0,4]$ das Verhalten der Folgeglieder x_{1024} bis x_{4096}.

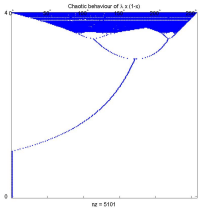

Abbildung 5.2: Chaotisches Verhalten von Fixpunktiterationen

Klick für Bild chaos
Klick für Matlab Figure chaos

```
function chaos( lambdao, lambda1, N ,M)
%CHAOS Display chaotic behaviour of g(x)=lambda x (1-x)
if (nargin<1)
    lambdao=0;
end
if (nargin<2)
```

Listing 5.2: Chaotisches Verhalten von Fixpunktiterationen (Banach/chaos.m)

Klicken für den Quellcode von Banach/chaos.m

5.2 Fixpunktverfahren zur Lösung linearer Gleichungen

Wir definieren und untersuchen die klassischen Verfahren zur iterativen Lösung linearer Gleichungen: Jakobi (Gesamtschrittverfahren), Gauss-Seidel (Einzelschrittverfahren), SOR (Successive Over–Relaxation). Wir nutzen 5.14 und zerlegen $A = M - N$ mit einer leicht invertierbaren Matrix M.

Definition 5.17 *(**Gesamtschrittverfahren, Einzelschrittverfahren**)*
Zu lösen sei die lineare Gleichung $Ax = b$, $A = (a_{i,j}) \in \mathbb{R}^{n\times n}$ invertierbar, $b \in \mathbb{R}^n$. Wir zerlegen

$$A = L + D + R,$$

wobei L die Einträge von A unterhalb der Hauptdiagonalen enthält, R die Einträge oberhalb der Hauptdiagonalen, D die Diagonaleinträge. Insbesondere werden zur Berechnung der Matrizen keine Rechenoperationen benötigt. Weiter sei D invertierbar.

1. *Wir setzen in 5.14 $M := D$, $N := -(L + R)$ und erhalten die Fixpunktiteration*

$$Dx^{(k+1)} = b - (L + R)x^{(k)}$$

oder

$$(x^{(k+1)})_i = \frac{1}{a_{i,i}} \left(b_i - \sum_{j\neq i} a_{i,j} x_j^{(k)} \right).$$

Wir wählen also in $x^{(k+1)}$ die die i. Komponente so, dass die i. Gleichung des Gleichungssystems erfüllt ist, wenn man sonst nichts ändert.
Falls $x^{(k)}$ konvergiert, so nach 5.14 gegen eine Lösung von $Ax = b$. Das Verfahren heißt Gesamtschritt– oder Jakobiverfahren (GSV).

2. *Wir setzen in 5.14 $M := D + L$, $N := -R$ und erhalten die Fixpunktiteration*

$$(D + L)x^{(k+1)} = b - Rx^{(k)}$$

oder

$$(x^{(k+1)})_i = \frac{1}{a_{i,i}} \left(b_i - \sum_{j>i} a_{i,j} x_j^{(k)} - \sum_{j<i} a_{i,j} x_j^{(k+1)} \right).$$

Wir wählen also in $x^{(k+1)}$ die i. Komponente so, dass die i. Gleichung des Gleichungssystems erfüllt ist, wenn man die Änderungen sequentiell berechnet und für $j < i$ die bereits berechneten Änderungen durchführt.
Falls $x^{(k)}$ konvergiert, so nach 5.14 gegen eine Lösung von $Ax = b$. Das Verfahren heißt Einzelschritt– oder Gauss–Seidel–Verfahren (ESV).

Bemerkung:

1. Der Aufwand zur Berechnung eines Schritts der Verfahren ist gleich der Anzahl der von Null verschiedenen Einträge in A.

2. Das Gesamtschrittverfahren ist leicht parallelisierbar, denn alle Änderungen werden unabhängig voneinander berechnet. Das Einzelschrittverfahren ist so nicht parallelisierbar, denn alle Änderungen müssen nacheinander durchgeführt werden.

3. Die Verfahren konvergieren für jeden Startwert gegen die Lösung von $Ax = b$, falls
$$\rho(D^{-1}(L + R)) < 1 \,(\text{Gesamtschrittverfahren})$$
bzw.
$$\rho((D + L)^{-1}R) < 1 \,(\text{Einzelschrittverfahren}).$$

Der Unterschied zwischen den Verfahren zeigt sich am einfachsten in der Implementation. Die Routinen führen jeweils N Schritte des Verfahrens zur Lösung von $Ax = b$ aus.

function Gesamtschritt (n,A,b,$x0$,N)
$x = x0$
Für $k = 1 \dots N$
    Für $i = 1 \dots n$
        $y_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j \right)$
    $x = y$
return x

function Einzelschritt (n,A,b,$x0$,N)
$x = x0$
Für $k = 1 \dots N$
    Für $i = 1 \dots n$
        $x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_i \right)$
return x

```
function x = einzelschritt( A,b,xo,N )
%EINZELSCHRITT Einzelschrittverfahren, Gauss-Seidel
%We should assume that A is sparse.
x=xo;
n=numel(xo);
for i=1:N
```

Listing 5.3: Einzelschrittverfahren (Einzelgesamtsor/einzelschritt.m)

Klicken für den Quellcode von Einzelgesamtsor/einzelschritt.m

```
function x = gesamtschritt( A,b,xo,N )
%GESAMTSCHRITT Gesamtschritt, Jakobiverfahren
%Assume that A is sparse.
n=numel(xo);
D=diag(A);
LU=A;
```

Listing 5.4: Gesamtschrittverfahren (Einzelgesamtsor/gesamtschritt.m)

Klicken für den Quellcode von Einzelgesamtsor/gesamtschritt.m

```
function doit( N )
%DOIT Solve Ax=b using Jakobi/Gauss-Seidel.
%Use discretization of Laplace operator
%Cheat: We offset by lambda, otherwise the iteration is
%painfully slow.
if (nargin<1)
```

Listing 5.5: Treiber für Einzel-Gesamtschritt (Einzelgesamtsor/doit.m)

Klicken für den Quellcode von Einzelgesamtsor/doit.m

Zur Untersuchung der Konvergenzeigenschaften müssen wir die Eigenwerte einer Matrix abschätzen. Dabei ist häufig der Satz von Gerschgorin nützlich.

Satz 5.18 *(**Satz von Gerschgorin**)*
Sei $A = (a_{ij}) \in \mathbb{C}^{n\times n}$. Sei $K_i \in \mathbb{C}$ (also in der komplexen Ebene) der Kreis um das Diagonalelement $a_{i,i}$ mit dem Radius der Summe der Beträge der Außerdiagonalelemente in Zeile i, also

$$r_i = \sum_{j\neq i} |a_{i,j}|,\ K_i = \{z : |z - a_{i,i}| \leq r_i\}\,.$$

Dann liegen alle Eigenwerte von A in der Vereinigung der Kreise K_i.
Falls die Vereinigung V von m Kreisen disjunkt ist zum Rest der Kreise, so liegen in V genau m Eigenwerte von A.
Also: Sei $M \subset \{1 \ldots n\}$, $m = |M|$. Weiter sei

$$\bigcup_{i\in M} K_i \cap \bigcup_{i\notin M} K_i = \emptyset,$$

dann ist

$$|\{\lambda_k \in \bigcup_{i\in M} K_i : \lambda_k \text{ Eigenwert von } A\}| = m,$$

wobei die Eigenwerte mit ihrer Vielfachheit im charakteristischen Polynom gezählt werden.

Zunächst ein kurzes Beispiel. Wir betrachten

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

Die Gerschgorinkreise sind der Kreis K_1 um 4 mit Radius 1, der Kreis K_2 um 1 mit Radius 1 und der Kreis K_3 um $1/2$ mit Radius 1 (alles in der komplexen Ebene, natürlich). Dann garantiert der Satz von Gerschgorin, dass in K_1 genau ein Eigenwert von A liegt, in $K_2 \cup K_3$ liegen zwei.
Ausdrücklich: Der Satz von Gerschgorin garantiert in diesem Fall **nicht**, dass in K_2 bzw. K_3 ein Eigenwert liegt (nur in der Vereinigung liegen zwei).

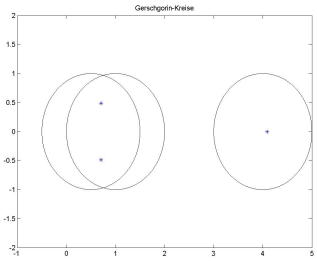

Abbildung 5.3: Gerschgorin–Kreise von A

Klick für Bild gerschgorin
Klick für Matlab Figure gerschgorin

```
function gerschgorin
%GERSCHGORIN Demo Gerschgorin-Kreise
%A=[1 3 3 ;4 5 3;7 8 2];
%A=diag([1 2 3 4])+rand(4);
%A=[5 1 0 1; 2 4 1 0; 0 1 4 1; 2 2 1 6];
A=[4 0 1; 1 1 0; 0 1 0.5];
```

Listing 5.6: Stetige Abhängigkeit der Nullstellen und Gerschgorinkreise (Gerschgorin/gerschgorin.m)

Klicken für den Quellcode von Gerschgorin/gerschgorin.m

Beweis:

1. Sei λ ein Eigenwert von A. Sei x Eigenvektor von A zum Eigenwert λ mit

$||x||_\infty = 1$. Es gibt also ein m mit $|x_m| = 1$.

$$\begin{aligned}(A - \lambda I)x = 0 &\implies (a_{m,m} - \lambda)x_m = -\sum_{j \neq m} a_{m,j}x_j \\ &\implies |a_{m,m} - \lambda| \leq \sum_{j \neq m} |a_{m,j}| \cdot |x_j| \\ &\implies |a_{m,m} - \lambda| \leq \sum_{j \neq m} |a_{m,j}| = r_m\end{aligned}$$

2. Wir zitieren den Satz: Die Nullstellen eines Polynoms hängen stetig von seinen Koeffizienten ab. Falls die Eigenwerte keine mehrfachen Nullstellen des charakteristischen Polynoms sind, so folgt das einfach mit dem Satz über implizite Funktionen, andernfalls muss man etwas mehr arbeiten. Ein vollständiger Beweis mit der Ordnung der Abhängigkeit findet sich in Kato [1995], Satz II.1.7.
   Sei
   $$V = \bigcup_{i \in M} K_i$$
   wie im Satz disjunkt zur Vereinigung W der restlichen Kreise. Sei weiter $A = D + L + R$ wie in 5.17. Wir betrachten die Matrizen
   $$A(t) = D + t(L + R), t \in [0, 1],\ A(0) = D,\ A(1) = A.$$
   $A(0) = D$ ist Diagonalmatrix, die Eigenwerte stehen alle auf der Hauptdiagonalen. In V liegen nach Voraussetzung also genau ihre Eigenwerte $a_{i,i}$, $i \in M$. $A(t)$ ist stetig in t, d.h. auch die Eigenwerte hängen stetig von t ab. Die Diagonalelemente von $A(t)$ sind die von A, die Außerdiagonalelemente werden mit t multipliziert. Nach Teil 1 liegen die Eigenwerte von $A(t)$ also in der Vereinigung der Kreise um $a_{i,i}$ mit Radius tr_i, also insbesondere in $V \cup W$.
   Wir betrachten nun die Abhängigkeit eines Eigenwerts $\lambda_i(t)$ von t. Für die Kurven $\lambda_i(t)$, $t \in [0, 1]$, gilt
   (a) Sie sind stetig.
   (b) Sie beginnen in einem $a_{i,i}$.
   (c) Die Anzahl der Kurven, die in V beginnen, ist gleich der Anzahl der Diagonalelemente, die in V liegen, also gerade m.
   (d) Sie enden in einem Eigenwert λ_i von $A = A(1)$.
   (e) Sie liegen ganz in $V \cup W$.

(f) Da V und W disjunkt sind, muss die (stetige) Kurve entweder ganz in V oder ganz in W liegen. Die Anzahl der Kurven, die in V enden, ist die Anzahl der Kurven, die in V beginnen, also m.

Also gilt insbesondere

$$|\{\lambda_i : \lambda_i \text{ Eigenwert von } A\}| = |M|.$$

□

Bemerkung: Da die Eigenwerte von A und A^t dieselben sind, kann man den Satz statt auf die Zeilensumme auch auf die Spaltensumme anwenden.
Häufig kann man die Abschätzung verschärfen, indem man das Kriterium statt auf A auf $\mathcal{D}A\mathcal{D}^{-1}$ mit einer Diagonalmatrix $\mathcal{D}$ anwendet (7.5).
Das Programm gerschgorin visualisiert die Kurve der Eigenwerte und die Gerschgorinkreise von $A(t)$ für A aus dem Beispiel.

Definition 5.19 *(**strikte Diagonaldominanz**) Sei $A \in \mathbb{R}^{n\times n}$. Sei*

$$r_i = \sum_{i\neq k} |a_{i,k}|$$

und

$$r_i < |a_{i,i}| \, \forall i \in \{1 \dots n\}.$$

Dann heißt A strikt diagonaldominant.

Korollar 5.20 *Sei A strikt diagonaldominant. Dann ist A invertierbar, Einzel– und Gesamtschrittverfahren zur Lösung von $Ax = b$ konvergieren.*

Beweis: Da $|a_{ii} - 0| > r_i$, ist 0 nicht im i–ten Gerschgorinkreis enthalten, also ist 0 kein Eigenwert von A und damit A invertierbar.
Zur Konvergenz des Gesamtschrittverfahrens ist zu zeigen, dass

$$\rho(B) = \rho(D^{-1}(L + R)) < 1.$$

$B_{i,i} = 0$, also sind alle Gerschgorinkreise Kreise um 0 mit Radius $r_i/|a_{i,i}| < 1$, also sind alle Eigenwerte kleiner als 1. Die Konvergenz des Einzelschrittverfahrens zeigen wir in 5.23 gleich mit. □

Leider ist dieses Korollar in der Praxis unbrauchbar, denn typischerweise addieren sich bei der Diskretisierung von Differentialgleichungen in den Außerdiagonalen in fast allen Zeilen die Beträge der Elemente zum Betrag des Diagonalelements auf, so

dass 5.19 nur mit $\leq$ statt $<$ erfüllt ist (siehe z.B. in den Übungen die Diskretisierungen der zweiten Ableitung). Dies reicht aber offensichtlich nicht aus, denn z.B. die Matrix

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

erfüllt 5.19 mit $\leq$, ist aber offensichtlich nicht invertierbar. Wir benötigen daher

Definition 5.21 *(**irreduzible Matrizen**)*
Sei $A = (a_{ik}) \in \mathbb{R}^{n\times n}$. A heißt reduzibel, falls man die Indexmenge $\{1 \dots n\}$ so in zwei nichtleere Teilmengen I_1 und I_2 zerlegen kann, dass

$$a_{i,k} = 0 \, \forall (i,k) \in I_1 \times I_2.$$

Also:

$$A \text{ reduzibel} \Leftrightarrow \exists \, I_1 \neq \emptyset \neq I_2 : I_1 \cup I_2 = \{1 \dots n\}, \; I_1 \cap I_2 = \emptyset : \; a_{i,k} = 0 \, \forall \, (i,k) \in I_1 \times I_2.$$

Ordnet man Zeilen und Spalten so an, dass zunächst die Indizes in I_1, dann die in I_2 berücksichtigt werden, so hat A mit dieser Anordnung die Darstellung

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

Falls keine solche Zerlegung existiert, so heißt A irreduzibel. Also:

$$\forall I_1, \, I_2 : I_1 \neq \emptyset \neq I_2, I_1 \cup I_2 = \{1 \dots n\}, \, I_1 \cap I_2 = \emptyset : \, \exists (i,k) \in I_1 \times I_2 : a_{i,k} \neq 0.$$

Definition 5.22 *(**schwach diagonaldominant**)*
Sei $A \in \mathbb{R}^{n\times n}$, $r_i = \sum_{k\neq i} |a_{i,k}|$. Es sei 5.19 erfüllt mit $\leq$, d.h.

$$r_i \leq |a_{i,i}| \, \forall i \in \{1 \dots n\}.$$

Zusätzlich sei die Ungleichung mit $<$ in einer Zeile erfüllt, d.h.

$$\exists m : r_m < |a_{m,m}|.$$

Weiter sei A irreduzibel. Dann heißt A schwach diagonaldominant.

Satz 5.23 *(**Konvergenz von GSV und ESV bei schwacher Diagonaldominanz**)*
Sei $A \in \mathbb{R}^{n\times n}$ schwach diagonaldominant. Dann konvergieren Gesamtschritt– und Eizelschrittverfahren. Insbesondere ist A invertierbar.

Beweis: Sei $A = D + L + R$ wie in 5.17. Sei m die Zeile mit $r_m < |a_{m,m}|$.

$$B := D^{-1}(L + R), \text{ also } B_{i,k} = \begin{cases} 0 & i = k \\ \frac{a_{i,k}}{a_{i,i}} & \text{sonst} \end{cases}.$$

Sei x Eigenvektor zum Eigenwert λ von B, und $||x||_\infty = 1$. Sei

$$I_1 := \{i : |x_i| = 1\},\ I_2 := \{1 \dots n\} \setminus I_1.$$

Da $||x||_\infty = 1$, ist I_1 nichtleer. Angenommen, $m \in I_1$. Dann gilt

$$|\lambda| = |(\lambda x)_m| = |(Bx)_m| = \frac{|\sum_{k \neq m} a_{m,k} x_k|}{|a_{m,m}|} \leq \frac{\sum_{k \neq m} |a_{m,k}|}{|a_{m,m}|} \leq \frac{r_m}{|a_{m,m}|} < 1.$$

Sei nun $m \notin I_1$, also $m \in I_2$. Dann sind I_1 und I_2 nichtleer, und nach Definition der Irreduzibilität gibt es $i \in I_1$, $j \in I_2$ mit $a_{i,j} \neq 0$. Dann gilt

$$|\lambda| = |(\lambda x)_i| = |(Bx)_i| \leq \frac{\sum_{k \neq i} |a_{i,k}| \cdot |x_k|}{|a_{i,i}|} < \frac{\sum_{k \neq i} |a_{i,k}|}{|a_{i,i}|} \leq \frac{r_i}{a_{i,i}} \leq 1.$$

Hier steht ein $<$, weil $|x_j| < 1$ ($j \in I_2$) und $a_{i,j} \neq 0$. Also gilt in jedem Fall, dass das Gesamtschrittverfahren konvergiert. Der Fixpunkt ist eindeutig, d.h. insbesondere ist auch A invertierbar.
Wir betrachten nun das Einzelschrittverfahren, also

$$B = (D + L)^{-1} R.$$

Für einen Eigenwert λ verschwindet das charakteristische Polynom $\chi_B(\lambda)$

$$\chi_B(\lambda) = \det(-(D + L)^{-1} R - \lambda I) = \det((D + L)^{-1}) \cdot \det(-R - \lambda(D + L)) = 0.$$

A ist schwach diagonaldominant. Sei $|\lambda| \geq 1$. Dann ist auch

$$C(\lambda) = R + \lambda(D + L)$$

mindestens schwach diagonaldominant (sogar strikt für $|\lambda| > 1$). Insbesondere ist $C(\lambda)$ damit nach Teil 1 invertierbar, ihre Determinante verschwindet also nicht, und damit ist λ kein Eigenwert. Es gibt also keinen Eigenwert λ von B mit $|\lambda| \geq 1$, und damit konvergiert das Einzelschrittverfahren (Beweis nach James [1973]).
Mit der gleichen Argumentation, angewandt auf starke Diagonaldominanz, ist das natürlich auch der Beweis für die Konvergenz des Einzelschrittverfahrens in 5.20. □

Die vorgestellten Verfahren neigen dazu, zu stark auszuschlagen. Üblicherweise nutzt man daher für $x^{(k+1)}$ eine Linearkombination aus dem berechneten und $x^{(k)}$.

Definition 5.24 *(**Relaxierte Verfahren**)*
Sei $A \in \mathbb{R}^{n\times n}$ und ω fest. Sei $A = L + D + R$ wie in 5.17 und $x^{(0)} \in \mathbb{R}^n$. Die Folge $x^{(k)}$ sei definiert durch

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega D^{-1}\left(b - (L+R)x^{(k)}\right).$$

$x^{(k)}$ heißt relaxiertes Gesamtschrittverfahren.
Für das Einzelschrittverfahren definieren wir die Relaxation wieder pro Element und erhalten

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k)}\right)/a_{i,i}$$

oder in Matrixschreibweise

$$(D+\omega L)x^{(k+1)} = (1-\omega)Dx^{(k)} + \omega\left(b - Rx^{(k)}\right)$$

und erhalten das relaxierte Einzelschrittverfahren.

Nach der heuristischen Herleitung würde man annehmen, dass $\omega \in [0,1]$ Sinn macht. Tatsächlich nutzt man sogar $\omega \in [0,2]$ für das relaxierte Einzelschrittverfahren und spricht von Überrelaxierung (successive over–relaxation).

Satz 5.25 *(**Konvergenz von SOR**, Ostrovski und Reich 1949/1954)*
Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit und $\omega \in (0,2)$. Dann konvergiert das relaxierte Einzelschrittverfahren (SOR–Verfahren).

Beweis: Sei $A = L + D + L^t$ wie in 5.17. Die Schrittmatrix für das SOR–Verfahren ist

$$B = (D+\omega L)^{-1}((1-\omega)D - \omega L^t).$$

Zu zeigen ist $\rho(B) < 1$. Sei x also Eigenvektor von B zum Eigenwert λ, also

$$((1-\omega)D - \omega L^t)x = \lambda(D+\omega L)x. \qquad (*)$$

Wegen

$$D_{k,k} = A_{k,k} = e_k^t A e_k = (Ae_k, e_k) > 0$$

ist D auf der Hauptdiagonalen positiv, also gilt

$$d := (Dx, x) > 0.$$

Sei

$$l := (Lx, x) = (x, L^t x) = (L^t x, x).$$

Dann gilt

$$0 < (Ax, x) = (L + D + L^t x, x) = d + 2l.$$

Mit $(*)$ und Skalarprodukt mit x gilt

$$(1 - \omega)d - \omega l = \lambda(d + \omega l)$$

oder

$$\lambda = \frac{(1 - \omega)d - \omega l}{d + \omega l}.$$

Es ist $d(2 - \omega) > 0$ und damit

$$d + \omega l > \omega l - (1 - \omega)d = -((1 - \omega)d - \omega l).$$

Andererseits ist $\omega(d + 2l) > 0$ und damit

$$d + \omega l > d - \omega(d + l) = (1 - \omega)d - \omega l.$$

Insgesamt gilt also

$$|\lambda| = \frac{|(1 - \omega)d - \omega l|}{d + \omega l} < 1$$

und damit ist das SOR–Verfahren für $0 < \omega < 2$ konvergent. □

Korollar 5.26 *Sei A positiv definit. Dann konvergiert das Einzelschritt–Verfahren.*

Satz 5.27 *(**Satz von Kahan**)*
Sei $A = (a_{i,j}) \in \mathbb{R}^{n \times n}$, $a_{i,i} \neq 0$, $i = 1 \ldots n$, und $\omega \notin (0, 2)$. Dann konvergiert das SOR–Verfahren nicht für alle $b \in \mathbb{R}^n$ und Startwerte $x^{(0)} \in \mathbb{R}^n$ gegen die Lösung von $Ax = b$.

Beweis: Wir schreiben die Schrittmatrix B des SOR–Verfahrens als

$$B = \left(I + \omega D^{-1} L\right)^{-1} \left((1 - \omega)\, I - \omega D^{-1} R\right).$$

Die Matrizen auf der rechten Seite sind Dreiecksmatrizen, ihre Determinante ist das Produkt der Diagonalelemente. Es gilt also

$$\det B = (1 - \omega)^n.$$

Seien λ_i die Eigenwerte von B. B ist ähnlich zu einer Jordanmatrix, auf deren Hauptdiagonale die Eigenwerte stehen, insbesondere haben diese dieselbe Determinante. Dann gilt

$$\rho(B)^n \geq |(\lambda_1 \cdot \cdots \cdot \lambda_n)| = |\det(B)| = |1 - \omega|^n.$$

□

```
function x = sor( A,b,xo,N,omega )
%EINZELSCHRITT Einzelschrittverfahren, Gauss-Seidel
%We should assume that A is sparse.
x=xo;
n=numel(xo);
for i=1:N
```

Listing 5.7: SOR–Verfahren (Einzelgesamtsor/sor.m)

Klicken für den Quellcode von Einzelgesamtsor/sor.m

```
function spektralradius( N )
%SPEKTRALRADIUS Berechne den Spektralradius von Iterationsmatrizen
if (nargin<1)
    N=10;
end
if (numel(N)>1)
```

Listing 5.8: Vergleich der Spektralradien klassischer Verfahren (Einzelgesamtsor/-spektralradius.m)

Klicken für den Quellcode von Einzelgesamtsor/spektralradius.m

```
function [ A,L,D,R,N ] = setup_matrix( N )
%SETUP_MATRIX setup matrix of discretized Laplace operator in 2D
if (nargin<1)
    N=10;
end
lambda=0;
```

Listing 5.9: Matrixgenerierung (Einzelgesamtsor/setupmatrix.m)

Klicken für den Quellcode von Einzelgesamtsor/setupmatrix.m

Abschließend geben wir ein Iterationsverfahren zur Berechnung der Minimum–Norm–Lösung an. Dazu starten wir mit der Normalgleichung

$$A^t A\overline{x} = A^t b.$$

Die direkte Anwendung der bisher hergeleiteten Verfahren würde die Berechnung von $A^t A$ erfordern, was nach der Rechnung auf Seite 76 nicht zu empfehlen ist. Wir

starten daher, indem wir die Gleichung durch Aufaddieren von x in ein Fixpunktproblem transformieren, also

$$\overline{x} = \overline{x} + A^t(b - A\overline{x}) =: g(\overline{x}).$$

Wir nutzen Relaxation mit Parameter ω und erhalten

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega(x^{(k)} - A^t(Ax^{(k)} - b)) = x^{(k)} - \omega A^t(Ax^{(k)} - b).$$

Definition 5.28 *(**Landweber–Verfahren**)*
Gesucht sei die Minimum–Norm–Lösung von $Ax = b$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. Sei $x^{(0)} \in \mathbb{R}^n$ und $\omega > 0$ fest. Dann ist das Landweberverfahren definiert durch

$$x^{(k+1)} := x^{(k)} - \omega A^t(Ax^{(k)} - b).$$

Satz 5.29 *Es sei $0 < \omega < 2/||A||_2^2$ und $x^{(0)} \in$ Bild (A^t). Dann konvergiert das Landweberverfahren gegen die Minimum–Norm–Lösung.*

Beweis: Sei also

$$g(x) := x - \omega A^t(Ax - b).$$

Wir betrachten das Verfahren auf dem Unterraum

$$U := \text{Bild}(A^t) = \text{Kern}(A) = \text{Kern}(A^tA).$$

g ist Abbildung von U nach U, und nach Voraussetzung $x^{(0)} \in U$. A^tA hat als Abbildung von U nach U keinen Eigenwert 0 (denn U steht senkrecht auf Kern(A^tA)).
Das Verfahren hat die Schrittmatrix $B := (I - \omega A^tA)$. Es konvergiert genau dann, wenn $\rho(B) < 1$ auf U. Die Eigenwerte sind aber gerade $1 - \omega\lambda_k$, wobei λ_k die Eigenwerte von A^tA auf U sind. Wegen

$$||A||_2^2 = \rho(A^tA) = \max \lambda_k$$

gilt dann nach Wahl von ω gerade $\lambda_k \in (-1, 1)$. Damit konvergiert das Landweberverfahren gegen eine Lösung $\overline{x}$ der Fixpunktgleichung, also gerade

$$A^tA\overline{x} = A^tb.$$

Wegen $x^{(0)} \in$ Bild (A^t) gilt auch $x^{(k)} \in$ Bild(A^t) und damit $\overline{x} \in$ Bild(A^t). □
Wir wollen die Fixpunktiterationen nun noch etwas uminterpretieren. Zur Herleitung

des Landweber–Verfahrens haben wir die einfachste Form der Umwandlung einer linearen Gleichung in ein Fixpunktproblem gewählt, wir lösen also statt

$$Ax = b$$

das äquivalente Fixpunktproblem

$$x = (I - A)x + b.$$

Natürlich können wir genau so auch das transformierte Gleichungssystem

$$QAx = Qb$$

betrachten für eine einfach invertierbare Matrix Q (Vorkonditionierung). Die zugehörige Iteration lautet dann

$$x^{(k+1)} = (I - QA)x^{(k)} + Qb.$$

Für $A = L + D + R$ und $Q = D^{-1}$ erhalten wir dann das Gesamtschrittverfahren

$$x^{(k+1)} = -D^{-1}(L + R)x^{(k)} + D^{-1}b$$

und für $Q = (D + L)^{-1}$ das Einzelschrittverfahren

$$x^{(k+1)} = -(D + L)^{-1}Rx^{(k)} + (D + L)^{-1}b.$$

Die Konvergenz ist offensichtlich dann besonders gut, wenn $I - QA$ besonders kleine Norm hat. Im Optimalfall wäre $Q = A^{-1}$, aber dann bräuchten wir natürlich gar nicht erst zu iterieren.
Wir benötigen also eine grobe Approximation an A^{-1}, die sich leicht berechnen lässt. Eine Möglichkeit sind Band–Vorkonditionierer: Wir streichen alle Elemente der Matrix A außerhalb eines Bandes der Breite p um die Hauptdiagonale. Die so entstehende Bandmatrix A' ist leicht zu invertieren, und tatsächlich gilt $QA \sim I$, falls die weggestrichenen Elemente nicht allzu groß waren (Saad [2003]).
Ein typischer Vertreter der Vorkonditionierer ist das ILU– (Incomplete LU)–Verfahren. Hierbei wird zunächst die LR– (LU–) Zerlegung von A berechnet. Zur Vermeidung des Fillins werden L und R nur dort berechnet, wo ohnehin schon Einträge in A standen. Im Programm auf Seite 38 wird die Zuweisung zu $a_{j,k}^{(i+1)}$ also nur durchgeführt, wenn dort schon vorher ein Eintrag ungleich 0 stand.
Es gilt dann natürlich nicht $A = LR$, sondern nur $A \sim LR$. L und R sind einfach invertierbar, und man wählt $Q = (LR)^{-1}$. Eine genauere Analyse dieses Verfahrens findet sich ebenfalls bei Saad [2003].

```
function [ output_args ] = ILU( input_args )
%ILU Incomplete LU
if (nargin<1)
    N=10;
end
[A,L,D,R,N]=setupmatrix(N);
```

Listing 5.10: Incomplete LU (Einzelgesamtsor/ILU.m)

Klicken für den Quellcode von Einzelgesamtsor/ILU.m

Im Allgemeinen lässt sich zu den klassischen iterativen Verfahren sagen, dass sie für praktische Zwecke ohne Vorkonditionierung zu langsam konvergieren. Die Eigenwerte der Schrittmatrizen B liegen ohne Vorbehandlung zu nah an 1, auch wenn Gauss in Hanke-Bourgeois [2006], S. 78, zum Einzelschrittverfahren zitiert wird:
Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigstens nicht, wenn Sie mehr als zwei Unbekannte haben. Das indirecte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken.
Die Nutzung geeigneter, problembezogener Vorkonditionierung ist der Schlüssel zum effizienten Einsatz von Fixpunkt–Verfahren. Noch attraktiver sind sie in Kombination mit Krylovraum–Methoden, von denen wir einige Vertreter im übernächsten Kapitel kennenlernen werden.
Für ein Anwendungsbeispiel iterativer Methoden bei der Wettervorhersage siehe z.B. Steppeler et al. [2003].

5.3 Iterative Lösung nichtlinearer Gleichungssysteme

Vorlesungsnotiz: 1.12.2012
Unsere Betrachtungen der Fixpunktverfahren waren nicht auf lineare Gleichungen beschränkt. Wir wollen in diesem Abschnitt Newton–artige Verfahren definieren und ihre Konvergenz untersuchen.
Die kurze Behandlung in diesem Abschnitt wird der Bedeutung der Newton–Verfahren nicht gerecht. Tatsächlich ist das Newton–Verfahren eins der am häufigsten genutzten numerischen Verfahren, die Konvergenzanalyse ist aber recht übersichtlich.
Sei zunächst $f : \mathbb{R} \mapsto \mathbb{R}$ stetig differenzierbar. Wir suchen eine Nullstelle $\overline{x}$ von f. Dazu müssen wir zunächst

$$f(\overline{x}) = 0$$

in eine Fixpunkgleichung umwandeln. Es bietet sich an eine Formulierung wie

$$x = x - f(x) =: g(x).$$

Damit die zugehörige Fixpunktiteration konvergiert, muss gelten

$$|g'(\overline{x})| < 1 \Longleftrightarrow f'(\overline{x}) \in (0, 2).$$

Diese Bedingung legt nahe, f in der Fixpunktgleichung mit $1/f'$ zu skalieren. Dazu gibt es eine geometrische Motivation.
Sei $x^{(0)}$ eine Näherung für $\overline{x}$. Wir approximieren die Funktion f in der Nähe des Punktes $(x^{(0)}, f(x^{(0)}))$ durch ihre Tangente, und suchen statt einer Nullstelle der Funktion die Nullstelle der Tangente. Falls $x^{(0)}$ nah an $\overline{x}$ liegt, so ist diese Approximation gut. Die Tangentenfunktion hat die Darstellung

$$T(x) = f(x^{(0)}) + f'(x^{(0)}) \cdot (x - x^{(0)})$$

mit der Nullstelle

$$x^{(0)} - f'(x^{(0)})^{-1} f(x^{(0)}).$$

Wir setzen also für eine gegebene Näherung

$$x^{(k+1)} = g(x^{(k)}),\ g(x) := x - (f'(x))^{-1} f(x),$$

und erhalten die Fixpunktiteration zu g, das Newton–Verfahren zur Bestimmung einer Nullstelle von f. So, wie wir es aufgeschrieben haben, ist das Verfahren auch in höheren Dimensionen definiert (hier ist dann $f'(x)$ die Jakobimatrix).

Definition 5.30 *(**Newton–Verfahren**, auch **Newton–Raphson–Verfahren**)*
Sei $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ differenzierbar. Sei $x^{(0)} \in \mathbb{R}^n$. Falls die auftretenden Ableitungen $f'(x^{(k)})$ invertierbar sind für alle $k \in \mathbb{N}$, so heißt die Folge mit

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

Newton–Verfahren zur Bestimmung einer Nullstelle von f. Hierbei ist $f'(x)$ für $n > 1$ die Jakobimatrix von f an der Stelle x. Für $n = 1$ ist natürlich einfach

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

Der Einfachheit halber beschränken wir uns bei den Beweisen auf das eindimensionale Verfahren, die höherdimensionalen Beweise sind immer analog (aber unübersichtlicher).

Satz 5.31 *(**Konvergenz des Newtonverfahrens**)*
Sei $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ zweimal stetig differenzierbar. Sei $\overline{x}$ eine Nullstelle von f.

1. *Sei $f'(\overline{x})$ invertierbar. Dann gibt es eine Umgebung U von $\overline{x}$, so dass das Newtonverfahren*

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

   für $x^{(0)} \in U$ gegen $\overline{x}$ konvergiert (lokale Konvergenz). Die Ordnung der Konvergenz ist quadratisch.

2. *Falls $f'(\overline{x})$ nicht invertierbar ist, aber $f'(x)$ invertierbar ist in einer kleinen Umgebung von $\overline{x}$ für $x \neq \overline{x}$, ist das Newtonverfahren immer noch lokal konvergent, aber die Konvergenz ist nur noch linear.*

Beweis: Sei also $n = 1$.

1. $f'(\overline{x})$ ist invertierbar, also gibt es auch eine kleine Umgebung U' von $\overline{x}$, so dass $f'(x)$ invertierbar ist für $x \in U'$ (die Menge der nicht invertierbaren Elemente ist offen) und damit g auf U' wohldefiniert und einmal stetig differenzierbar ist.
   Es gilt

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

   also insbesondere

   $$g'(\overline{x}) = 0.$$

Damit sind alle Voraussetzungen aus 5.10 erfüllt, und das Newtonverfahren konvergiert in einer kleinen abgeschlossenen Umgebung U von $\overline{x}$.
$f'(x)$ ist stetig und invertierbar auf U, also gilt

$$C_1 = \sup_{x \in U} ||f'(x)^{-1}|| < \infty.$$

f ist zweimal stetig differenzierbar, d.h.

$$0 = f(\overline{x}) = f(x) + f'(x)(\overline{x} - x) + \frac{1}{2} f''(\xi)(\overline{x} - x)^2. \qquad (*)$$

f'' ist stetig, also gilt

$$C_2 := \sup_{x \in U} ||\frac{1}{2} f''(x)|| < \infty.$$

Für die Newton–Iteration gilt somit

$$\begin{aligned} ||\overline{x} - x^{(k+1)}|| &= ||\overline{x} - x^{(k)} + f'(x^{(k)})^{-1} f(x^{(k)})|| \\ &= ||\overline{x} - x^{(k)} + f'(x^{(k)})^{-1}[-f'(x^{(k)})(\overline{x} - x^{(k)}) \\ &\qquad - \frac{1}{2} f''(\xi)(\overline{x} - x^{(k)})^2)]|| \qquad \text{(nach *)} \\ &\leq C_1 \cdot C_2 ||\overline{x} - x^{(k)}||^2. \end{aligned}$$

2. Sei also

$$f'(\overline{x}) = f(\overline{x}) = 0.$$

Nach l'Hospital ist g stetig fortsetzbar durch $g(\overline{x}) = \overline{x}$. Der Einfachheit halber sei $f''(\overline{x}) \neq 0$. f und f' sind differenzierbar, es gilt also wieder mit Taylor

$$f(x) = f(\overline{x}) + f'(\overline{x})(x - \overline{x}) + \frac{1}{2} f''(\overline{x})(x - \overline{x})^2 + \underbrace{\frac{1}{2}(f''(\xi(x)) - f''(x))}_{=:h_1(x) \mapsto 0, x \mapsto \overline{x}} (x - \overline{x})^2$$

und ebenso

$$f'(x) = f'(\overline{x}) + f''(\overline{x})(x - \overline{x}) + (x - \overline{x}) h_2(x), \lim_{x \mapsto \overline{x}} h_2(x) = 0.$$

In höheren Dimensionen ist dies einfach nur die Definition der Ableitung.
Damit gilt für $x \neq \overline{x}$, $x \in U$, $f''(x) \neq 0$

$$\begin{aligned} g(x) - \overline{x} &= x - \overline{x} - \frac{\frac{1}{2} f''(\overline{x})(x - \overline{x})^2 + (x - \overline{x})^2 h_1(x)}{f''(\overline{x})(x - \overline{x}) + (\overline{x} - x) h_2(x)} \\ &= x - \overline{x} - \frac{\frac{1}{2} f''(\overline{x})(x - \overline{x}) + (x - \overline{x}) h_1(x)}{f''(\overline{x}) + h_2(x)}. \end{aligned}$$

Es gilt also

$$g'(\overline{x}) = \lim_{x \mapsto \overline{x}} \frac{g(x) - g(\overline{x})}{x - \overline{x}} = 1 - \frac{1}{2} < 1$$

und damit haben wir auch in diesem Fall (allerdings nur lineare) Konvergenz nach dem lokalen Konvergenzsatz und der Kontraktionskonstante $q = 1/2$. Sollte auch $f''(\overline{x}) = 0$ sein, so entwickelt man einfach noch einen Schritt in der Taylorentwicklung weiter und erhält dieselbe Aussage mit $q = 2/3$ usw. **Achtung: Dieser Beweis geht so nur durch, falls eine Ableitung verschwindet. Was passiert, falls alle Ableitungen verschwinden (etwa $e^{1/(x*x-1)}$ auf dem Rand)**

□

Für ausreichend häufig differenzierbare Funktionen kann der Beweis auch einfacher geführt werden. Mit den Beweisideen aus dem letzten Satz gilt

Korollar 5.32
Sei g p–mal stetig differenzierbar. Sei $\overline{x}$ ein Fixpunkt von g, und sei $g^{(k)}(\overline{x}) = 0$, $k = 1 \dots p-1$, $p > 1$. Dann gibt es eine Umgebung U von $\overline{x}$, so dass die Fixpunktiteration von g mit Anfangswerten in U mindestens mit der Ordnung p konvergiert.

Beweis: Wegen $g'(\overline{x}) = 0$ gibt es nach 5.10 eine abgeschlossene Umgebung U von $\overline{x}$, so dass die Fixpunktiteration konvergiert. Sei $C = \sup_{x \in U} |g^{(p)}(x)|$. Mit Taylorentwicklung gilt, da die ersten $p-1$ Ableitungen von g verschwinden,

$$g(x) = g(\overline{x}) + g^{(p)}(\xi)\frac{(x - \overline{x})^p}{p!}$$

und damit für die Fixpunktiteration

$$|x^{(k+1)} - \overline{x}| = |g(x^{(k)}) - g(\overline{x})| = \left| g^{(p)}(\xi)\frac{(x^{(k)} - \overline{x})^p}{p!} \right| \leq \frac{C}{p!}|x^{(k)} - \overline{x}|^p.$$

□

Um diesen Satz auf die Newtoniteration anzuwenden, muss g in der Newtoniteration zweimal stetig differenzierbar sein, also f dreimal stetig differenzierbar. Tatsächlich ist nicht einmal die Existenz der zweiten Ableitung notwendig, es reicht eine Lipschitzbedingung an f'. Der Vollständigkeit halber sei auch dieser Beweis hier angeführt.
Beweis: Sei $f : \mathbb{R}^n \mapsto \mathbb{R}^n$. Sei $\overline{x} \in \mathbb{R}^n$ eine Nullstelle von f, und f sei in einer abgeschlossenen Umgebung U von $\overline{x}$ einmal stetig differenzierbar und die Jakobimatrix f' sei dort invertierbar. Zusätzlich gelte

$$||f'(x)^{-1}(f'(y) - f'(x))|| \leq C||y - x||$$

(dies ist natürlich insbesondere der Fall, wenn auch f' noch einmal differenzierbar ist, dann kann die Differenz gegen $||f''||_\infty ||y-x||$ abgeschätzt werden). Nach Definition von g und wegen $f(\overline{x}) = 0$ ist für $x \in U$

$$\begin{aligned} g(x) - \overline{x} &= f'(x)^{-1} \left(f(\overline{x}) - f(x) - f'(x)(\overline{x} - x) \right) \\ &= f'(x)^{-1} \left(\int_0^1 f'(x+th)h - f'(x)h \mathrm{dt} \right). \end{aligned}$$

mit $h = \overline{x} - x$ und damit

$$||g(x) - \overline{x}|| \leq C||h||^2 = C||\overline{x} - x||^2.$$

Damit ist die Newtoniteration konvergent, falls $x^{(0)}$ nah genug an $\overline{x}$ liegt. □

Beispiel 5.33

1. *Sei $f(x) = x^n - a$, $a > 0$, $n \in \mathbb{N}$. Gesucht wird die Nullstelle $\overline{x} = a^{1/n}$ von f. Für das Newtonverfahren gilt*

$$x^{(k+1)} = x^{(k)} - \frac{(x^{(k)})^n - a}{n(x^{(k)})^{n-1}} = \frac{n-1}{n} x^{(k)} + \frac{a}{n} \frac{1}{(x^{(k)})^{n-1}} =: g(x^{(k)})$$

und

$$g'(x) = \frac{n-1}{n} - \frac{a(n-1)}{n} \frac{1}{x^n} = \frac{n-1}{n} \left(1 - \frac{a}{x^n} \right).$$

Die einzige positive Nullstelle von g' ist $\overline{x}$, und offensichtlich nimmt g für $x > 0$ dort sein Minimum $\overline{x}$ an. Es gilt also

$$g(x) \geq g(\overline{x}) = \overline{x} \, \forall x > 0.$$

Weiter ist

$$|g'(x)| < \frac{n-1}{n} < 1 \, \forall x > \overline{x}.$$

Also ist g kontrahierende Selbstabbildung auf $[\overline{x}, \infty)$ und das Newtonverfahren konvergiert mindestens für $x^{(0)} > \overline{x}$. Sei nun $0 < x^{(0)} < \overline{x}$, dann ist $x^{(1)} > \overline{x}$ und das Newtonverfahren konvergiert ebenso. Ingesamt konvergiert das Newtonverfahren also für positive Anfangswerte. Das Verfahren ist eine Möglichkeit, die n. Wurzel einer Zahl näherumgsweise zu berechnen und bekannt unter dem Namen Verfahren von Heron.

2. *Formal können wir in Beispiel 1 auch $n = -1$ setzen, also*

$$f(x) = 1/x - a,\ g(x) = x - \frac{1/x - a}{-\frac{1}{x^2}} = x + (x - ax^2) = x(2 - ax).$$

f hat den einzigen Nullpunkt $\overline{x} = 1/a$. Die Argumente aus Beispiel 1 kehren sich gerade um, $\overline{x}$ ist ein Maximum von g und für $0 < x < \overline{x}$ monoton steigend. Das Newtonverfahren konvergiert für $x \in (0, 2/a)$.
Das sieht nicht besonders sinnvoll aus – wir erhalten eine Iteration, die gegen $1/a$ konvergiert. Tatsächlich ist diese Formel schon seit einigen Jahren die wohl mit riesigem Abstand am häufigsten benutzte Anwendung des Newton–Verfahrens.
$g(x)$ benutzt nur Multiplikationen und Additionen. Wir erhalten also ein Verfahren, um den Kehrwert einer Zahl nur mit Multiplikationen und Additionen zu realisieren. Dies ist interessant für CPU–Designer, die sich damit die komplizierte Realisierung der Division in Hardware sparen können. Intel hat die genutzten Algorithmen für seine IA64–Prozessoren offengelegt in Harrison [2000], der Newtonschritt steht in 3.2. Leider ist die dort angesprochene Beispielimplementation nicht mehr verfügbar. In groben Zügen wird zunächst eine Approximation z.B. durch einen Lookup–Table gefunden, die dann durch wenige Schritte des Newton–Verfahrens verbessert wird.

3. *Für $n = 2$ betrachten wir die Funktion*

$$f : \mathbb{R}^2 \mapsto \mathbb{R}^2,\ f(x, y) = \begin{pmatrix} x - \frac{1}{4}(\cos x - \sin y) \\ y - \frac{1}{4}(\cos x - 2\sin y) \end{pmatrix}.$$

Die Jakobimatrix lautet

$$f'(x, y) = \begin{pmatrix} 1 + 1/4\sin x & 1/4\cos y \\ 1/4\sin x & 1 + 1/2\cos y \end{pmatrix}.$$

Das g aus der Newton–Iteration ist in der Nähe von $(0.16, 0.2)$ kontrahierende Selbstabbildung, also gibt es dort eine Nullstelle von f. Nach dem Satz von Gerschgorin (oder dem starken Zeilensummenkriterium) ist $f'(x, y)$ immer invertierbar, also konvergiert das Newtonverfahren bei geeignet gewählten Startwerten quadratisch. Die Iteration lautet

$$\begin{pmatrix} x^{(k+1)} \\ y^{(k+1)} \end{pmatrix} = \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix} - \frac{1}{\det f'(x^{(k)}, y^{(k)})} \cdot \begin{pmatrix} 1 + 1/2\cos y^{(k)} & -1/4\cos x^{(k)} \\ -1/4\sin x^{(k)} & 1 + 1/4\sin x^{(k)} \end{pmatrix} \cdot \begin{pmatrix} x^{(k)} - 1/4(\cos x^{(k)} - \sin y^{(k)}) \\ y^{(k)} - 1/4(\cos x^{(k)} - 2\sin y^{(k)}) \end{pmatrix}$$

mit

$$\det f'(x, y) = (1 + 1/4\sin x)(1 + 1/2\cos y) - (1/4\sin x)(1/4\cos y).$$

Bemerkung:

1. Natürlich invertiert man im $\mathbb{R}^n$ die Jakobi–Matrizen im Newtonverfahren nicht explizit sondern nutzt statt dessen

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

2. In jedem Schritt des Newton–Verfahrens muss einmal die Funktion und einmal ihre Ableitung ausgewertet werden. Im $\mathbb{R}^n$ muss zusätzlich ein Gleichungssystem gelöst werden.
Falls f' nicht explizit zur Verfügung steht, muss es durch Differenzen approximiert werden, wir berechnen also

$$\frac{df}{dx_i}(x^{(k)}) \sim \frac{f(x^{(k)}) - f(x^{(k)} + he_i)}{h}$$

mit den Einheitsvektoren e_i und berechnen daraus eine Approximation der Jakobimatrix. In diesem Fall werden $n + 1$ Funktionsauswertungen benötigt.

3. Ausdrücklich: Das Newtonverfahren ist im Allgemeinen **nicht** global konvergent. Falls die zugrundeliegende Funktion einen Nullpunkt $\overline{x}$ besitzt, so konvergiert das Newtonverfahren gegen $\overline{x}$, falls der Anfangspunkt nah genug an $\overline{x}$ liegt.

Definition 5.34 *(Varianten des Newtonverfahrens)*
f erfülle die Voraussetzungen des Newtonverfahrens.

1. *Für großes n ersetzt man im Newtonverfahren die Jakobimatrix an der Stelle $x^{(k)}$ durch die Matrix an der Stelle $x^{(0)}$ und spart sich damit die Berechnung der Ableitung und der Zerlegung für $n > 1$. Wir erhalten*

$$f'(x^{(0)})(x^{(k)} - x^{(k+1)}) = f(x^{(k)}).$$

Diese Iteration heißt **vereinfachtes Newtonverfahren**. *Das vereinfachte Newtonverfahren ist lokal linear konvergent.*

2. *Für eine Funktion $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ ist die Jakobimatrix nicht quadratisch. Es liegt daher nahe, die inverse Matrix durch die Pseudoinverse zu ersetzen. Wir erhalten die* **Gauss–Newton–Methode**

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

Dies lässt sich so motivieren: Wie bei der Definition der Kleinste Quadrate–Lösung suchen wir ein x, so dass $||f(x)||$ sein Minimum annimmt. Sei $x^{(0)}$ eine Näherung an dieses Minimum. Wir approximieren wieder f durch die Tangente und suchen das Minimum $x^{(1)}$ der Funktion

$$G(x) := ||f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)})||_2^2.$$

Damit ist $(x^{(1)} - x^{(0)})$ aber kleinste Quadrate–Lösung von

$$f'(x^{(0)})z = -f(x^{(0)}).$$

Gehen wir hier nun zur Minimum Norm–Lösung über, so erhalten wir gerade

$$x^{(1)} = x^{(0)} - f'(x^{(0)})^+ f(x^{(0)}).$$

Modifizieren wir die Funktion G wie in 4.13 zu

$$G(x) := ||f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)})||_2^2 + \gamma^2||x - x^{(0)}||_2^2$$

so bekommen wir das Verfahren

$$x^{(1)} = x^{(0)} - f'(x^{(0)})_\gamma^+ f(x^{(0)})$$

Eine geschickte Wahl von γ liefert die weitverbreiteten Levenberg–Marquardt–Verfahren nach Levenberg [1944] und Marquardt [1963].

3. *Statt durch die Tangente kann man in einer Dimension die Funktion auch durch die Verbindungsgerade (Sekante) zweier Punkte auf der Kurve approximieren. Hierzu wählt man zwei Startwerte $x^{(0)}$, $x^{(1)}$. Die Verbindungsgerade der zugehörigen Punkte auf der Kurve hat die Gleichung*

$$S(x) = f(x^{(0)}) + (f(x^{(1)}) - f(x^{(0)}))\frac{x - x^{(0)}}{x^{(1)} - x^{(0)}}.$$

Die Nullstelle dieser Geraden ist

$$x^{(0)} - \frac{x^{(1)} - x^{(0)}}{f(x^{(1)}) - f(x^{(0)})} f(x^{(0)})$$

und wir erhalten das **Sekantenverfahren**

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

Das Sekantenverfahren ist lokal konvergent mit der Konvergenzordnung $(1 + \sqrt{5})/2 \sim 1.62$ (Übungen).

4. *Durch Berücksichtigung von weiteren Termen in der Taylorentwicklung (neben der Linearisierung) kann man Verfahren höherer Ordnung herleiten.*

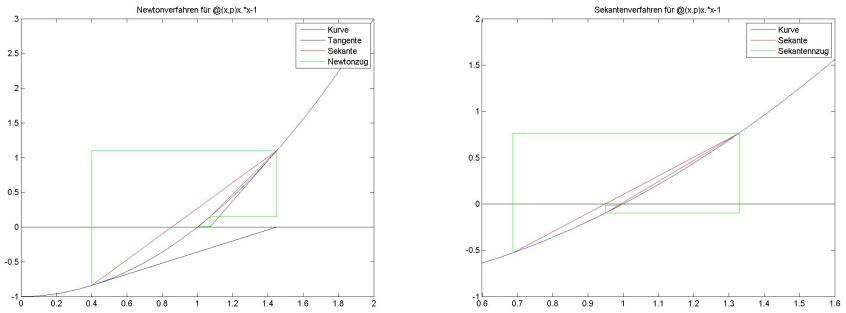

Abbildung 5.4: Newtonverfahren und Sekantenverfahren für $x^2 - 1$ und Startwert 0.7

Klick für Bild Newton
Klick für Matlab Figure Newton
Klick für Bild Sekante
Klick für Matlab Figure Sekante

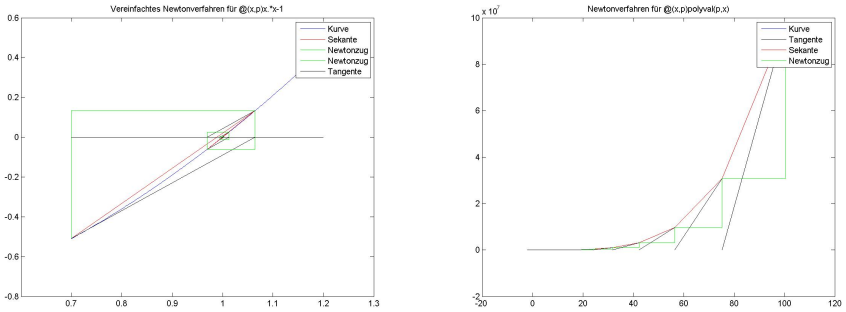

Abbildung 5.5: Vereinfachtes Newtonverfahren und typisches Verhalten bei Nicht–Konvergenz

Klick für Bild Vereinfacht
Klick für Matlab Figure Vereinfacht
Klick für Bild Newtonnoconv
Klick für Matlab Figure Newtonnoconv

```
function xo = newtonneu( f, df, p, xo, N,a,b )
%NEWTONNEU Perform N steps of Newtons algorithm.
%draw function in [a,b}.
format long
format compact
doplot=1;
```

Listing 5.11: Newtonverfahren (Newton/newtonneu.m)

Klicken für den Quellcode von Newton/newtonneu.m

```
function y = newtonneu( f, df, p, xo, N,a,b )
%NEWTONNEU Perform N steps of Newtons algorithm.
%draw function in [a,b}.
format long
x=(0:200)*(b-a)/200+a;
plot(x,f(x,p));
```

Listing 5.12: vereinfachtes Newtonverfahren (Newton/vereinfachtneu.m)

Klicken für den Quellcode von Newton/vereinfachtneu.m

```
function y = sekanteneu( f, p, xo, x1, N,a,b )
%SEKANTENEU Summary of this function goes here
%   Detailed explanation goes here
format long
x=(0:200)*(b-a)/200+a;
plot(x,f(x,p));
```

Listing 5.13: Sekantenverfahren (Newton/sekanteneu.m)

Klicken für den Quellcode von Newton/sekanteneu.m

```
function newtonneudemo(  )
%NEWTONNEUDEMO
N=40;
f=@(x,p) x.*x-1;
df=@(x,p) 2*x;
close all;
```

Listing 5.14: Steuerprogramm zum Newtonverfahren (Newton/newtonneudemo.m)

Klicken für den Quellcode von Newton/newtonneudemo.m

```
function [ output_args ] = polynewton( p,xo, x1, x2, N)
%POLYNEWTON
%  Fuehre N Newtonschritte fuer p(x)=o aus.
%  Zeichne im Intervall [x1,x2],
%  Startwert xo.
x=(0:100)*(x2-x1)/100+x1;
```

Listing 5.15: Newtonverfahren für Polynome (Newton/polynewton.m)

Klicken für den Quellcode von Newton/polynewton.m

```
function [ output_args ] = demo( input_args )
%DEMO
polynewton([1 0 -4],0.2,-3,3,10);
polynewton([1 4 5 3 2 1],0.2,-3,3,10);
polynewton(rand(10,1),0.2,-3,3,10);
polynewton(rand(10,1),0.2,-3,3,10);
```

Listing 5.16: Steuerprogramm zum Newtonverfahren für Polynome (Newton/demo.m)

Klicken für den Quellcode von Newton/demo.m

Ein Problem beim Newtonverfahren ist das Finden einer geeigneten Anfangsnäherung. Hat man eine solche, konvergiert das Newton–Verfahren meist mit wenigen Schritten. Für Polynome kann man mit Hilfe des Satzes von Gerschgorin Einschließungskriterien für die Nullstellen gewinnen (Übungen).
Global konvergente Verfahren lassen sich mit **Homotopiemethoden** gewinnen. Eine solche Methode haben wir bereits im Beweis zum Satz von Gerschgorin angewandt. Gesucht sei die Nullstelle von $f(x)$. Wir definieren eine Funktion $f(x,t)$, $t \in [0,1]$, mit den Eigenschaften:

1. $f(x,1) = f(x)$.
2. $f(x,t)$ ist zweimal stetig differenzierbar in x.
3. Die Nullstelle $x(t)$ von $f(x,t)$ hängt stetig von t ab.
4. Die Nullstelle $x(0)$ lässt sich einfach bestimmen.

Damit können wir die Nullstellen $x(t)$ verfolgen. Sei $h = 1/N$ und N fest. Ausgehend von der Nullstelle $x(0)$ bestimmen wir mit einigen Schritten des Newton–Verfahrens eine Näherung für $x(h)$. Da die Nullstellen stetig von t abhängen, wird das Newton–Verfahren schnell konvergieren, falls h klein genug ist. Ausgehend von dieser Näherung an $x(h)$ bestimmen wir dann eine Näherung an $x(2h)$ usw. bis zur Nullstelle $x(1)$ von $f(x)$.
Im Matlab–Beispiel wird eine Homotopiemethode gerechnet zur Bestimmung der Nullstellen von

$$p(x) = x^4 - 3x^3 + 5x^2 + x - 2$$

mit der Homotopiefunktion

$$f(x,t) = (1-t)(x^4 - 1) + tp(x).$$

Es illustriert das große Problem der Homotopiemethoden: Will man alle Nullstellen einer Funktion bestimmen, muss man, sobald zwei Nullstellen zusammen– und wieder auseinanderlaufen (Bifurkation), sicherstellen, dass man alle Zweige weiterverfolgt (dies ist im Programm nicht der Fall, deshalb erhält man am Ende nur drei der vier Nullstellen).

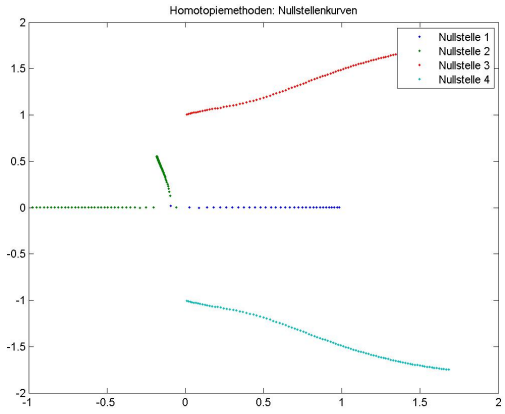

Abbildung 5.6: Nullstellen $x_k(t)$ für $f(x,t)$.

Klick für Bild Homotopy
Klick für Matlab Figure Homotopy

```
function xo=homotopyneu( f,df,fo,dfo,p,xo,N,M,a,b )
%HOMOTOPYNEU find zeros of f with homotopy method
x=(0:200)*(b-a)/200+a;
```

```
plot(x,f(x,p),x,fo(x,p));
z=zeros(numel(xo),N);
for k=1:N
```

Listing 5.17: Homotopiemethoden (Newton/homotopyneu.m)

Klicken für den Quellcode von Newton/homotopyneu.m

```
function [ output_args ] = homotopyneudemo( input_args )
%HOMOTOPYNEUDEMO
close all;
p.orig=[1 -3 5 1 2];
%p.orig=rand(5,1)*40;
p.ref =[1 0 0 0 -1];
```

Listing 5.18: Steuerprogramm zu Homotopiemethoden (Newton/homotopyneudemo.m)

Klicken für den Quellcode von Newton/homotopyneudemo.m

Kapitel 6

Krylovraumverfahrren zur Lösung linearer Gleichungen

Die heute weitaus am häufigsten genutzten Verfahren zur iterativen Lösung linearer Gleichungen sind die Krylovraumverfahren, und hier besonders das cg–Verfahren (siehe z.B. den schon zitierten Artikel Steppeler et al. [2003] für die Wettervorhersage). Wir können hier nur einen kleinen Einblick in die Thematik geben. Eine klassische Einführung mit Einblick in die Anwendung für partielle Differentialgleichungen finden Sie in Braess [2007], Kapitel 4 (Achtung: Der Text von 1992 wurde in der Auflage von 2007 stark ergänzt), eine Einordnung in allgemeine iterative Methoden für lineare Gleichungen in Saad [2003], und einen weiteren klassischen, gut geschriebenen Text in Greenbaum [1987].
Das letzte Buch hat zwei Teile: Algorithmen und Vorkonditionierer, was unsere Bemerkung zur Bedeutung der Vorkonditionierung auf Seite 112 unterstreicht. Hier werden auch die klassischen Iterationsverfahren nur als Vorkonditionierer eingeführt, so, wie wir es auf Seite 111 getan haben.
Wie immer ist dies eine persönliche Auswahl, die Zahl der Lehrbücher ist völlig unüberschaubar. Insbesondere enthält jedes Lehrbuch zur Numerischen Linearen Algebra inzwischen auch einen Abschnitt zu Krylovraumverfahren, z.B. auch Hanke-Bourgeois [2006].

6.1 Gradientenverfahren

In diesem Kapitel sei zunächst immer A eine symmetrisch positiv definite $n \times n$–Matrix. Insbesondere ist dann auf dem $\mathbb{R}^n$ das Skalarprodukt mit zugehöriger (Energie–) Norm

$$(x,y)_A = (Ax,y)\,\forall x,y \in \mathbb{R}^n,\ ||x||_A = (x,Ax)^{1/2}$$

wohldefiniert. Wir haben bereits in den Übungen gezeigt, dass für $b \in \mathbb{R}^n$ gilt

$$A\overline{x} = b \Longleftrightarrow \overline{x} = \arg\min_{x \in \mathbb{R}^n} f(x),\ f(x) := \frac{1}{2}(x, Ax) - (b, x). \tag{6.1}$$

Tatsächlich gilt für $A\overline{x} = b$

$$\frac{1}{2}(\overline{x} + x, A(\overline{x} + x)) - (b, \overline{x} + x) = \frac{1}{2}(\overline{x}, A\overline{x}) + (x, x)$$

und diese Funktion nimmt ihr eindeutiges Minimum an für $x = 0$. Wir haben also unser Problem durch ein Minimierungsproblem ersetzt.
Wir wollen dieses Problem wieder iterativ lösen. Sei also $x^{(k)}$ eine Näherung für $\overline{x}$. Wir verbessern diese Lösung, indem wir zunächst eine Suchrichtung $d^{(k)}$ und dann einen Skalar $\alpha^{(k)}$ wählen mit

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)},\ f(x^{(k+1)}) < f(x^{(k)})$$

(**line search–Verfahren**). Damit liegt $x^{(k+1)} - x^{(0)}$ in dem Untervektorraum, der von den Vektoren $d^{(0)} \dots d^{(k)}$ aufgespannt wird.
Im günstigsten Fall wählen wir $\alpha^{(k)}$ so, dass die Funktion

$$g(\alpha) = f(x^{(k)} + \alpha d^{(k)})$$

für $\alpha = \alpha^{(k)}$ ihr Minimum annimmt und $d^{(k)}$ so, dass die Funktion am Punkt $x^{(k)}$ in dieser Richtung am stärksten abfällt.
Dadurch erhalten wir die Gradientenverfahren: Wie bei Newton ersetzen wir die zu minimierende Funktion f lokal durch eine lineare Funktion. Damit ist

$$f(x + \mathsf{dx}) \sim f(x) + (\nabla f)(x)\mathsf{dx}\,.$$

Die größte Abnahme von f erreicht man also durch Wahl der Richtung $\mathsf{dx} = -\nabla f(x)$.

Definition 6.1 *(**Gradientenverfahren, Verfahren des steilsten Abstiegs, Steepest Descent**)*
Sei $f \in C^1(\mathbb{R}^n)$. Sei $x^{(0)}$ eine Schätzung für das Minimum von f, und seien $\alpha^{(k)} \in \mathbb{R}^+$. Dann heißt

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)},\ d^{(k)} = -\nabla f(x^{(k)})$$

Gradientenverfahren zur Minimierung von f.

Wir betrachten nun wieder 6.1. Wir definieren das Residuum r_k und den Approximationsfehler e_k durch

$$r^{(k)} = b - Ax^{(k)},\ e^{(k)} = \overline{x} - x^{(k)} = A^{-1}b - x^{(k)} = A^{-1}r^{(k)}.$$

Für die Funktion f aus 6.1 gilt

$$f(x + \mathsf{dx}) - f(x) = (Ax, \mathsf{dx}) - (\mathsf{dx}, b) + \frac{1}{2}(\mathsf{dx}, \mathsf{dx}) = (Ax - b, \mathsf{dx}) + \frac{1}{2}(\mathsf{dx}, \mathsf{dx})$$

und damit $d^{(k)} = -\nabla f(x^{(k)}) = b - Ax^{(k)} = r^{(k)}$ für das Gradientenverfahren.
Die Funktion

$$g(\alpha) := f(x^{(k)} + \alpha d^{(k)}) = f(x^{(k)}) + \alpha(Ax^{(k)} - b, d^{(k)}) + \frac{1}{2}\alpha^2(d^{(k)}, Ad^{(k)}) \qquad (6.2)$$

nimmt ihr Minimum an für

$$\alpha = \frac{(r^{(k)}, d^{(k)})}{(d^{(k)}, Ad^{(k)})}. \qquad (6.3)$$

Damit erhalten wir das **Gradientenverfahren zur Lösung linearer Gleichungen**

Definition 6.2 *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit, $b \in \mathbb{R}^n$, $r^{(k)} = b - Ax^{(k)}$. Dann heißt die Folge*

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}r^{(k)},\ \alpha^{(k)} = \frac{(r^{(k)}, r^{(k)})}{(r^{(k)}, Ar^{(k)})}$$

Gradientenverfahren zur Lösung von $Ax = b$.

$x^{(k+1)}$ minimiert auch $||x - \overline{x}||_A^2$ für $x \in x^{(k)} + \alpha r^{(k)}$, denn

$$G(\alpha) = ||x^{(k)} + \alpha r^{(k)} - \overline{x}||_A^2 = ||-e^{(k)} + \alpha r^{(k)}||_A^2 = ||e^{(k)}||_A^2 + 2\alpha(\underbrace{-Ae^{(k)}}_{=-r^{(k)}}, r^{(k)}) + \alpha^2||r^{(k)}||_A^2$$

nimmt ebenfalls sein Minimum für $\alpha = \alpha^{(k)}$ an. Wir bekommen also in diesem Unterraum die beste Approximation an $\overline{x}$ (gemessen in der Energienorm).

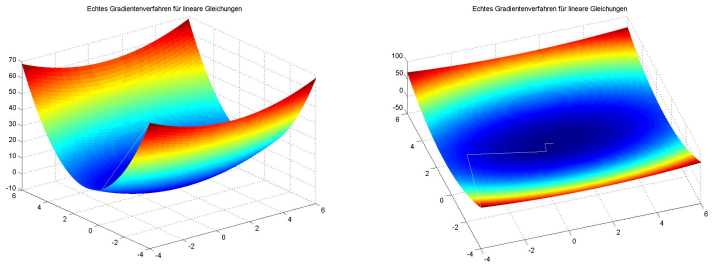

Abbildung 6.1: Surface Plot von f in 2D und Iterationsverlauf im echten Gradientenverfahren

Klick für Bild Gradientenverf
Klick für Matlab Figure Gradientenverf
Klick für Bild Gradientenverf2
Klick für Matlab Figure Gradientenverf2

```
function [ output_args ] = SteepestDescent( input_args )
%STEEPESTDESCENT
A=rand(2);
A=(A+A')/2;
lambda=min(eig(A));
if (lambda<0)
```

Listing 6.1: Gradientenverfahren (Krylov/SteepestDescent.m)

Klicken für den Quellcode von Krylov/SteepestDescent.m

Es gilt

$$r^{(k+1)} = b - Ax^{(k+1)} = b - Ax^{(k)} - \alpha^{(k)} Ad^{(k)} = (I - \alpha^{(k)} A) r^{(k)}$$

und damit

$$r^{(k+1)} = (I - \alpha^{(k)} A)(I - \alpha^{(k-1)} A) r^{(k-1)} = (I - \alpha^{(k)} A) \cdots (I - \alpha^{(0)} A) r^{(0)}$$

und somit $d^{(k)} = r^{(k)}$ Linearkombination von $r^{(0)}, \ldots, A^k r^{(0)}$. Nach der Vorbemerkung gilt damit

$$x^{(k+1)} \in x^{(0)} + \text{span}(r^{(0)}, \ldots, A^k r^{(0)}).$$

Der von den Suchrichtungen $r^{(0)}, \ldots, A^{k-1} r^{(0)}$ aufgespannte Unterraum $V^{(k)}$ heißt **Krylovraum** $\mathcal{K}^{(k)}(A, r^{(0)})$ und gibt den hier betrachteten Algorithmen seinen Namen

(nach Aleksey Krylov, geboren am 3.8.1863). Das interessante an den Krylov–Räumen ist, dass man sie durch Polynome beschreiben kann. Offensichtlich ist y genau dann in $\mathcal{K}^{(k)}(A, r^{(0)})$, wenn es ein Polynom p gibt mit

$$y = p(A)r^{(0)}.$$

Es ist also möglich, Eigenschaften der Polynome zum Beweis von Eigenschaften der Krylovräume zu nutzen, dies werden wir bei der Berechnung der Konvergenzgeschwindigkeit des cg–Verfahrens tun.

Satz 6.3 *(**Konvergenz des Gradientenverfahrens**)*
Sei $A \in \mathbb{R}^{n\times n}$ positiv definit. Dann gilt:

1. *Das Gradientenverfahren 6.2 konvergiert.*
2. *Mit der Kondition $\kappa = k(A) = ||A||_2 \cdot ||A^{-1}||_2$ gilt*

$$||e_k||_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k ||e_0||_A.$$

Beweis: Die Funktion G aus 6.1 nimmt für α_k ihr Minimum an, und es gilt

$$G(\alpha) = || - e^{(k)} + \alpha r^{(k)}||_A^2 = ||(I - \alpha A)e^{(k)}||_A^2.$$

A ist positiv definit, also gibt es eine unitäre Matrix U und eine Diagonalmatrix Σ mit $A = U^t\Sigma^2 U$. Eingesetzt in die Definition der Norm bekommen wir

$$\begin{aligned}
||(I - \alpha A)||_A^2 &= \sup \frac{((I - \alpha A)x, Ax)}{(x, Ax)} \\
&= \sup \frac{(\Sigma Ux - \alpha\Sigma^3 Ux, \Sigma Ux)}{(\Sigma Ux, \Sigma Ux)} \\
&= \sup \frac{((I - \alpha\Sigma^2)y, y)}{(y, y)},\ y = \Sigma Ux \\
&= ||I - \alpha\Sigma^2||_2^2 \\
&= \max_j |1 - \alpha\lambda_j|^2,\ \lambda_j \text{ Eigenwert von} A.
\end{aligned}$$

Also gilt

$$\begin{aligned}
||e^{(k+1)}||_A^2 &\leq G(\alpha) \\
&= ||(I - \alpha A)e^{(k)}||_A^2 \\
&\leq ||(I - \alpha A)||_A^2 ||e^{(k)}||_A^2 \\
&\leq \max_j |1 - \lambda_j| \cdot ||e^{(k)}||_A^2 \,\forall \alpha \in \mathbb{R}.
\end{aligned}$$

Seien λ_1, λ_n der größte bzw. kleinste Eigenwert von A. Es gilt

$$\kappa = ||A||_2 \cdot ||A^{-1}||_2 = \frac{\lambda_n}{\lambda_1}.$$

Wir setzen

$$\alpha = \frac{2}{\lambda_1 + \lambda_n}$$

und erhalten

$$1 - \alpha\lambda_1 = \frac{\lambda_1 + \lambda_n - 2\lambda_1}{\lambda_1 + \lambda_n} = \frac{\lambda_n - \lambda_1}{\lambda_1 + \lambda_n} = \frac{\kappa - 1}{\kappa + 1}$$

und

$$1 - \alpha\lambda_n = \frac{\lambda_1 + \lambda_n - 2\lambda_n}{\lambda_1 + \lambda_n} = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = -\frac{\kappa - 1}{\kappa + 1}.$$

Insgesamt liegt also immer $|1 - \alpha\lambda_k|$ im Intervall $[-\frac{\kappa-1}{\kappa+1}, \frac{\kappa-1}{\kappa+1}]$ und wir erhalten

$$||e^{(k+1)}||_A \leq \frac{\kappa - 1}{\kappa + 1}||e^{(k)}||_A$$

und daraus folgt die Behauptung. □

Bemerkung: Für große Konditionen bekommen wir die Kontraktionskonstante $1 - \frac{2}{\kappa+1} \sim 1$. Die Iteration ist in diesen Fällen also extrem langsam.
Dies klärt nun auch endlich den Namen Vorkonditionierer: Die Konvergenzgeschwindigkeit hängt von der Kondition ab. Vorkonditionierer verkleinern die Kondition und beschleunigen damit die Konvergenz.
Im Bild ist gut zu sehen, dass die Iterationen hin und her springen. Die scheinbar so günstige Wahl der $d^{(k)}$ liefert zwar eine konvergente, aber keine optimale Folge. Falls wir an den Krylovräumen festhalten wollen, könnten wir unsere Iterationen verbessern:
Wähle $X^{(k+1)}$ in $x^{(0)} + \mathcal{K}^{(k)}(A, r^{(0)})$ so, dass

$$||b - Ax||_2^2$$

für $x = X^{(k+1)}$ minimiert wird. Mit Hilfe der Matrix

$$W^{(k)} = \begin{pmatrix} r^{(0)} & \cdots & A^{(k-1)}r^{(0)} \end{pmatrix}$$

gilt

$$V^{(k)} = \{W^{(k)}x : x \in \mathbb{R}^k\}$$

und daher gibt es ein $z \in \mathbb{R}^k$ mit

$$X^{(k+1)} = x^{(0)} + Wz.$$

Nach Definition von $X^{(k+1)}$ minimiert z das Funktional

$$||b - Ax^{(0)} - AWx||_2^2,\ x \in \mathbb{R}^k$$

und damit ist z Minimum–Norm–Lösung von $AWz = b - Ax^{(0)}$. Hieraus lässt sich $X^{(k+1)}$ berechnen. Dies führt auf die häufig verwendeten Arnoldi– und GMRES–Verfahren (Greenbaum [1987]). Dies sind eigentlich gar keine iterativen Verfahren mehr. Wir werden zeigen, dass entweder die Suchrichtungen linear unabhängig sind oder die gesuchte Lösung bereits im Suchraum enthalten ist. Wegen der Optimalität ist damit nach spätestens n Schritten das optimale Ergebnis erreicht.
Zur Durchführung des Verfahrens müssen die QR–Zerlegungen der Matrizen $AW^{(k)}$ berechnet werden. Dies ist sehr effizient möglich (siehe Greenbaum [1987]). Wir werden aber eine noch effizientere Alternative über die konjugierte Gradienten–Methode kennenlernen.

6.2 Konjugierte Richtungen und das CG–Verfahren

Sei wieder immer $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit.

Definition 6.4 *(**konjugierte Vektoren**)*
Seien $x,\ y \in \mathbb{R}^n$,$x \neq 0 \neq y$. x und y heißen (A–) konjugiert genau dann, wenn

$$(x, y)_A = (x, Ay) = 0.$$

Satz 6.5 *Seien $d^{(0)}, \ldots, d^{(j-1)}$ paarweise konjugiert. Sei $A\overline{x} = b$. Dann gilt für das line search–Verfahren für lineare Gleichungen mit den Suchrichtungen $d^{(k)}$ und gemäß Gleichung 6.3 optimalen*

$$\alpha^{(k)} = \frac{(r^{(k)}, d^{(k)})}{(d^{(k)}, Ad^{(k)})}$$

1. *Falls $j = n$, so gilt*
$$x^{(n)} = \overline{x}.$$

2. *$x^{(k)}$ minimiert die Funktion*
$$G(x) = ||\overline{x} - x||_A$$
im Unterraum $x^{(0)} +$ span $(d^{(0)}, \ldots, d^{(k-1)})$.

3. *$x^{(k)}$ minimiert die Funktion*
$$f(x) = \frac{1}{2}(x, Ax) - (x, b)$$
im Unterraum $x^{(0)} +$ span $(d^{(0)}, \ldots, d^{(k-1)})$.

Wählt man die Suchrichtungen also konjugiert, bricht das Verfahren spätestens für $x^{(n)}$ mit der exakten Lösung ab. Zum Beweis zeigen wir zunächst ein ganz kleines Lemma, das im übernächsten Kapitel noch nützlich sein wird.

Lemma 6.6 *(***Bestapproximation bzgl. der induzierten Norm***)*
Sei V ein euklidischer Raum mit Skalarprodukt $(,)$, $x \in V$ und $v^{(k)}$, $k = 0 \dots n-1$, ein Orthonormalsystem in V. Dann gilt:

1. *Falls $\dim V = n$, so ist*
$$x = \sum_{k=0}^{n-1} (x, v^{(k)}) v^{(k)}.$$
2. *Sei $j \leq n$.*
$$y^{(j)} = \sum_{k=0}^{j-1} (x, v^{(k)}) v^{(k)}$$
minimiert die Funktion
$$G(z) = ||x - z||$$
für $z \in$ span $(v^{(0)}, \dots, v^{(k-1)})$. $y^{(j)}$ ist also die beste Approximation an x in diesem Unterraum.

Beweis: (des Lemmas)

1. $v^{(0)}, \dots, v^{(n-1)}$ ist Basis von V, also gibt es $\alpha^{(k)}$ mit $x = \sum_k \alpha^{(k)} v^{(k)}$.
$$(v^{(l)}, x) = (v^{(l)}, \sum_{k=0}^{n-1} \alpha^{(k)} v^{(k)}) = (v^{(l)}, \alpha^{(l)} v^{(l)}) = \alpha^{(l)}$$
und daraus folgt die Behauptung.
2. Für $l \leq j$ gilt
$$(x - y^{(j)}, v^{(l)}) = (x - \sum_{k=0}^{j-1} (x, v^{(k)}) v^{(k)}, v^{(l)}) = (x, v^{(l)}) - (x, v^{(l)})(v^{(l)}, v^{(l)}) = 0.$$
Sei $z \in$ span $(v^{(0)}, \dots, v^{(j-1)})$. Dann gilt $(x - y^{(j)}, y^{(j)} - z) = 0$ und
$$||x - z||^2 = ||(x - y^{(j)}) + (y^{(j)} - z)||^2 = ||x - y^{(j)}||^2 + ||y^{(j)} - z||^2 \geq ||x - y^{(j)}||^2$$
und das war die Behauptung.

□

Beweis: (des Satzes)

1. Die Vektoren $d^{(k)}/||d^{(k)}||_A$ sind eine Orthonormalbasis des $\mathbb{R}^n$ bzgl. $(,)_A$, also gilt mit dem Lemma

$$\begin{aligned}
\overline{x} - x^{(0)} &= \sum_{k=0}^{n-1} (\overline{x} - x^{(0)}, \frac{d^{(k)}}{||d^{(k)}||_A})_A \frac{d^{(k)}}{||d^{(k)}||_A} \\
&= \sum_{k=0}^{n-1} (\overline{x} - x^{(k)}, \frac{d^{(k)}}{||d^{(k)}||_A})_A \frac{d^{(k)}}{||d^{(k)}||_A} \text{ denn } x^{(k)} - x^{(0)} \in < d^{(0)} \dots d^{(k-1)} > \\
&= \sum_{k=0}^{n-1} \frac{(A\overline{x} - Ax^{(k)}, d^{(k)})}{||d^{(k)}||_A^2} d^{(k)} \\
&= \sum_{k=0}^{n-1} \frac{(r^{(k)}, d^{(k)})}{(d^{(k)}, Ad^{(k)})} d^{(k)} \\
&= \sum_{k=0}^{n-1} \alpha^{(k)} d^{(k)}
\end{aligned}$$

   und damit $x^{(n)} = \overline{x}$.

2. Mit dem zweiten Teil des Lemmas und derselben Rechnung.

3. Übungen.

□

Da dies immer wieder missverstanden wird, hier direkt eine Warnung: Wir denken **nicht** daran, Teil 1 von 6.5 wirklich zu nutzen und bis zum n. Schritt zu iterieren (ansonsten könnten wir gleich das Gleichungssystem mit direkten Verfahren lösen). Viel interessanter ist Teil 2, der uns garantiert, in den untersuchten Teilräumen die beste Approximation unserer Lösung zu finden (bezüglich der Energienorm $|| \cdot ||_A$). Wir möchten also ein line search–Verfahren definieren mit den folgenden Eigenschaften:

1. Die Suchrichtungen sollten zueinander konjugiert sein.

2. Der Raum, den die Suchrichtungen aufspannen, sollte derselbe sein wie beim Gradientenverfahren.

Es stellt sich also die Frage: Können wir Vektoren $d^{(0)}, \ldots, d^{(k-1)}$ einfach so wählen, dass sie zueinander konjugiert sind und den Krylovraum $\mathcal{K}^{(k)}(A, r^{(0)})$ aufspannen? Tatsächlich tut das cg–Verfahren dies mit einer einfachen Rekursion.
Die Grundidee ist, wie im Gradientenverfahren als Abstiegsrichtungen $r^{(0)}$, $r^{(1)}$ usw. zu wählen, aber die Richtungen mit dem Schmidschen Orthogonalisierungsverfahren bzgl. des Skalarprodukts $(,)_A$ zu orthogonalisieren.
Wir starten mit

Lemma 6.7 *Seien $d^{(k)}$ paarweise konjugiert. Dann gilt*

$$(d^{(i)}, r^{(k)}) = 0 \, \forall i < k,$$

also bezüglich des euklidischen Skalarprodukts

$$r^{(k)} \perp \mathcal{K}^{(k)}(A, r^{(0)}).$$

Außerdem gilt bezüglich des A–Skslarprodukts

$$r^{(k)} \perp_A \mathcal{K}^{(k-1)}(A, r^{(0)}).$$

Beweis: Zunächst gilt

$$\begin{aligned}(d^{(k)}, r^{(k+1)}) &= (d^{(k)}, b - Ax^{(k+1)}) \\ &= (d^{(k)}, b - A(x^{(k)} + \frac{(r^{(k)}, d^{(k)})}{(d^{(k)}, Ad^{(k)})} d^{(k)})) \\ &= (d^{(k)}, r^{(k)}) - (d^{(k)}, Ad^{(k)}) \cdot \frac{(r^{(k)}, d^{(k)})}{(d^{(k)}, Ad^{(k)})} = 0.\end{aligned}$$

Daraus folgt wegen

$$r^{(k+2)} = b - Ax^{(k+2)} = b - A(x^{(k+1)} + \alpha^{(k+1)} d^{(k+1)}) = r^{k+1} - \alpha^{(k+1)} Ad^{(k+1)}$$

und

$$\begin{aligned}(d^{(k)}, r^{(k+2)}) &= (d^{(k)}, r^{(k+1)} - \alpha^{(k+1)} Ad^{(k+1)}) \\ &= -\alpha^{(k+1)} (d^{(k)}, Ad^{(k+1)}) = 0\end{aligned}$$

usw. per Induktion.
Nach Definition der Krylovräume gilt

$$A\mathcal{K}^{(k-1)}(A, r^{(0)}) \subset \mathcal{K}^{(k)}(A, r^{(0)}).$$

Sei also $y \in \mathcal{K}^{(k-1)}(A, r^{(0)})$, so gilt

$$(r^{(k)}, y)_A = (r^{(k)}, \underbrace{Ay}_{\in \mathcal{K}^{(k)}(A, r^{(0)})}) = 0.$$

□

Mit diesem Satz bekommen wir auch einen leichten Beweis des letzten Teils von 6.7. Da

$$x^{(k)} \in x^{(0)} + \text{span}\,(d^{(0)}, \ldots, d^{(k-1)}),$$

sind alle Elemente aus $x^{(0)} + \text{span}\,(d^{(0)}, \ldots, d^{(k-1)})$ von der Form

$$x^{(k)} + w,\ w \in \text{span}\,(d^{(0)}, \ldots, d^{(k-1)}).$$

Dann gilt aber

$$f(x^{(k)} + w) = f(x^{(k)}) + \underbrace{(Ax^{(k)}, w) - (b, w)}_{=(-r^{(k)}, w)=0} + \frac{1}{2}(w, Aw) \geq f(x^{(k)}).$$

Wir wenden nun das Schmidtsche Orthogonalisierungsverfahren bezüglich des Skalarprodukts $(,)_A$ auf die $r^{(k)}$ an. Dies wird uns Vektoren liefern, die zueinander konjugiert sind und dieselben Vektorräume aufspannen wie die $r^{(k)}$, also gerade die Krylovräume. Wir setzen also $d^{(0)} = r^{(0)}$ und

$$\begin{aligned} d^{(k+1)} &= r^{(k+1)} - \sum_{i=0}^{k} \frac{(d^{(i)}, r^{(k+1)})_A}{||d^{(i)}||_A^2} d^{(i)} \\ &= r^{(k+1)} - \frac{(d^{(k)}, r^{(k+1)})_A}{||d^{(k)}||_A^2} d^{(k)} \\ &= r^{(k+1)} - \frac{(Ad^{(k)}, r^{(k+1)})}{(d^{(k)}, Ad^{(k)})} d^{(k)}. \end{aligned}$$

Die Terme in der Summe verschwinden wegen 6.7.
Damit erhalten wir

Definition 6.8 cg-Verfahren *(**Verfahren der konjugierten Gradienten**)*
Sei $A \in \mathbb{R}^{n \times n}$ positiv definit. Seien $b,\ x^{(0)} \in \mathbb{R}^n$. Dann heißt die Folge $x^{(k)}$, definiert durch

$$\begin{aligned} r^{(0)} &= b - Ax^{(0)},\ d^{(0)} = r^{(0)} \\ \alpha^{(k)} &= \frac{(d^{(k)}, r^{(k)})}{(d^{(k)}, Ad^{(k)})} \\ x^{(k+1)} &= x^{(k)} + \alpha^{(k)} d^{(k)} \\ r^{(k+1)} &= b - Ax^{(k+1)} = r^{(k)} - \alpha^{(k)} Ad^{(k)} \\ \beta^{(k+1)} &= \frac{(d^{(k)}, r^{(k+1)})_A}{(d^{(k)}, d^{(k)})_A} = \frac{(d^{(k)}, Ar^{(k+1)})}{(d^{(k)}, Ad^{(k)})} \\ d^{(k+1)} &= r^{(k+1)} - \beta^{(k+1)} d^{(k)}. \end{aligned}$$

Verfahren der konjugierten Gradienten. Die Folge endet, sobald $d^{(k)} = 0$.

Tatsächlich kann man diese Definition noch etwas stabiler und effizienter hinschreiben.
Für die Eigenschaften müssen wir nur die bereits bewiesenen Sätze zusammentragen.

Korollar 6.9 *(Eigenschaften des cg–Verfahrens)*

1. *Solange $r^{(k)} \neq 0$, ist auch $d^{(k)} \neq 0$, d.h. das Verfahren bricht genau dann ab, wenn die korrekte Lösung erreicht ist.*
2. *$x^{(k)}$ ist Bestapproximation an die Lösung von $Ax = b$ im Raum $x^{(0)} + \mathcal{K}^{(k)}(A, r^{(0)})$, und minimiert dort auch die Funktion f aus 6.1.*

Beweis: Das cg-Verfahren ist gerade so konstruiert, dass die $d^{(k)}$ konjugiert sind und den Raum $\mathcal{K}^{(k)}(A, r^{(0)})$ aufspannen, daraus folgt die zweite Aussage nach 6.5. Nach 6.7 gilt $d^{(k)} \perp r^{k+1}$, falls also $d^{k+1} = r^{(k+1)} - \beta^{(k)} d^{(k)} = 0$, so ist bereits $r^{(k+1)} = 0$ und umgekehrt. □

Natürlich müssen wir die Konvergenz des cg–Verfahrens nicht zeigen - es bricht spätestens nach n Schritten ab. Für die Konvergenzgeschwindigkeit gilt, dass sie mindestens so gut sein muss wie für das Gradientenverfahren 6.2, denn die dort gelieferten Folgenglieder liegen in den Krylovräumen, in denen das cg–Verfahren eine optimale Wahl liefert.
Wir erwarten aber natürlich, dass das cg–Verfahren bessere Konvergenzgeschwindigkeit liefert. Dies ist tatsächlich der Fall. Wir beginnen mit

Lemma 6.10
Sei $A \in \mathbb{R}^{n \times n}$ s.p.d., also hat $\mathbb{R}^n$ eine ONB $v_1, \ldots, v_n$ aus Eigenvektoren von A zu Eigenwerten $\lambda_1, \ldots, \lambda_n$ (zum euklidischen Skalarprodukt).
Sei p ein Polynom vom Grad $\leq k$ mit

$$p(0) = 1 \text{ und } |p(\lambda_j)| \leq r \, \forall j.$$

Dann gilt für das cg–Verfahren

$$||e^{(k)}||_A \leq r||e^{(0)}||_A.$$

Beweis: Wir nutzen den schon angesprochenen Zusammenhang zwischen Polynomen und den Krylovräumen.

1. Es gilt $p(0) - 1 = 0$, also ist

$$q(z) := (p(z) - 1)/z$$

Polynom vom Grad $\leq k - 1$ und es ist

$$1 + zq(z) = p(z).$$

Setzen wir wieder formal die Matrix A ein, so gilt entsprechend

$$I + Aq(A) = p(A).$$

2. Sei

$$y := x^{(0)} - q(A)r^{(0)}.$$

Dann gilt

$$y \in x^{(0)} + \mathcal{K}^{(k)}(A, r^{(0)})$$

und

$$\begin{aligned} \overline{x} - y &= A^{-1}b - x^{(0)} + q(A)AA^{-1}r^{(0)} \\ &= (I + Aq(A))e^{(0)} \\ &= p(A)e^{(0)}. \end{aligned}$$

3. Sei $e^{(0)} = \sum_{i=1}^{n} \alpha_i v_i$. Dann ist

$$\begin{aligned} \overline{x} - y &= p(A)\left(\sum_{i=1}^{n} \alpha_i v_i\right) \\ &= \sum_{i=1}^{n} \alpha_i p(\lambda_i) v_i \end{aligned}$$

und

$$\begin{aligned}
||\overline{x} - y||_A^2 &= ||\sum_{i=1}^{n} \alpha_i p(\lambda_i) v_i||_A^2 \\
&= \left(\sum_{i=1}^{n} \alpha_i p(\lambda_i) v_i, \sum_{i=1}^{n} \alpha_i p(\lambda_i) \lambda_i v_i \right) \\
&= \sum_{i=1}^{n} \lambda_i (p(\lambda_i)\alpha_i)^2 \\
&\leq \sum_{i=1}^{n} \lambda_i r^2 \alpha_i^2 \\
&= r^2 \left(\sum_{i=1}^{n} \alpha_i v_i, A \sum_{i=1}^{n} \alpha_i v_i \right) \\
&= r^2 ||\overline{x} - x_0||_A^2 = r^2 ||e_0||_A^2.
\end{aligned}$$

4. Da $x^{(k)}$ die Bestapproximation in $x^{(0)} + \mathcal{K}^{(k)}(A, r^{(0)})$ ist, gilt

$$||e^{(k)}||_A = ||x^{(k)} - \overline{x}||_A \leq ||y - \overline{x}||_A \leq r||e^{(0)}||_A.$$

□

Satz 6.11 *(**Konvergenzgeschwindigkeit des cg–Verfahrens**)*
Für das cg–Verfahren gilt

$$||e^{(k)}||_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k ||e^{(0)}||_A.$$

Beweis: Wir nutzen die Tschebyscheff–Polynome

$$T_k(x) = \cos(k \arccos x).$$

Sie besitzen mit der Eulerschen Formel die alternative Darstellung

$$\begin{aligned}
T_k(x) &= \frac{1}{2}(e^{ik \arccos x} + e^{-ik \arccos x}) \\
&= \frac{1}{2}((\cos(\arccos x) + i \sin(\arccos x))^k + (\cos(\arccos x) + i \sin(\arccos x))^{-k}) \\
&= \frac{1}{2}((x + i\sqrt{1 - x^2})^k + (x + i\sqrt{1 - x^2})^{-k}) \\
&= \frac{1}{2}((x + \sqrt{x^2 - 1})^k + (x + \sqrt{x^2 - 1})^{-k})
\end{aligned}$$

(für positives Vorzeichen des $\sin$ und mit einer losen Definition von $\sqrt{-a} = i\sqrt{a}$). Diese Darstellung ist auch außerhalb von $[-1, 1]$ gültig. Dort gilt dann

$$|T_k(x)| \geq \frac{1}{2}(x + \sqrt{x^2 - 1})^k. \tag{6.4}$$

Diese überschlägige Motivation mag hier genügen, korrekt weist man nach, dass der Ausdruck auf der rechten Seite der rekursiven Definition der Tschebyscheff–Polynome aus den Übungen genügt.
In den Übungen wurde auch bereits gezeigt, dass $T_k(x)$ Polynom vom Grad k ist. Nach Definition über den $\arccos$ ist klar, dass $|T_k(x)| \leq 1$ für $x \in [-1, 1]$. Seien wieder λ_1 und λ_n der kleinste bzw. größte Eigenwert von A. Wir skalieren das Argument von T_k so, dass es für x zwischen λ_1 und λ_n zwischen -1 und 1 liegt:

$$T(x) = T_k(1 - 2\frac{x - \lambda_1}{\lambda_n - \lambda_1})$$

und damit

$$|T(x)| \leq 1 \, \forall \lambda_1 \leq x \leq \lambda_n.$$

Nun definieren wir p so, dass $p(0) = 1$:

$$p(x) = \frac{T(x)}{T(0)}$$

wobei

$$T(0) = T_k\left(\frac{\lambda_n - \lambda_1}{\lambda_n - \lambda_1} + 2\frac{\lambda_1}{\lambda_n - \lambda_1}\right) = T_k\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right) = T_k\left(\frac{\kappa + 1}{\kappa - 1}\right).$$

Für $x \in [\lambda_1, \lambda_n]$ gilt also

$$|p(x)| \leq \frac{1}{T(0)} = \frac{1}{T_k(\frac{\kappa+1}{\kappa-1})}$$

(zur Erinnerung: κ war die Kondition von A in der euklidischen Norm, also gerade λ_n/λ_1). Im Nenner wollen wir die Abschätzung 6.4 einsetzen. Mit

$$\frac{\kappa + 1}{\kappa - 1} + \sqrt{\left(\frac{\kappa + 1}{\kappa - 1}\right)^2 - 1} = \frac{\kappa + 1 + 2\sqrt{\kappa}}{\kappa - 1} = \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}$$

und wegen $(\kappa + 1)/(\kappa - 1) > 1$ gilt

$$T(0) = T_k\left(\frac{\kappa + 1}{\kappa - 1}\right) \geq \frac{1}{2}\left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^k.$$

Damit gilt

$$|p(\lambda_j)| \leq \frac{1}{T(0)} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k$$

und daraus folgt die Behauptung mit 6.10. □

Dies sieht zunächst noch nicht sehr beeindruckend aus. Tatsächlich ist diese Abschätzung für viele Probleme viel zu pessimistisch (was man allein schon daran sieht, dass der Fall $k > n$, für den die Norm des Residuums verschwindet, natürlich gar nicht korrekt abgebildet wird).
Die Idee des cg–Verfahrens ist eigentlich leicht zu merken:

1. Es wird ein normales line search–Verfahren durchgeführt, d.h. die Funktion f wird in jedem Iterationsschritt auf Geraden durch die letzte Iterierte minimiert.
2. Die Richtungen dieser Geraden werden zueinander orthogonal gewählt bezüglich des A–Skalarprodukts (konjugiert). Dadurch minimiert man nicht nur auf einer Geraden, sondern im gesamten von den Geradenrichtungen aufgespannten Teilraum.
3. Dies garantiert man, indem man das Schmidtsche Orthogonalisierungsverfahren auf die Richtungen des Gradientenverfahrens anwendet.
4. Es stellt sich heraus, dass bei der Orthogonalisierung fast alle Terme wegfallen, so dass sie sehr einfach berechenbar ist.

Man nutzt wieder nicht das Problem $Ax = b$ direkt, sondern implementiert

$$BAx = Bb$$

für eine einfach zu invertierende Matrix B mit der Eigenschaft, dass BA möglichst kleine Kondition hat. Dies ist nicht problemlos, denn BA ist nur symmetrisch, wenn B und A vertauschen. Am einfachsten löst man dies durch Nutzung der beidseitigen Vorkonditionierung

$$BAB^t((B^t)^{-1}x) = Bb.$$

Eine genaue Analyse des cg–Algorithmus zeigt aber, dass eine leicht angepasste Variante des cg–Verfahrens auch für $BAx = Bb$ optimale Ergebnisse im Krylovraum liefert mit derselben Konvergenzgeschwindigkeit Braess [2007]. Es sei noch einmal darauf hingewiesen, dass erst die Wahl geeigneter Vorkonditionierer die hier vorgestellten Methoden wirklich effizient macht.

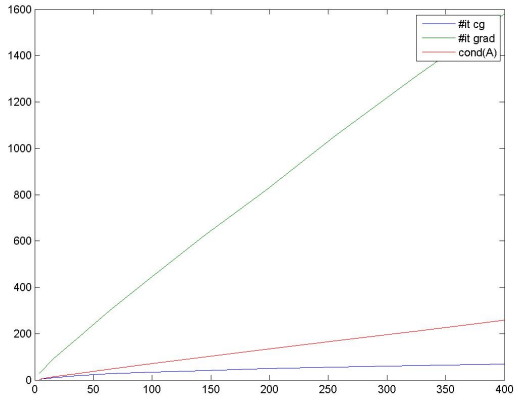

Abbildung 6.2: Vergleich der Iterationszahlen von cg und Gradientenverfahren

Klick für Bild vergleichcg
Klick für Matlab Figure vergleichcg

```
function [ x,n ] = cg( A,b,xo,eps )
%CG cg-Verfahren wie in der Aufgabe
if (nargin<1)
    A=setupmatrix(10);
end
if (nargin<2)
```

Listing 6.2: Konjugierte Gradienten (Krylov/cg.m)

Klicken für den Quellcode von Krylov/cg.m

```
function [ x,n ] = lingrad( A,b,xo,eps )
%lingrad Gradienten-Verfahren wie in der Aufgabe
if (nargin<1)
    A=setupmatrix(10);
end
if (nargin<2)
```

Listing 6.3: Gradientenverfahren (Krylov/lingrad.m)

Klicken für den Quellcode von Krylov/lingrad.m

```
function cgdemo
%CGDEMO
for i=1:10
    n=i*2;
    n
    A=setupmatrix(n);
```

Listing 6.4: Treiber zu cg (Krylov/cgdemo.m)

Klicken für den Quellcode von Krylov/cgdemo.m

```
function A = setupmatrix( N )
%SETUP_MATRIX setup matrix of discretized Laplace operator in 2D
if (nargin<1)
    N=10;
end
lambda=0;
```

Listing 6.5: Aufstellung der Matrix (Krylov/setupmatrix.m)

Klicken für den Quellcode von Krylov/setupmatrix.m

6.3 Der Uzawa–Algorithmus: Optimierung mit Nebenbedingungen

Abschließend schauen wir noch auf eine häufig auftretende Modifikation unserer Minimierungsaufgabe aus Gleichung 6.1. Sei wieder $A \in \mathbb{R}^{n\times n}$ s.p.d., $b \in \mathbb{R}^n$, $B \in \mathbb{R}^{m\times n}$ vom Rang $m < n$ und $g \in \mathbb{R}^m$. Wir suchen das Minimum nur unter den Vektoren mit $Bx = g$, also:
Suche $\overline{x} \in \mathbb{R}^n$ mit

$$\overline{x} = \arg\min_{Bx=g} f(x),\ f(x) = \frac{1}{2}(Ax, x) - (b, x). \tag{6.5}$$

Statt dessen können wir auch das Gleichungssystem

$$\begin{pmatrix} A & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \overline{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ g \end{pmatrix} \tag{6.6}$$

lösen. Sei nämlich $(\overline{x}, \lambda)$ eine Lösung von 6.6. Wir betrachten $f(\overline{x}+x)$ mit $B(\overline{x}+x) = g$, also $Bx = 0$:

$$\begin{aligned} f(\overline{x}+x) &= \frac{1}{2}(A(\overline{x}+x),\overline{x}+x) - (b,\overline{x}+x) \\ &= f(\overline{x}) + (A\overline{x},x) + (x,Ax) - (b,x) \\ &\geq f(\overline{x}) + (A\overline{x}-b,x) \\ &= f(\overline{x}) - (B^t\lambda,x) \\ &= f(\overline{x}) - (\lambda,Bx) = f(\overline{x}). \end{aligned}$$

Wir notieren

Korollar 6.12 *Sei $\overline{x} \in \mathbb{R}^n$. Falls es ein $\lambda \in \mathbb{R}^m$ gibt, so dass $(\overline{x},\lambda)$ 6.6 lösen, so löst $\overline{x}$ auch das Minimierungsproblem 6.5.*

Uzawa schreibt dies als Fixpunktgleichung. Sei $\widehat{C}$ leicht invertierbare positiv definite Matrix im $\mathbb{R}^{m\times m}$ (z.B. eine Diagonalmatrix oder ein Vielfaches der Einheitsmatrix).

$$\begin{pmatrix}\overline{x}\\ \lambda\end{pmatrix} = \begin{pmatrix} A^{-1}(b-B^t\lambda) \\ \lambda + \widehat{C}^{-1}(B\overline{x}-g)\end{pmatrix}.$$

Die dadurch definierte Fixpunktiteration für λ lautet

$$\lambda^{(k+1)} = \lambda^{(k)} + \widehat{C}^{-1}(BA^{-1}(b-B^t\lambda^{(k)})-g).$$

Dieses Verfahren ist konvergent, falls die Eigenwerte der Iterationsmatrix

$$G = I - \widehat{C}^{-1}BA^{-1}B^t$$

zum Betrag kleiner als 1 sind. Sei u ein Eigenvektor von G zum Eigenwert μ. Dann gilt

$$\widehat{C}u - BA^{-1}B^tu = \mu\widehat{C}u$$

und damit

$$(1-\mu)\underbrace{(u,\widehat{C}u)}_{>0} = \underbrace{(u,BA^{-1}B^tu)}_{>0}$$

und damit $\mu < 1$. Wir wählen nun $\widehat{C}$ so groß, dass $\widehat{C} - BA^{-1}B^t$ s.p.d. ist. Dann liegen alle Eigenwerte von G zwischen 0 und 1, und das Uzawa–Verfahren ist konvergent.
Üblicherweise definiert man noch eine zusätzliche s.p.d. Vorkonditionsmatrix $C \in \mathbb{R}^{n\times n}$. Dann gilt

$$B^tC^{-1}B\overline{x} = B^tC^{-1}g$$

und nutzt zur Definition des Augmented Lagrangian–Verfahrens die Fixpunktiteration zu

$$\begin{pmatrix} \overline{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} (A + BC^{-1}B^t)^{-1}(b + B^tC^{-1}g - B^t\lambda) \\ \lambda + \widehat{C}^{-1}(B\overline{x} - g) \end{pmatrix}.$$

Die Gleichung in λ lässt sich auch als Gradientenverfahren interpretieren. Also hat das Uzawa–Verfahren in dieser Form dieselben Probleme wie das normale Gradientenverfahren. Mit denselben Methoden wie beim cg–Verfahren lässt sich auch hier eine Methode der konjugierten Richtungen entwickeln (Braess [2007]).

Kapitel 7

Numerische Berechnung von Eigenwerten

Die Berechnung der Eigenwerte einer Matrix spielt in der Numerik eine große Rolle, z.B.

1. Bestimmung optimaler Iterationsparameter (max. Eigenwert einer hermiteschen Matrix).

2. Bestimmung der Kondition einer Matrix (wie oben, zusätzlich Bestimmung des kleinsten Eigenwerts).

3. Bestimmung von Eigenschwingungen einer Brücke. Dies lässt sich (stark vereinfacht) so erklären: Eine Brücke reagiere auf eine Belastung L von außen mit einer Stressverteilung $p = AL$. Falls x ein Eigenvektor zu einem Eigenwert von A größer als 1 ist, so wird die wirkende Belastung durch die Brücke nicht verteilt (als Gegendruck), sondern sogar noch verstärkt. Die Belastung kann sich also immer weiter aufbauen.
   In der Matlab–Demo truss lässt sich das an einem sehr einfachen zweidimensionalen Beispiel beobachten. Insbesondere sieht man, dass für höhere Moden (kleinere Eigenwerte) die Eigenschwingungen eine komplexe Struktur zeigen. Eine genauere Analyse dieses Beispiels finden Sie in Hanke-Bourgeois [2006], Kapitel V.22.
   Diese Untersuchung ist keineswegs akademisch. Immer wieder gern zitiertes Standardbeispiel ist die Tacoma Narrows Bridge, bei der eine (gar nicht so große) kontinuierliche Windanregung in der falschen Frequenz zu großen Auslenkungen und letztlich zur Zerstörung der Brücke führte. Der Film zeigt, dass die Brücke keineswegs nur einfach schwingt, sondern zusätzlich eine Torsionsstruktur hat (wie wir sie nach der Matlab–Analyse erwarten würden). Eine mehrmals überarbeitete mathematische Untersuchung dieser Zerstörung finden Sie unter anderem in McKenna [1999].

Aus diesem Grund sind Eigenwertanalysen in der Statik unerlässlich. Die Tatsache, dass es nur wenige Beispiele für solche Komplettzerstörungen gibt, zeigt, dass dieses Problem gelöst ist (und andererseits dieses Phänomen nur recht selten auftritt).

Wir bemerken, dass insbesondere ein Interesse daran besteht, große oder kleine Eigenwerte von (hermiteschen) Matrizen zu berechnen.
Zur Motivation beginnen wir mit einer Idee zur Berechnung eines Eigenvektors, die uns später zu allen numerischen Verfahren führen wird. Wir wollen natürlich wieder iterativ vorgehen. Sei A eine $n \times n$–Matrix. Da auch bei reellen Matrizen Eigenwerte und Eigenvektoren komplex sein könnten, ist es bei allgemeinen Matrizen keine Vereinfachung, sich auf reelle Matrizen zu beschränken, wir nehmen diese komplex an. Bei hermiteschen beschränken wir uns der Einfachheit halber wieder auf reelle (symmetrische) Matrizen.
Wie bei den Krylov–Räumen starten wir mit einem Vektor $x^{(0)}$ und wenden Potenzen der $n \times n$–Matrix A auf $x^{(0)}$ an. Wir setzen

$$x^{(j)} = A^j x^{(0)}, \text{ also } x^{(j+1)} = Ax^{(j)}.$$

Zur Vereinfachung nehmen wir zunächst an, dass der $\mathbb{C}^n$ eine Basis aus Eigenvektoren y_k zu Eigenwerten λ_k, $k = 1 \dots n$, besitzt. In allen Betrachtungen seien unsere Eigenwerte immer der Größe des Betrages nach geordnet, d.h. $|\lambda_k| \geq |\lambda_{k+1}|$. Es sei

$$x^{(0)} = \sum_{k=1}^{n} \alpha_k v_k$$

mit $\alpha_1 \neq 0$. Dann ist

$$x^{(j)} = A^j \sum_{k=1}^{n} \alpha_k v_k = \sum_{k=1}^{n} \alpha_k \lambda_k^j v_k = \lambda_1^j \underbrace{\left(\alpha_1 v_1 + \sum_{k=2}^{n} \alpha_k \left(\frac{\lambda_k}{\lambda_1} \right)^j v_k \right)}_{=:w^{(j)}} \tag{7.1}$$

Es sei nun λ_1 der betragsmäßig echt größte Eigenwert, d.h. $|\lambda_1| > |\lambda_2|$. Dann ist klar, dass in der Darstellung 7.1 die Summe für $j \mapsto \infty$ verschwindet, also $w^{(j)}$ gegen $\alpha_1 v_1$ konvergiert. Für große j wird also $x^{(j)}$ zu einem Vielfachen des ersten Eigenvektors y_1. Um zu einer Konvergenz zu kommen, müsste man nun nur noch die $x^{(j)}$ normieren, was den Faktor vor $w^{(j)}$ eliminiert. Hierzu kann man z.B. einen Vektor $\widetilde{x}$ fest wählen und die Vektoren

$$y^{(j)} = \frac{1}{(x^{(j)}, \widetilde{x})} x^{(j)} = \frac{1}{\lambda_1^j (w^{(j)}, \widetilde{x})} \lambda_1^j w^{(j)} \to_{j \mapsto \infty} \frac{1}{(v_1, \widetilde{x})} v_1$$

betrachten (mit der Voraussetzung, dass $(\widetilde{x}, v_1) \neq 0$). Für die Bestimmung des Eigenwerts betrachtet man entsprechend den Quotienten

$$a^{(j)} = \frac{(x^{(j+1)}, \widetilde{x})}{(x^{(j)}, \widetilde{x})} = \frac{\lambda_1^{j+1}(w^{(j+1)}, \widetilde{x})}{\lambda_1^{j}(w^{(j)}, \widetilde{x})} \to_{j \mapsto \infty} \lambda_1$$

mit derselben Bedingung. Dies ist die Potenzmethode (Vektoriteration). Wir werden diese später genauer untersuchen und eine bessere Bedingung zur Konvergenz angeben, halten aber schon mal fest:

Definition 7.1 *(**Potenzmethode, Vektoriteration nach von Mises**)*
Die Folge $a^{(j)}$ heißt Potenzmethode zur Bestimmung des betragsmaximalen Eigenwerts von A.

Korollar 7.2 *(**Konvergenz der Potenzmethode, einfache Version**)*
Die Potenzmethode konvergiert, falls

1. *Die Matrix A eine Basis aus Eigenvektoren besitzt.*
2. *A einen einzigen echt betragsmäßig größten Eigenwert besitzt, d.h.*

$$|\lambda_1| > |\lambda_k|,\ k = 2 \dots n.$$

3. $\alpha_1 \neq 0$.
4. $(\widetilde{x}, v_1) \neq 0$.

In diesem Fall gilt $\lim y^{(j)} = \frac{1}{(\widetilde{x}, v_1)} v_1$, $\lim a^{(j)} = \lambda_1$.

Bedingung 3 und 4 spielen numerisch keine Rolle (s. auch Beispiel in den Übungen), dies werden wir nicht weiter betrachten. Bedingung 1 und 2 dagegen sind problematisch. Wir werden bei dem echten Beweis der Potenzmethode zeigen, dass ein alternativer Beweis im Fall 1. immerhin noch (langsame) Konvergenz beweist, im Fall 2. versagt die Potenzmethode in der vorgelegten Form, falls unterschiedliche betragsmaximale Eigenwerte existieren.
Zusätzlich lösen wir hier natürlich nur ein Teilproblem: Etwa zur Berechnung der Singulärwertzerlegung benötigen wir alle Eigenwerte einer Matrix, das wird hier nicht geliefert. Zunächst schauen wir auf die Kondition des Eigenwertproblems.

7.1 Kondition des Eigenwertproblems

Zunächst betrachten wir den Spezialfall, dass A hermitesch ist, und geben ein Einschließungskriterium an.

Satz 7.3 *Es sei $A \in C^{n\times n}$ hermitesch. Weiter seien λ und x Näherungen für einen Eigenwert von A mit zugehörigem Eigenvektor, und es sei $d := Ax - \lambda x$ das Residuum. Dann gibt es einen Eigenwert von A mit*

$$|\lambda_i - \lambda| \leq \frac{||d||_2}{||x||_2}.$$

Beweis: A ist hermitesch, also hat der $\mathbb{C}^n$ eine Orthonormalbasis $v_1, \ldots, v_n$ mit zugehörigen Eigenwerten $\lambda_1, \ldots, \lambda_n$. Nach Satz 6.6 gilt

$$x = \sum_{k=1}^{n} c_k v_k,\ c_k = (x, v_k),\ ||x||_2^2 = \sum_{k=1}^{n} |c_k|^2$$

und

$$d = Ax - \lambda x = \sum_{k=1}^{n} c_k(\lambda_k - \lambda)v_k.$$

Sei λ_i der zu λ nächste Eigenwert, also

$$|\lambda_i - \lambda| \leq |\lambda_k - \lambda|,\ k = 1 \ldots n.$$

Damit ist

$$||d||_2^2 = \sum_{k=1}^{n} |c_k|^2 |\lambda_k - \lambda|^2 \geq \sum_{k=1}^{n} |c_k|^2 |\lambda_i - \lambda|^2 = ||x||_2^2 |\lambda_i - \lambda|^2.$$

□

Für die Störungstheorie ist interessant

Korollar 7.4 *Statt der hermiteschen Matrix A sei nur eine Näherung $\tilde{A} = A + S$ bekannt. x sei Eigenvektor von $\tilde{A}$ zum Eigenwert λ. Dann gibt es einen Eigenwert λ_i von A mit*

$$|\lambda_i - \lambda| \leq \frac{||Sx||_2}{||x||_2} \leq ||S||_2.$$

Beweis: Wir wenden den Satz auf A an und erhalten die Existenz eines Eigenwerts λ_i mit

$$|\lambda_i - \lambda| \leq \frac{||Ax - \lambda x||_2}{||x||_2} = \frac{||Ax - \tilde{A}x||_2}{||x||_2} = \frac{||Sx||_2}{||x||_2} \leq ||S||_2.$$

□

Damit ist der Fehler in den berechneten Werten nicht größer als die Norm der Störung.
Leider sind die Verhältnisse für nicht–diagonalisierbare Matrizen viel schlechter. Falls ν die Dimension des größten Jordankästchens von A ist, so liegen die Fehler in der Größenordnung $||B||^{1/\nu}$, den Beweis finden Sie wieder im Artikel Kato [1995]. Genauer gilt

Satz 7.5 *Sei $A \in \mathbb{C}^{n\times n}$ mit Eigenwerten $\lambda_1, \ldots, \lambda_n$, und $J = X^{-1}AX$ die Jordan–Normalform von A. Sei ν die Dimension des größten Jordankästchens von J. Sei $A_\epsilon = A + \epsilon F$, $\epsilon < 1$.Dann liegen sämtliche Eigenwerte von $\tilde{A}$ in der komplexen Ebene in der Vereinigung der Kreise*

$$K_l = \{z \in \mathbb{C} : |z - \lambda_l| \leq \epsilon^{1/\nu}(1 + k_\infty(X))||F||_\infty.$$

Dabei ist $k_\infty(X)$ die Kondition von X in der Unendlichnorm, also

$$k_\infty(X) = ||X||_\infty ||X^{-1}||_\infty.$$

Der Satz sagt also: Für kleine Störungen geht der Fehler bei der Berechnung der Eigenwerte nicht notwendig linear mit dem Fehler in der Matrixnorm gegen 0 (wie bei Hermiteschen Matrizen), sondern nur mit ν–ten Wurzel. Für diagonalisierbare Matrizen erhalten wir wieder einen linearen Zusammenhang.
Beweis: Mit der Formulierung des Satzes ist schon klar, dass der Beweis über die Gerschgorin–Kreise laufen muss, und nutzen diesen Beweis als Übung für den Umgang mit dem Satz von Gerschgorin 5.18. Wir betrachten (ohne Einschränkung) den Fall, dass J nur aus einem Jordankästchen besteht, also $\nu = n$, andernfalls teilt man die Betrachtung in kleine Teilmatrizen auf, was sie unübersichtlich, aber nicht spannender macht.
Es gilt

$$A + \epsilon F = X(J + \epsilon \underbrace{X^{-1}FX}_{=:G})X^{-1}.$$

Dann gilt

$$||G||_\infty \leq ||X||_\infty ||F||_\infty ||X^{-1}||_\infty = k_\infty(X)||F||_\infty.$$

Sei nun $\delta = \epsilon^{1/n}$. Wir würden gern den Satz von Gerschgorin anwenden, tun wir das aber auf J, so erhalten wir immer einen Radius von mindestens 1 für die Gerschgorinkreise – Gerschgorin liefert für nicht–diagonalisierbare Matrizen sehr

schlechte Abschätzungen, wenn man ihn direkt anwendet. Wir skalieren J daher so, wie wir es schon einmal getan haben.
Sei $D \in \mathbb{C}^{n\times n}$ die Diagonalmatrix mit δ^{k-1} auf der Hauptdiagonalen (siehe 5.13). Ebenfalls wie dort betrachten wir die Matrix $D^{-1}JD$.

$$D = \begin{pmatrix} 1 & & & \\ & \delta & & \\ & & \ddots & \\ & & & \delta^{n-1} \end{pmatrix}, \; D^{-1}JD = \begin{pmatrix} \lambda & \delta & & & \\ & \lambda & \delta & & \\ & & \ddots & \ddots & \\ & & & \lambda & \delta \\ & & & & \lambda \end{pmatrix}.$$

Sei $G = (g_{i,j})$, dann gilt entsprechend

$$|(D^{-1}GD)_{i,j}| = |g_{i,j}\delta^{j-i}| \leq |g_{i,j}\delta^{1-n}| = |g_{i,j}|\delta/\epsilon.$$

Wir wenden den Satz von Gerschgorin 5.18 auf die Matrix

$$G' = D^{-1}(J + \epsilon G)D = D^{-1}JD + \epsilon D^{-1}GD$$

an, diese hat dieselben Eigenwerte wie $J + \epsilon G$ und damit wie A_ϵ.
Auf der Hauptdiagonalen von G' stehen die Einträge $\lambda + \epsilon g_{i,i}$. Die Summe der Beträge auf der Nebendiagonale ist

$$r_i \leq \delta + \epsilon\delta/\epsilon \sum_{j\neq i} |g_{i,j}| = \delta \left(1 + \sum_{j\neq i} |g_{i,j}|\right).$$

5.18 garantiert nun, dass jeder Eigenwert μ von A_ϵ in einem dieser Kreise liegt, also gibt es ein i mit

$$|\mu - (\lambda + \epsilon g_{i,i})| \leq r_i.$$

Damit ist aber

$$\begin{aligned} |\mu - \lambda| &\leq \epsilon|g_{i,i}| + r_i \\ &\leq \delta \left(1 + \sum_{j=1}^{n} |g_{i,j}|\right) && (\epsilon < \delta \text{ wegen } \epsilon < 1) \\ &\leq \delta(1 + ||G||_\infty) && \text{(nach 2.30)} \\ &\leq \delta(1 + k_\infty(X)||F||_\infty) \end{aligned}$$

und das war die Behauptung. □

Dieser Beweis ist ein schönes Beispiel für die schon beim Satz von Gerschgorin gemachte Bemerkung, dass sich beim Übergang zu ähnlichen Matrizen die Gerschgorin–Abschätzungen erheblich verschärfen können.

Der Vollständigkeit halber erwähnen wir noch eine in der Numerik häufig genutzte Abschätzung für die Eigenwerte hermitescher Matrizen.

Definition 7.6 *(**Rayleigh–Quotient**)*
Sei $A \in \mathbb{C}^{n\times n}$ hermitesch. Dann heißt

$$R_A : \mathbb{C}^n \mapsto \mathbb{R},\ R_A(x) := \frac{(Ax, x)}{(x, x)}$$

Rayleigh–Quotient von A.

Satz 7.7 *(**Courantsches Minimum–Maximum–Prinzip**)*
Sei $A \in \mathbb{C}^{n\times n}$ hermitesch. Seien $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von A, hier ausnahmsweise ihrer Größe nach geordnet, also $\lambda_k \geq \lambda_{k+1}$. Dann gilt

$$\lambda_j = \min_{U\ n+1-j\text{–dimensionaler Unterraum von } \mathbb{C}^n} \max_{0\neq x\in U} R_A(x).$$

Beweis: Wir nutzen den Satz nicht, daher hier ohne Beweis. Sie finden ihn z.B. in Schaback and Wendland [2004], Satz 15.3. □

7.2 Potenzmethode

In der Vorbemerkung haben wir bereits die Potenzmethode hergeleitet und den Beweis für ihre Konvergenz in einem einfachen Fall geführt. Wir sind nun etwas genauer und betrachten zusätzlich nicht–diagonalisierbare Matrizen.

Satz 7.8 *(**Konvergenz der Potenzmethode: Allgemeine Version**)*
Sei $A \in \mathbb{C}^{n\times n}$, $x^{(0)} \in C^n$, $\widetilde{x} \in \mathbb{C}^n$. Wir definieren die Vektoriteration durch

$$x^{(j)} = A^j x^{(0)},\ a^{(j)} = \frac{(x^{(j+1)}, \widetilde{x})}{(x^{(j)}, \widetilde{x})}.$$

Es seien $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von A (gezählt mit ihrer Vielfachheit im charakteristischen Polynom). Die Eigenwerte seien so angeordnet, dass

$$|\lambda_1| = |\lambda_2| = \ldots = |\lambda_r| > |\lambda_{r+1}| \geq \ldots \geq |\lambda_n|.$$

Falls $\lambda_1 = \lambda_2 = \ldots = \lambda_r$, so konvergiert die Potenzmethode für fast alle Werte von $x^{(0)}$ und $\tilde{x}$ gegen λ_1. Falls es zusätzlich r linear unabhängige Eigenvektoren zum Eigenwert λ_1 gibt, so gilt

$$|a^{(j)} - \lambda_1| \leq C(\lambda_r/\lambda_{r+1})^j,$$

sonst gilt nur

$$|a^{(j)} - \lambda_1| \leq C/j$$

jeweils für ein $C > 0$.
Falls es ein k *gibt mit* $|\lambda_1| = |\lambda_k|$*, aber* $\lambda_1 \neq \lambda_k$*, so konvergiert die Potenzmethode im allgemeinen nicht.*

Die genaue Konvergenzbedingung legen wir im Beweis fest. Wir schauen zunächst auf einige Beispiele.

Beispiel 7.9 *Wir geben immer die Jordannormalform der Matrix an.*

1. $$J = \begin{pmatrix} 4 & & \\ & 1 & \\ & & 1 \end{pmatrix}.$$

   Die Matrix ist diagonalisierbar, es gibt also eine Basis aus Eigenvektoren, die Potenzmethode konvergiert schnell mit der Rate $(1/4)^j$.

2. $$J = \begin{pmatrix} 4 & & \\ & 4 & \\ & & 1 \end{pmatrix}.$$

   Die Matrix ist diagonalisierbar, es gibt also ein linear unabhängiges System aus Eigenvektoren, es gibt zwei gleiche betragsmaximale Eigenwerte, die Potenzmethode konvergiert wieder mit der Rate $(1/4)^j$.

3. $$J = \begin{pmatrix} 4 & 1 & \\ & 4 & \\ & & 1 \end{pmatrix}.$$

   Zum maximalen Eigenwert 4 *gibt es nur einen Eigenvektor, deshalb ist die Konvergenz langsam wie* $1/j$.

4. $$J = \begin{pmatrix} 4 & & \\ & 4 & \\ & & -4 \end{pmatrix}.$$

   Die Matrix hat zwei unterschiedliche betragsmaximale Eigenwerte, die Potenzmethode konvergiert im allgemeinen nicht.

Hier nun einige typische Beispielverläufe im Diagramm. Zunächst bei eindeutigem betragsmaximalem Eigenwert und Diagonalisierbarkeit (7.1):

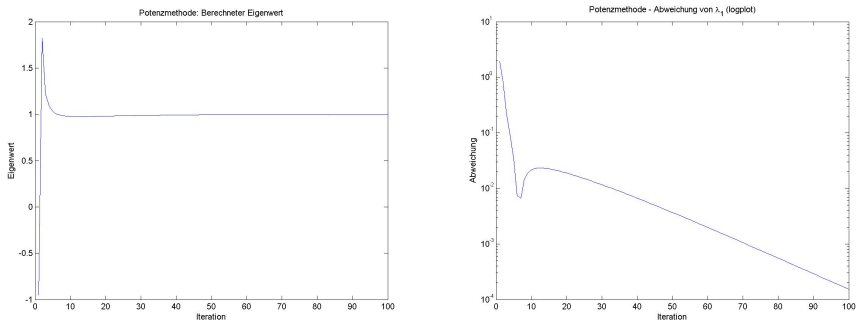

Abbildung 7.1: Schnelle Konvergenz der Potenzmethode. Links: Folge $a^{(j)}$, rechts: $\log|\lambda_1 - a^{(j)}||$

Klick für Bild PotenzFast
Klick für Matlab Figure PotenzFast
Klick für Bild PotenzFastlogplot
Klick für Matlab Figure PotenzFastlogplot

Die Folge konvergiert. Die Konvergenz geht mit $(\lambda_2/\lambda_1)^j$. *Rechts plotten wir den Logarithmus der Abweichung* $|\lambda_1 - a^{(j)}|$. *Wegen* $\log(|\lambda_2/\lambda_1|^j) = j\log|\lambda_2/\lambda_1|$ *erwarten wir, dass dieser linear ist in den Iterationen, dies ist tatsächlich der Fall.*

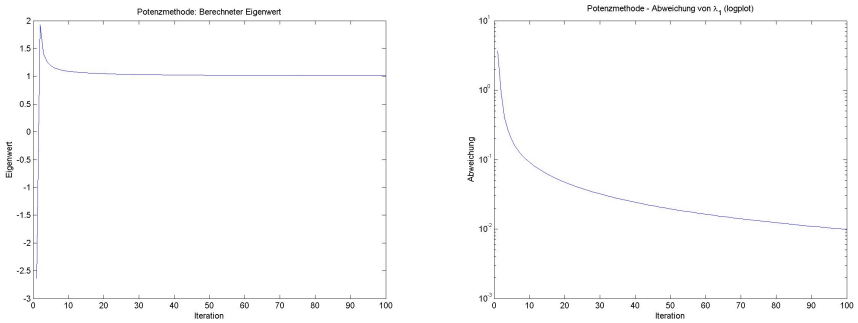

Abbildung 7.2: Langsame Konvergenz der Potenzmethode.

Klick für Bild PotenzSlow
Klick für Matlab Figure PotenzSlow
Klick für Bild PotenzSlowlogplot
Klick für Matlab Figure PotenzSlowlogplot

Hier sind nicht ausreichend viele Eigenvektoren zum betragsmaximalen Eigenwert vorhanden, die Matrix ist nicht diagonalisierbar. Wir erwarten langsame Konvergenz, dies ist der Fall, der Logarithmus ist sub–linear (7.2).

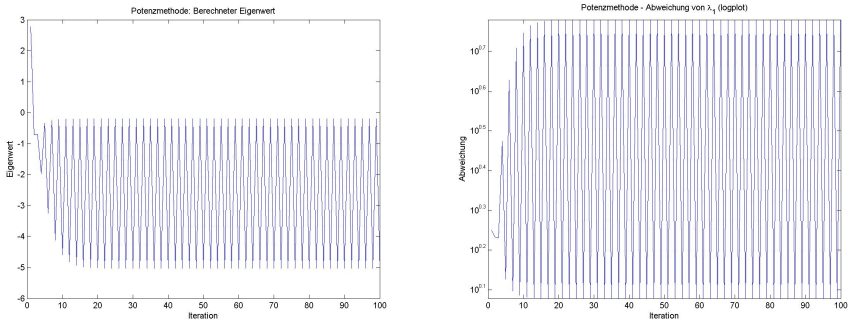

Abbildung 7.3: Divergenz der Potenzmethode.

Klick für Bild PotenzNoConv
Klick für Matlab Figure PotenzNoConv
Klick für Bild PotenzNoConvlogplot
Klick für Matlab Figure PotenzNoConvlogplot

Hier haben wir zwei unterschiedliche betragsmaximale Eigenwerte, wir erwarten, dass die Folge divergiert (und zwar hin– und herspringt) (7.3). Dies ist der Fall.

```
function [ vl,pos ] = Potenz( A,x,N,d,filename )
%POTENZ Potenzmethode
n=size(A,1);
if (nargin<3)
    N=x;
    x=rand(n,1);
```

Listing 7.1: Potenzmethode (Eigenwerte/Potenz.m)

Klicken für den Quellcode von Eigenwerte/Potenz.m

```
function [ output_args ] = demopotenz( q )
%DEMOPOTENZ
n=10;
format compact;
D=diag(sort(rand(n,1),1,'descend'));
%D(2,2)=-D(1,1);
```

Listing 7.2: Treiber für Potenzmethode (Eigenwerte/demopotenz.m)

Klicken für den Quellcode von Eigenwerte/demopotenz.m

Beweis: Falls wir r linear unabhängige Eigenvektoren zum Eigenwert λ_1 haben, so kann der Beweis wie in der Einleitung geführt werden, die Konvergenzgeschwindigkeit der $w^{(j)}$ und damit der $a^{(j)}$ ist $O(\lambda_{r+1}/\lambda_1)^j$, die genaue Konvergenzbedingung haben wir bereits angegeben.
Ebenso: Falls unterschiedliche Eigenwerte mit gleichem Maximalbetrag existieren, so konvergieren die Vektoren $w^{(j)}$ aus der Einleitung nicht, und wir erhalten auch keine Konvergenz.
Der einzige interessante Fall: Was passiert bei nicht–diagonalisierbaren Matrizen? Um Bezeichnungswirrwarr zu vermeiden, beschränken wir uns auf die Betrachtung des folgenden Falls (alle anderen sind mit derselben Idee zu führen): A ist nicht diagonaliserbar, aber ihre Jordannormalform $J = X^{-1}AX$ hat nur eine 1 an der Stelle $(1,2)$, und es sei $|\lambda_1| > |\lambda_3|$, also

$$J = \begin{pmatrix} \lambda_1 & 1 & & & \\ & \lambda_1 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ & & & & \lambda_n \end{pmatrix}.$$

Seien v_k die Spalten von X. Wegen

$$(v_1, \ldots, v_n) \begin{pmatrix} \lambda_1 & 1 & & & \\ & \lambda_1 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ & & & & \lambda_n \end{pmatrix} = XJ = AX = (Av_1, \ldots, Av_n)$$

gilt

$$Av_k = \lambda_k v_k \, (k \neq 2); \; Av_2 = v_1 + \lambda_1 v_2.$$

Also gilt insbesondere

$$A^j v_k = \lambda_k^j v_k (k \neq 2); \; A^j v_2 = \lambda_1^j v_2 + j\lambda_1^{j-1} v_1.$$

Sei nun wieder

$$x^{(0)} = \sum_{k=1}^{n} \alpha_k v_k$$

und wie in 7.1

$$\begin{aligned} x^{(j)} &= A^j \sum_{k=1}^{n} \alpha_k v_k \\ &= \lambda_1^j \alpha_1 v_1 + \lambda_1^j \alpha_2 v_2 + j\lambda_1^{j-1} \alpha_2 v_1 + \sum_{k=3}^{n} \alpha_k \lambda_k^j v_k \\ &= j\lambda_1^j \underbrace{\left((\alpha_1/j + \alpha_2/\lambda_1) v_1 + (\alpha_2/j) v_2 + \sum_{k=3}^{n} (\alpha_k/j) \left(\frac{\lambda_k}{\lambda_1} \right)^j v_k \right)}_{=:w^{(j)}}. \end{aligned}$$

Also konvergiert $w^{(j)}$ gegen $(\alpha_2/\lambda_1)v_1$, aber nur sehr langsam (nämlich wie $1/j$) und unter der Voraussetzung, dass $\alpha_2 \neq 0$.
Für die $a^{(j)}$ gilt

$$a^{(j)} = \frac{(x^{(j+1)}, \widetilde{x})}{(x^{(j)}, \widetilde{x})} = \frac{j+1}{j} \lambda_1 \frac{(w^{(j+1)}, \widetilde{x})}{(w^{(j)}, \widetilde{x})} \to_{j \mapsto \infty} \lambda_1$$

und natürlich auch nur mit langsamer Konvergenz wie $1/j$ und unter der Voraussetzung, dass $(v_2, \widetilde{x}) \neq 0$. □

Es stellt sich die Frage, wie häufig es auftritt, dass eine Matrix tatsächlich zwei unterschiedliche Eigenwerte von gleichem Betrag hat. Tatsächlich ist dies oft der Fall und liefert die Erklärung für das folgende Phänomen:

Sei A eine reelle Matrix, der betragsmaximale Eigenwert λ_1 sei komplex (also nicht reell). Sei weiter $x^{(0)}$ ein reeller Startvektor für die Potenzmethode, und $\widetilde{x}$ ein reeller Referenzvektor. Dann kann die Potenzmethode nicht konvergieren, denn alle $a^{(j)}$ sind reell.
Wie passt das mit unserem Satz zusammen? Klarerweise ist für eine reelle Matrix mit λ_1 auch $\overline{\lambda}_1$ eine Nullstelle des (reellen) charakteristischen Polynoms, also Eigenwert. Da $|\lambda_1| = |\overline{\lambda}_1|$, aber $\lambda_1 \neq \overline{\lambda}_1$, konvergiert die Potenzmethode nicht.
Eine Methode, auch bei betragsgleichen Eigenwerten zu Konvergenz zu kommen, ist die Nutzung von **Shifts**. Im einfachsten Fall wird die Potenzmethode auf die Matrix $A - \sigma I$ angewandt, was alle Eigenwerte und damit die Beträge verschiebt.
Eine weitere Methode ist die **inverse Iteration nach Wielandt**, dort wird die Potenzmethode auf die Matrix A^{-1} angewandt, und bestimmt (bei Konvergenz) den betragskleinsten Eigenwert von A.
Im Folgenden werden wir Verfahren zur Bestimmung aller Eigenwerte betrachten. Eine mögliche Idee wäre: Wir bestimmen zunächst mit der Potenzmethode den betragsmaximalen Eigenwert λ_1.
Dann wählen wir x^0 und $\tilde{x}$ absichtlich so, dass die Konvergenzbedingung aus der Einleitung verletzt wird. Das Verfahren kann in diesem Fall nicht mehr gegen λ_1 konvergieren, im allgemeinen wird es gegen λ_2 konvergieren. Für Hermitesche Matrizen müssen wir dazu nur $x^{(0)}$ orthogonal zum Eigenvektor von λ_1 wählen. Schaut man sich den typischen Iterationsverlauf an, so scheint das auch zu funktionieren (im Beispiel 7.4 bis zum Iterationsschritt 40), aber plötzlich schlägt die Folge um und konvergiert doch gegen λ_1.
Dies ist leicht zu erklären: Da α_2 sehr klein (oder sogar Null) ist, bleibt der erste Term in der Summe in w_k zunächst klein. Durch Rundungsfehler schleichen sich aber kleine Fehler ein, die durch den Exponentialterm groß werden. Das ist auch der Grund, warum die Konvergenzbedingungen 3 und 4 praktisch keine Rolle spielen - man muss nur lang genug iterieren. Zur Bestimmung aller Eigenwerte können wir die Grundidee dieses Ansatzes trotzdem weiterverwenden.

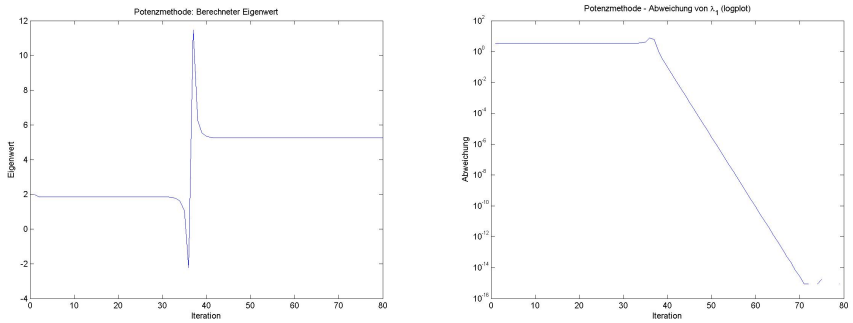

Abbildung 7.4: Konvergenz bei ungünstigem Anfangswert

Klick für Bild Semikonvergenz
Klick für Matlab Figure Semikonvergenz
Klick für Bild Semikonvergenzlogplot
Klick für Matlab Figure Semikonvergenzlogplot

```
function [ output_args ] = Semikonvergenz( input_args )
%SEMIKONVERGENZ
A=[5 1 0;1 1 1; 0 1 1];
[V D]=eig(A);
diag(D)'
D(2,2)/D(3,3)
```

Listing 7.3: Semikonvergenz für die Potenzmethode bei ungünstigen Anfangswerten (Eigenwerte/Semikonvergenz.m)

Klicken für den Quellcode von Eigenwerte/Semikonvergenz.m

7.3 Der QR–Algorithmus zur Bestimmung aller Eigenwerte einer Matrix

Die Potenzmethode und ihre Modifikationen haben den Nachteil, dass sie nur einen Eigenwert einer Matrix bestimmen. Manchmal benötigt man aber alle (etwa für die Singulärwertzerlegung).
Hierzu werden wir die Idee am Ende des letzten Abschnitts weiterentwickeln. Sei zunächst einmal die Matrix A hermitesch und positiv definit mit betragsmäßig verschiedenen Eigenwerten. Dann gibt es eine ONB aus Eigenvektoren. Mit der Potenzrechnung berechnen wir eine Näherung für λ_1 und einen zugehörigen Eigenvektor

v_1. Wählen wir nun $x^{(0)}$ senkrecht zu v_1, so ist in 7.1 $\alpha_1 = 0$, der dominierende Term ist damit $\alpha_2\lambda_2^j v_2$, und mit derselben Betrachtung wie oben konvergiert die Potenzmethode gegen λ_2.
Leider ist dies, wie schon gesehen, nicht praktikabel. α_2 ist nicht genau 0, und deshalb setzt sich der Term in v_1 am Ende doch durch. Dies lässt sich aber leicht beheben: In jedem einzelnen Schritt ziehen wir die Projektion auf v_1 von $x^{(j)}$ ab und garantieren dadurch, dass die Iteration im Orthogonalraum zu v_1 stattfindet, also

$$\widetilde{x}^{(j+1)} = Ax^{(j)}, x^{(j+1)} = \widetilde{x}^{(j+1)} - (x^{(j+1)}, v_1)v_1.$$

Tatsächlich erhalten wir auf diese Art eine stabile Konvergenz gegen λ_2. Wir führen diesen Gedanken nun noch fort und warten nicht, bis der erste Eigenvektor auskonvergiert ist, sondern ziehen die Iterationen gleichzeitig auf n Vektoren durch. Sei also

$$X^{(0)} = \left(x_1^{(0)}, \ldots, x_n^{(0)}\right).$$

Es sei $X^{(j)}$ bereits berechnet. Zunächst wenden wir die Matrix A auf alle Spalten an, berechnen also

$$\left(\widetilde{x}_1^{(j+1)}, \ldots, \widetilde{x}_n^{(j+1)}\right) = \widetilde{X}^{(j+1)} = AX^{(j)} = \left(Ax_1^{(j)}, Ax_n^{(j)}\right).$$

Gemäß unserer Idee führen wir für $x_1^{(j)}$ eine normale Potenzmethode durch. Um Konvergenz zu erreichen, normieren wir noch mit einem Faktor, wir wählen $r_{1,1}^{(j)} = ||\widetilde{x}^{(j+1)}||$, so dass

$$r_{1,1}^{(j)} x_1^{(j+1)} = \widetilde{x}_1^{(j+1)}$$

mit $||x_1^{(j+1)}|| = 1$. $x_2^{(j+1)}$ soll orthogonal werden zu $x_1^{(j+1)}$. Wir wählen also $r_{2,2}^{(j)}$ und $r_{2,1}^{(j)}$ so, dass

$$r_{2,2}^{(j+1)} x_2^{(j+1)} = \widetilde{x}_2^{(j+1)} - r_{1,2}^{(j)} x_1^{(j+1)}$$

und

$$(x_1^{(j+1)}, x_2^{(j+1)}) = 0,\ ||x_2^{(j+1)}|| = 1.$$

Es wird natürlich sofort klar, was wir hier tun: Wir führen einfach nur das Schmidtsche Orthonormalisierungsverfahren an den Spalten von $\widetilde{X}^{(j+1)}$ durch. Wie schon in 3.12 gesehen, entspricht dies einfach der Berechnung der QR–Zerlegung von $\widetilde{X}^{(j+1)}$, es ist

$$\widetilde{X}^{(j+1)} = X^{(j+1)} R^{(j)} \text{ mit } R^{(j)} = (r_{i,k}^{(j)})$$

(und $X^{(j+1)}$ ist dabei natürlich unitär, $R^{(j)}$ eine rechte obere Dreiecksmatrix).
Diese Idee wollen wir nun noch etwas leichter zugänglich und implementierbar machen. Wegen

$$X^{(j+1)} R^{(j)} = AX^{(j)} \text{ gilt } R^{(j)} = (X^{(j+1)})^{-1} AX^{(j)}.$$

Wir definieren das QR–Verfahren als die Folge von Matrizen

$$A^{(j)} = \left(X^{(j)}\right)^{-1} A X^{(j)} = \underbrace{\left(X^{(j)}\right)^{-1} X^{(j+1)}}_{=:Q^{(j)}} \underbrace{\left(X^{(j+1)}\right)^{-1} A \left(X^{(j)}\right)}_{=R^{(j)}}.$$

$A^{(j)}$ ist also ähnlich zu A und hat damit dieselben Eigenwerte. Nach unserer Herleitung erwarten wir, dass $X^{(j)}$ gegen die Eigenvektoren von A konvergiert, also sollte $A^{(j)}$ gegen die Jordan–Normalform von A konvergieren. $Q^{(j)}$ und $R^{(j)}$ sind die eindeutige QR–Zerlegung von $A^{(j)}$ mit positiver Hauptdiagonale von $R^{(j)}$ (siehe 3.15). $A^{(j+1)}$ lässt sich einfach aus $A^{(j)}$ berechnen. Es gilt

$$\begin{aligned} A^{(j+1)} &= \left(X^{(j+1)}\right)^{-1} A X^{(j+1)} \\ &= \left(X^{(j+1)}\right)^{-1} A X^{(j)} \left(X^{(j)}\right)^{-1} X^{(j+1)} \\ &= R^{(j)} Q^{(j)} \end{aligned}$$

und $R^{(j)}$ und $Q^{(j)}$ berechnen wir mit Householder. Da alle $A^{(j)}$ zu A ähnlich sind, haben alle dieselben Eigenwerte wie A.
Der so entstehende Algorithmus ist einer der grundlegenden Algorithmen der linearen Algebra, und wohl der mit der kuriosesten Geschichte. Er wurde 1961 publiziert von John Francis,Francis [1961] und Francis [1962]. Francis hat nie einen Universitätsabschluss erhalten und den Algorithmus als Student entwickelt, aber dann das Feld verlassen, ihm ist die Bedeutung erst 2007 klargeworden, als Gene Golub ihn in Kleinarbeit aufstöberte. Golub, einer der Väter von Matlab und selbsternannter Professor SVD, schreibt zum QR–Algorithmus (übrigens nicht in einem Nachruf, Francis lebt noch im Gegensatz zu Golub (Stand 2012)):
Along with the conjugate gradient method, it provided us with one of the basic tools of numerical analysis.
Eine Würdigung des Beitrags von Francis findet sich in Golub and Uhlig [2009].
Wir können den **QR–Algorithmus** so zusammenfassen (mit der Wahl $X^{(0)} = I$):

Setze $A^{(0)} := A$.
Für $j = 0 \ldots \infty$
    Berechne die QR–Zerlegung $Q^{(j)} R^{(j)}$ von $A^{(j)}$
        mit positiver Hauptdiagonale von $R^{(j)}$.
    Setze $A^{(j+1)} := R^{(j)} Q^{(j)}$.
    Falls A nah genug an einer Diagonalmatrix ist, brich das Verfahren ab.

Anschließend können wir Näherungen für die Eigenwerte auf der Hauptdiagonalen von $A^{(j+1)}$ ablesen.
Momentan hoffen wir natürlich nur, dass dieser Algorithmus konvergiert. Nach unserer Motivation vielleicht überraschend, bekommen wir eine Teilkonvergenz sogar für nicht–hermitesche Matrizen. Wir zeigen dies für einen Spezialfall.

Satz 7.10 *(**Konvergenz des QR–Algorithmus**)*
Sei $A \in \mathbb{C}^{n\times n}$. A habe betragsmäßig unterschiedliche Eigenwerte λ_k mit

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|.$$

Dann hat A n linear unabhängige Eigenvektoren, ist also diagonalisierbar, d.h.

$$D = X^{-1}AX$$

mit der Diagonalmatrix

$$D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}.$$

X^{-1} besitze eine LR–Zerlegung $X^{-1} = L' \cdot R'$. Weiter sei $X = Q \cdot R$ die QR–Zerlegung von X mit positiver Hauptdiagonale von R. Dann gilt für die Matrizen des QR–Algorithmus

$$A_{i,k}^{(j)} \mapsto \begin{cases} \lambda_i & i = k \\ 0 & i > k \\ ? & i < k \end{cases}$$

wobei das Fragezeichen dafür steht, dass in diesem Fall die Folge nicht notwendig überhaupt konvergiert. Die Folge der Matrizen konvergiert also auf der Hauptdiagonalen gegen die Eigenwerte und darunter gegen 0, oberhalb treffen wir keine Aussage.

Vorlesungsnotiz: 1.1.2013
Die Bedingungen des Satzes sind viel zu scharf, mögen hier aber ausreichen. Wir machen zunächst einige Vorbemerkungen.

Lemma 7.11 *Mit den Bezeichnungen des Satzes und des QR–Algorithmus gilt*

1. $$A^j = Q^{(0)} \cdot \ldots Q^{(j-1)} R^{(j-1)} \cdot \ldots \cdot R^{(0)}.$$

2. *Falls $A^{(j)}$ gegen eine Matrix A konvergiert, so konvergieren auch die QR–Zerlegungen mit positiver Hauptdiagonale von $A^{(j)}$ gegen die QR–Z erlegung von A.*

3. *Seien*
$$A = Q \cdot R = \widetilde{Q} \cdot \widetilde{R}$$
zwei QR–Zerlegungen von A. Dann gibt es eine Diagonalmatrix $\widetilde{D}$ mit $|\widetilde{D}_{i,i}| = 1$ und
$$\widetilde{Q}^t Q = \widetilde{R} R^{-1} = \widetilde{D}.$$

4.
$$\left(RDR^{-1}\right)_{k,k} = D_{k,k}.$$

5.
$$\left(D^j L' D^{-j}\right)_{i,k} = \lambda_i^j L_{i,k} \lambda_k^{-j} = \begin{cases} 0 & i < k \\ 1 & i = k \\ \mapsto 0 & i > k \end{cases}$$

und damit
$$D^j L' D^{-j} = I + F^{(j)},\ F^{(j)} \mapsto 0.$$

Beweis: (des Lemmas)

1.
$$Q^{(0)} R^{(0)} = A$$
$$Q^{(0)} Q^{(1)} R^{(1)} R^{(0)} = Q^{(0)} A^{(1)} R(0) = Q^{(0)} R^{(0)} Q^{(0)} R^{(0)} = A^2$$
usw.

2. Für die erste Spalte ist die Aussage klar, dann weiter per Induktion über die Spalten.

3. Das ist 3.15.

4. Durch Hinschreiben des Matrixprodukts.

5. Genau wie im Beweis zu 7.5 und wegen $|\lambda_k| > |\lambda_{k+1}|$.

□

Beweis: (des Satzes)
Der Beweis orientiert sich an der Originalpublikation Francis [1961].
Wir leiten eine zweite QR–Zerlegungen von A^j her und vergleichen sie mit dem Lemma. Es gilt

$$\begin{aligned} A^j &= X D^j X^{-1} \\ &= X D^j L' R' \\ &= X D^j L' D^{-j} D^j R' \\ &= X(I + F^{(j)}) D^j R' \\ &= QR(I + F^{(j)}) D^j R' \\ &= Q \underbrace{(I + RF^{(j)} R^{-1})}_{=:P^{(j)} S^{(j)} \text{ (QR–Zerlegung, HD von } S^{(j)} \geq 0)} RD^j R' \\ &= \underbrace{(QP^{(j)})}\ \underbrace{(S^{(j)} RD^j R')} \end{aligned}$$

und dies ist eine weitere QR–Zerlegung von A^j. Nach Teil 2 des Lemmas konvergieren $P^{(j)}$ und $S^{(j)}$ gegen die Einheitsmatrix I. Nach Teil 3 des Lemmas gibt es also eine Diagonalmatrix $D^{(j)}$ mit Elementen vom Betrag 1 auf der Hauptdiagonalen und

$$Q^{(0)} \cdot \ldots \cdot Q^{(j-1)} = QP^{(j)}D^{(j)}$$

$$R^{(j-1)} \cdot \ldots \cdot R^{(0)} = \left(D^{(j)}\right)^{-1} S^{(j)}RD^jR'$$

Es gilt

$$\begin{aligned} Q^{(j)} &= (Q^{(0)} \cdot \ldots \cdot Q^{(j-1)})^{-1}(Q^{(0)} \cdot \ldots \cdot Q^{(j)}) \\ &= \left(QP^{(j)}D^{(j)}\right)^{-1} \left(QP^{(j+1)}D^{(j+1)}\right) \end{aligned}$$

und

$$\begin{aligned} R^{(j)} &= (R^{(j)} \cdot \ldots \cdot R^{(0)}) \left(R^{(j-1)} \cdot \ldots \cdot R^{(0)}\right)^{-1} \\ &= \left(\left(D^{(j+1)}\right)^{-1} S^{(j+1)}RD^{j+1}R'\right) \left(\left(D^{(j)}\right)^{-1} S^{(j)}RD^jR'\right)^{(-1)}. \end{aligned}$$

Wegen $A^{(j)} = Q^{(j)}R^{(j)}$ gilt also insgesamt

$$A^{(j)} = \left(D^{(j)}\right)^{-1} \underbrace{\left(P^{(j)}\right)^{-1} P^{(j+1)}S^{(j+1)}RDR^{-1} \left(S^{(j)}\right)^{-1}}_{\mapsto RDR^{-1}} D^{(j)}$$

Da RDR^{-1} rechte obere Dreiecksmatrix ist mit der Hauptdiagonalen von D, konvergiert $A^{(j)}$ also unterhalb der Hauptdiagonalen gegen 0, auf der Hauptdiagonalen gegen die Diagonale von D, oberhalb der Hauptdiagonalen können wir keine Aussage treffen. □

Der Beweis ist nicht schwierig, aber leider extrem technisch. Die zentrale Idee des Beweises ist der Eindeutigkeitssatz der QR–Zerlegung 3.15, und Teil 5 des Lemmas. Dort kann man auch die Konvergenzgeschwindigkeiten ablesen, nicht sehr überraschend, entsprechen sie denen der Potenzmethode.
Tatsächlich kann man auf die meisten Voraussetzungen verzichten, sie sind rein technischer Natur, um den Beweis halbwegs übersichtlich zu halten. Da die Idee aber auf der Potenzmethode beruht, erwarten wir natürlich dieselben Schwierigkeiten wie dort.
Bemerkung: (Bemerkungen zum QR–Algorithmus zur Bestimmung der Eigenwerte einer Matrix)

1. Falls die Matrix nicht diagonalisierbar ist, konvergiert der QR–Algorithmus für die entsprechenden Eigenwerte nur wie $1/j$.

2. Falls betragsgleiche, unterschiedliche Eigenwerte existieren, so konvergiert der QR–Algorithmus für die entsprechenden Eigenwerte nicht. In diesem Fall bleiben auch unterhalb der Hauptdiagonalen einige Elemente stehen, wir erhalten Kästchen, die den Jordan–Kästchen entsprechen.

3. Auch die Voraussetzung, dass die Hauptdiagonalen der QR–Zerlegungen positiv sein müssen, ist rein technisch. Man kann mit beliebigen QR–Zerlegungen arbeiten.

Unsere Demo zeigt das typische Konvergenzverhalten. In unserem Beispiel konvergieren einige Eigenwerte schnell (sie sind betragsmäßig getrennt bzw. haben vollen Eigenraum–Rang) (Block oben links), einige langsam (dort gibt es nicht ausreichend viele Eigenvektoren) (unten rechts), einige gar nicht (dort gibt es betragsgleiche, verschiedene Eigenwerte) (in der Mitte).

Abbildung 7.5: QR-Algorithmus: Typisches Konvergenzverhalten

Klick für Bild qralgdemo
Klick für Matlab Figure qralgdemo

```
function doiteigmovie( input_args )
%DOITEIGMOVIE
N=8;
A=zeros(N);
A(1,1)=1;
A(2,2)=1;
```

Listing 7.4: Konvergenzverhalten des QR-Algorithmus (Eigenwerte/doiteigmovie.m)

Klicken für den Quellcode von Eigenwerte/doiteigmovie.m

```
function S = qralg( A,N,p )
%QRALG QR-Algorithmus
S=A;
for k=1:p
for i=1:N
    [Q,R]=qr(S);
```

Listing 7.5: QR–Verfahren (Eigenwerte/qralg.m)

Klicken für den Quellcode von Eigenwerte/qralg.m

```
function [ output_args ] = demoqralg( input_args )
%DEMOQRALG
A=rand(4);
A=A+A';
qralg(A,20,1);
A=rand(128);
```

Listing 7.6: QR–Verfahren: Treiber 1 (Eigenwerte/demoqralg.m)

Klicken für den Quellcode von Eigenwerte/demoqralg.m

```
function [ output_args ] = demoqr2( n,m,p,q, r )
%DEMOQR2 Summary of this function goes here
%   Detailed explanation goes here
format compact;
if (nargin<1)
    n=10;
```

Listing 7.7: QR–Verfahren: Treiber 2 (Eigenwerte/demoqr2.m)

Klicken für den Quellcode von Eigenwerte/demoqr2.m

Nach unserer Motivation hätte man vielleicht erwartet, dass der QR–Algorithmus nur für selbstadjungierte Matrizen konvergiert. Für allgemeine Matrizen ist die Motivation:
Für die erste Spalte von $X^{(j)}$ führen wir eine normale Potenzmethode durch. Für die zweite Spalte auch, aber wir ziehen immer ein Vielfaches der ersten Spalte so ab, dass die Spalten senkrecht aufeinander stehen. Dadurch verhindern wir die Konvergenz gegen v_1 und erreichen eine Konvergenz in einem Teilraum, der von v_1 und v_2 aufgespannt wird, und das reicht tatsächlich schon aus.

Also: Wir führen n Potenzmethoden gleichzeitig durch, verhindern aber durch eine kleine Modifikation, dass alle Folgen gegen denselben Vektor v_1 konvergieren.
Dies kann man auch leichter haben und war drei Jahre vorher die Idee von Heinz Rutishauser, veröffentlicht in den Mitteilungen der US–Amerikanischen Normierungsbehörde (Rutishauser [1958]), die auch die bekannteste Formel– und Tabellensammlung Abramowitz and Stegun [1965] veröffentlicht hat. Leider nicht online. Auch Francis zitiert Rutishausers LR–Algorithmus als Motivation für den QR–Algorithmus. Eine amüsante Geschichte der QR– und LR–Algorithmen findet sich in Gutknecht and Parlett [2011].
Wir nehmen an, dass $(v_1)_1 \neq 0$. Auf der ersten Spalte führen wir eine normale Potenzmethode durch. Diesmal skalieren wir so, dass in jedem Schritt $x_1^{(j)} = 1$, dies entspricht der Wahl $\tilde{x} = e_1$ in der Potenzmethode.
Wir ziehen nun in der zweiten Spalte einfach ein Vielfaches der ersten Spalte so ab, dass in der zweiten Spalte in der ersten Zeile eine 0 steht, normieren so, dass in der zweiten Zeile eine 1 steht ($\tilde{x} = e_2$), und verhindern damit die Konvergenz gegen ein Vielfaches von v_1. Für die weiteren Spalten geht man ebenso vor. Dann ist $X^{(k)}$ eine linke untere Dreiecksmatrix, $R^{(k)}$ wie im QR–Algorithmus, und damit $X^{(k)}R^{(k)}$ LR–Zerlegung von $AX^{(k)}$.
Mit den gleichen Überlegungen wie oben führt dies zum **LR–Algorithmus**, bei dem einfach statt der QR–Zerlegung die LR–Zerlegung durchgeführt wird.

Setze $A^{(0)} := A$.
Für $j = 0 \dots \infty$
    Berechne die LR–Zerlegung $L^{(j)}R^{(j)}$ von $A^{(j)}$.
    Setze $A^{(j+1)} := R^{(j)}L^{(j)}$.
    Falls A nah genug an einer Diagonalmatrix ist, brich das Verfahren ab.

und wieder liest man Näherungen für die Eigenwerte auf der Hauptdiagonalen ab. Der Algorithmus hat dieselben Konvergenzeigenschaften wie der QR–Algorithmus, da aber keine Spaltenpivotsuche durchgeführt werden kann, gilt er als notorisch instabil und kann unter Umständen gar nicht ausgeführt werden, wenn im Verlauf der Iteration eine Matrix auftaucht, die keine LR–Zerlegung besitzt. Eine kleine Demo zum LR– und QR–Algorithmus findet sich hier.
Wir müssen auch noch auflösen, warum der QR–Algorithmus für allgemeine Matrizen oberhalb der Hauptdiagonalen nicht konvergiert. Sei z.B. $\lambda_1 = -1$. Nehmen wir an, dass $x_1^{(0)}$ ein Eigenvektor zu λ_1 ist. Dann gilt

$$x_1^{(j)} = (-1)^j x_1^{(0)},$$

d.h. die $x_1^{(j)}$ oszillieren, und damit müssen wir in der ersten Reihe von $A^{(j)}$ mit Ausnahme des Diagonalelements auch mit einem oszillierenden Verhalten rechnen.

Für die LR–Zerlegung gilt dieser Einwand nicht, dort wird eine echte Potenzmethode mit der vorgesehenen Normierung durchgeführt, und damit erwarten wir dort Konvergenz (nicht notwendig gegen 0) auch oberhalb der Hauptdiagonalen, wenn die Voraussetzungen erfüllt sind.

Praktische Durchführung des QR–Algorithmus

Da die QR–Zerlegung recht aufwändig ist ($2/3n^3$ Rechenoperationen nach unserer Berechnung) führt man den Algorithmus nicht direkt auf der Matrix A durch. Im Allgemeinen wird zunächst mit 3.16 die **Hessenberg–Form** von A berechnet. Nach Definition ist diese ähnlich zu A, hat also dieselben Eigenwerte. In den Übungen haben wir bereits gezeigt, dass sie einfach mit einer Modifikation des Householder–Algorithmus in $2/3n^3$ Rechenoperationen berechenbar ist. Die QR–Zerlegung dieser Matrix ist dann in $O(n^2)$ berechenbar, so dass man insgesamt auf einen Aufwand von $O(n^2)$ für einen Schritt des QR–Algorithmus kommt.

Noch einfacher wird dies für hermitesche Matrizen. Dann ist auch die Hessenbergform hermitesch, also eine Tridiagonalmatrix! Die QR–Zerlegung von Tridiagonalmatrizen ist in $O(n)$ berechenbar, und so kommt man auf einen Aufwand von $O(n)$ für einen Schritt des QR–Verfahrens, d.h. die Eigenwerte von hermiteschen Matrizen, z.B. für die Singulärwertzerlegung, sind extrem schnell berechenbar.

Es bleibt das Problem, dass bei betragsgleichen, unterschiedlichen Eigenwerten keine Konvergenz vorliegt. Dies löst man wieder mit geeigneten Shiftstrategien.

Kapitel 8

Numerische Approximation in metrischen Räumen

Eine typische Aufgabe der Numerik ist die Approximation von Funktionen. Bei der Nutzung eines Taschenrechners etwa nutzen wir bedenkenlos die Funktion zur Berechnung des Sinus einer Zahl - wohl wissend, dass dieser natürlich eigentlich nur die Grundrechenarten nativ beherrscht (und unter Umständen nicht mal diese, siehe unsere Diskussion zur Division mit dem Newtonverfahren in 5.33).
Einige Möglichkeiten, etwa die trigonometrischen Funktion $f(x) = \sin(x)$ näherungsweise zu berechnen, fallen uns im Lichte der Vorlesung gleich ein:

1. **Abgeschnittene Taylorentwicklung**: Es wird die Taylorentwicklung der Funktion bestimmt, aber bei der Auswertung nur bis zu einem endlichen Glied berechnet, also
$$f(x) \sim \sum_{k=0}^{n} (-1)^k \frac{1}{2k+1)!} x^{2k+1}.$$
Für $n \mapsto \infty$ liefert diese Formel den korrekten Wert.

2. **Lookup–Table**: Es werden sehr viele Werte der Funktion mit einem hochgenauen Algorithmus (etwa abgeschnittene Taylorentwicklung mit sehr großem n) ausgerechnet und vertafelt. Der gesuchte Wert wird dann durch Interpolation approximiert. Dies ist die Methode der großen Tafelwerke, siehe etwa Abramowitz and Stegun [1965]. Dies ist ein Spezialfall der bekannten Spline–Interpolation (s. Numerische Analysis).
Diese Methode ist sehr schnell ausführbar, sobald die Tafeln zur Verfügung stehen, benötigt aber einen langen Vorlauf und ist nicht mehr praktikabel, wenn hochgenaue Ergebnisse benötigt werden.

3. **Polynom–Interpolation**: Es werden nur relativ wenige Werte der Funktion hochgenau berechnet. Anschließend legt man ein Polynom durch die berechneten Werte und wertet dieses Polynom aus (das ist möglich nach 4.1). Dies hört sich attraktiv an, ist aber leider extrem instabil (und wird in der Vorlesung Höhere Analysis behandelt).

4. **Polynomiale Regression**: Es werden einige Werte der Funktion hochgenau berechnet. Anschließend sucht man ein Ausgleichspolynom niedrigen Grades. Dies funktioniert tatsächlich sehr gut. Für sehr viele berechnete Auswertungen entspricht dies der Gauss–Approximation (siehe unten).

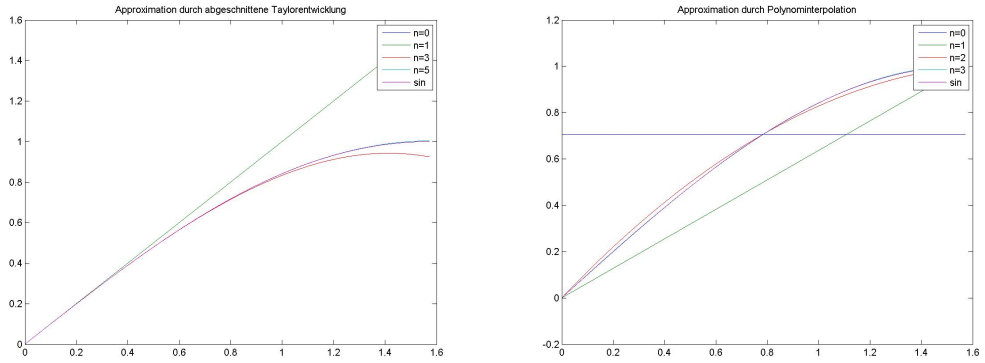

Abbildung 8.1: Approximation durch abgeschnittene Taylorentwicklung, Polynominterpolation

Klick für Bild abgtaylor
Klick für Matlab Figure abgtaylor
Klick für Bild polynominterp
Klick für Matlab Figure polynominterp

```
function abgtaylor
%ABGTAYLOR
%Approximation des Sinus durch abgeschnittene Taylorentwicklung

order=2;
N=10000;
```

Listing 8.1: Approximation des Sinus durch abgeschnittene Taylorentwicklung (Approximation/abgtaylor.m)

Klicken für den Quellcode von Approximation/abgtaylor.m

```
function [ output_args ] = polynom( input_args )
%POLYNOM
%   Approximation durch Polynominterpolation
order=2;
N=10000;
x=(0:N)/N*pi/2;
```

Listing 8.2: Approximation des Sinus durch Polynominterpolation (Approximation/-polynom.m)

Klicken für den Quellcode von Approximation/polynom.m

Im Bild sind abgeschnittene Taylorentwicklung und Polynominterpolation für den Sinus dargestellt. Es fällt auf, dass in beiden Fällen die lineare Approximation keineswegs optimal ist.
Wir vergleichen mit der polynomialen Regression. Hier sind die Ergebnisse für lineare Approximationen schon im Bild sehr viel besser.

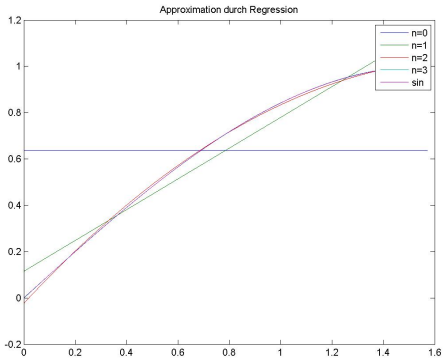

Abbildung 8.2: Approximation durch Regression

Klick für Bild regressioninterp
Klick für Matlab Figure regressioninterp

Zur Referenz geben wir hier die maximalen Fehler für eine Approximation mit Polynomen vom Grad ≤ 3 an:

	maximaler Fehler
abg. Taylor	0.075
Polynominterp.	0.0024
Regression	0.0027

Dabei haben wir den seltsamen Effekt, dass die Approximation durch Regression nach unserer Herleitung eigentlich besser sein sollte als die Polynominterpolation, die Kurve eigentlich auch sehr gut aussieht, der maximale Fehler aber trotzdem bei der Regression größer ist. Dies werden wir später noch untersuchen.
Kriterien für eine gute Approximation sind natürlich, dass der entstehende Approximationsfehler und der Aufwand zur Berechnung der Approximation möglichst klein sind. Keiner der vorgeschlagenen Algorithmen erfüllt dies. Tatsächlich werden in der Implementation in Hardware hochgenaue Algorithmen benutzt, die diesen im Aufwand um Größenordnungen überlegen sind (siehe Risse [2004] für einen guten Überblick). Wir werden hier nicht diese (CORDIC–) Algorithmen behandeln, sondern Approximationen, die eigentlich noch bessere Ergebnisse liefern, aber aufwändiger zu implementieren sind.
Wir werden in diesem Kapitel einige Grundbegriffe kennenlernen. Gute Referenzen für weitergehende Sätze und Beweise sind der klassische Text Meinardus [1964] und die neue Edition des Buchs Muller [2005]. Wieder betrachten wir ausschließlich reelle Vektorräume.

8.1 Bestapproximationen

Wir wollen nach Bestapproximationen suchen, also z.B. Polynome, die bezüglich einer vorgegebenen Norm eine Funktion f am besten unter allen Polynomen vom Grad $\leq n$ approximieren. Zunächst suchen wir nach allgemeinen Kriterien dafür, wann solche Bestapproximationen existieren und wann sie eindeutig sind.

Definition 8.1 *(***Minimalabstand** *und* **Bestapproximation***)*
Sei $(X, ||\cdot||)$ *ein normierter Raum. Sei* $T \subset X$ *nichtleer,* $f \in X$, $p^* \in T$. *Falls*

$$||p^* - f|| \leq ||\tilde{p} - f|| \, \forall \, \tilde{p} \in T$$

so heißt p^* *Bestapproximation an* f *in* T *bzgl.* $||\cdot||$.

$$d(f, T) = \inf\{||f - p|| : p \in T\}$$

heißt Minimalabstand von f *und* T.

Hierbei denken wir natürlich vor allem an Funktionenräume für X, also etwa den Raum aller stetigen Funktionen, versehen mit der Unendlich– oder euklidischen Norm. Wir betrachten trotzdem zunächst einige einfache Beispiele im $\mathbb{R}^2$.

Beispiel 8.2

1. *Sei* $X = (\mathbb{R}^2, ||\cdot||_2)$*,* $T = \{p \in X : ||p||_2 = 1\}$*. Sei* $f \in X$*.*
   Für $||f||_2 \leq 1$ *ist* f *Bestapproximation an* f *in* T*.*
   Für $||f||_2 \geq 1$ *ist* $f/||f||_2$ *eindeutige Bestapproximation an* f *in* T*.*

2. *Wie oben, aber diesmal sei* $T = \{p \in X : ||p||_2 < 1\}$*. In diesem Fall gibt es für* $||f|| \geq 1$ *keine Bestapproximation.*

3. *Jetzt wählen wir die Unendlichnorm, d.h.*
   $X = (\mathbb{R}^2, ||\cdot||_\infty)$*,* $T = \{p \in X : ||p||_\infty = 1\}$*,* $f = (3, 0)$*.*
   Natürlich ist $(1, 0)^t$ *eine Bestapproximation von* f*, aber wegen*

   $$||(3,0)^t - (1,1)^t||_\infty = 2 = ||(3,0)^t - (1,0)^t||_\infty$$

   auch $(1, 1)^t$*, d.h. in diesem Fall ist die Bestapproximation nicht eindeutig.*

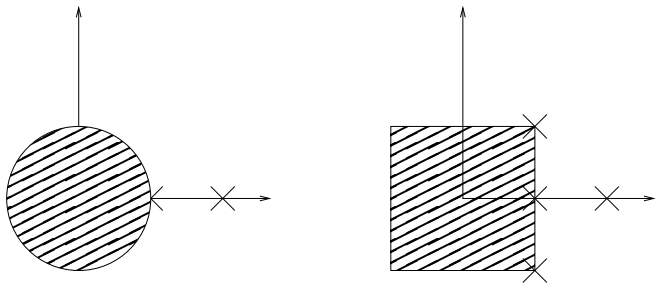

Abbildung 8.3: Approximation im $\mathbb{R}^2$. Links bzgl. der euklidischen Norm, rechts bzgl. der ∞–Norm.

Die Existenz und Eindeutigkeit der Bestapproximation hängt also von der Norm ab. Insbesondere geben andere Normen auch andere Bestapproximationen - die Wahl der richtigen Norm ist entscheidend für das Ergebnis. Tatsächlich liefert eine falsche Normwahl unsinnige Ergebnisse.
Im folgenden betrachten wir immer einen Raum X mit Norm $||\cdot||$. Wir vermuten natürlich, dass Kompaktheit bereits ausreicht, um die Existenz einer Bestapproximation zu garantieren.

Satz 8.3 *(***Existenz der Bestapproximation***)*

1. *Sei* T *kompakt,* $f \in X$*. Dann gibt es eine Bestapproximation an* f *in* T*.*

2. *Sei T endlich–dimensionaler affiner Unterraum von X, $f \in X$. Dann gibt es eine Bestapproximation an f in T.*

Beweis:

1. Die Funktion
$$D_f(x) = ||x - f||$$
ist stetig, T ist kompakt, also nimmt D_f auf T sein Minimum an.

2. Sei $g \in T$. Wir bezeichnen immer mit
$$K_r(f) = \{h \in X : ||h - f|| \leq r\}$$
die abgeschlossene Kugel in X um f mit Radius r.
$$K := T \cap K_{||f-g||}$$
ist kompakt als abgeschlossene und beschränkte Teilmenge eines affinen endlich–dimensionalen Unterraums von X. Also gibt es eine Bestapproximation p^* an f in K mit
$$||f - p|| \geq ||f - p^*)|| \, \forall p \in K.$$
Für $p \in T$, $p \notin K$ gilt aber dann wegen $g \in K$
$$||f - p|| > ||f - g|| \geq ||f - p^*||.$$

Damit ist p^* Bestapproximation an f in T. □

Teil 1 dieses Satzes klingt großartig – er ist auch einfach anzuwenden für endlich–dimensionale Räume, bei denen klar ist, was die kompakten Teilmengen sind (nämlich die abgeschlossenen und beschränkten). Für $X = C([a, b])$ lassen sich diese charakterisieren mit dem Satz von Arzela–Ascoli, hier kommt die Bedingung der gleichgradigen Stetigkeit hinzu.
Oben haben wir bereits gesehen, dass wir für die Eindeutigkeit zusätzliche Bedingungen an die Norm benötigen.

Definition 8.4 *(**strikt konvex, strikte Norm**)*

1. *Sei $K \subset X$. K heißt strikt konvex, falls für alle unterschiedlichen x,y aus K die Verbindungsstrecke zwischen x und y ganz im Inneren von K verläuft, also*
$$\lambda x + (1 - \lambda)y \in \mathring{K} \, \forall \lambda \in (0, 1) \, \forall x \neq y, \, x, y \in K.$$

2. $||\cdot||$ *heißt strikt, falls die Einheitskugel*

$$E_{||\cdot||} = \{p \in X : ||p|| \leq 1\}$$

strikt konvex ist.

Beispiel 8.5

1. $||\cdot||_2$ *im* $\mathbb{R}^2$ *ist strikte Norm, denn für zwei beliebige unterschiedliche Punkte auf dem Einheitskreis liegt die Verbindungsstrecke ganz im Inneren der Einheitskugel.*
2. $||\cdot||_\infty$ *im* $\mathbb{R}^2$ *ist keine strikte Norm. Die zugehörige Einheitskugel ist ein Quadrat mit Seitenlänge 2 um den Nullpunkt. Für* $(1,1)^t$ *und* $(1,-1)^t$ *verläuft die Verbindungsstrecke ganz auf dem Rand dieses Quadrats.*

Damit können wir zwei Eindeutigkeitssätze zeigen.

Satz 8.6 *(***Eindeutigkeit der Bestapproximation***)*

1. *Sei* T *strikt konvex,* $f \in X$*. Dann existiert höchstens eine Bestapproximation an* f *in* T*.*
2. *Sei* T *konvex, und* $||\cdot||$ *sei strikt konvex. Dann gibt es höchstens eine Bestapproximation an* f *in* T*.*

Beweis: Sei $B(f,T)$ die Menge aller Bestapproximationen nichtleer, und seien p_1 und p_2 zwei Bestapproximationen. Sei $f \notin T$. Wir zeigen jeweils $p_1 = p_2$.
Sei

$$K_r(f) = \{p \in X : ||p - f|| \leq r\}$$

die abgeschlossene Kugel um f mit Radius r und $d = d(f,T)$. Dann ist

$$B(f,T) = T \cap K_d(f)$$

als Schnitt konvexer Mengen konvex. Damit ist also

$$p^* = \frac{1}{2}(p_1 + p_2)$$

ebenfalls eine Bestapproximation.

1. Angenommen, $p_1 \neq p_2$. Da T strikt konvex ist, liegt p^* im Inneren von T. Es gibt also ein $1 > \epsilon > 0$, so dass

   $$q = p^* + \epsilon(f - p^*) \in T.$$

   Damit gilt

   $$||f - q|| = ||f - p^* - \epsilon(f - p^*)|| = (1 - \epsilon)\,||f - p^*|| < d$$

   und damit wären p_1 und p_2 keine Bestapproximationen. Also gilt $p_1 = p_2$.

2. Mit der Einheitskugel ist auch $K_d(f)$ strikt konvex. p_1 und p_2 liegen in $K_d(f)$. Angenommen, $p_1 \neq p_2$. Dann liegt p^* im Inneren von $K_d(f)$ und damit gilt

   $$||f - p^*|| < d$$

   im Widerspruch zur Minimalität. Also ist $p_1 = p_2$.

□

Bemerkung: Üblicherweise führt man strikte Normen über eine äquivalente Formulierung ein (siehe Übungen).

Korollar 8.7 *Sei $||\cdot||$ strikt und T affiner endlich–dimensionaler Teilraum von X. Dann ist die Bestapproximation an T eindeutig.*

Unglücklicherweise sind die interessanten Normen nicht strikt, mit Ausnahme der Normen in euklidischen Vektorräumen.

8.2 Gauss–Approximation

Als Gauss–Approximation bezeichnen wir die Bestapproximation in euklidischen Vektorräumen, bei denen also die Norm durch ein Skalarprodukt definiert ist. Dieses Problem haben wir bereits in 6.6 gelöst. Wir bemerken zunächst

Satz 8.8
Normen in euklidischen Vektorräumen sind strikt.

Beweis: Zu zeigen ist, dass die Verbindungsstrecke zweier unterschiedlicher Punkte x und y der Einheitskugel ganz im Inneren der Einheitskugel verläuft. Sei $||x|| = ||y|| = 1$ (sonst ist das sowieso klar). Sei $\lambda \in [0, 1]$. Es gilt

$$\begin{aligned}
||\lambda x + (1-\lambda)y||^2 &= \lambda^2||x||^2 + 2\lambda(1-\lambda)(x,y) + (1-\lambda)^2||y||^2 \\
&\leq \lambda^2||x||^2 + 2\lambda(1-\lambda)||x||\,||y|| + (1-\lambda)^2||y||^2 \quad \text{(Cauchy–Schwartz)} \\
&= (\lambda||x|| + (1-\lambda)||y||)^2 \\
&= 1.
\end{aligned}$$

Bei der Anwendung von Cauchy–Schwartz steht genau dann ein $=$, wenn $x = \mu y$ mit positivem μ. Da aber $||x|| = ||y|| = 1$ und $x \neq y$, ist diese Bedingung nicht erfüllbar. Also steht dort ein echtes $<$–Zeichen. □

Das Approximationsproblem in euklidischen Vektorräumen ist also immer eindeutig lösbar. Dies haben wir bereits in 6.6 bemerkt und auch die Lösung für endlich–dimensionale Unterräume angegeben. Wir fassen das Ergebnis noch einmal zusammen.

Korollar 8.9 *(**Bestapproximation in euklidischen Vektorräumen, Gauss–Approximation**)*
Sei $(X, (\cdot, \cdot))$ ein euklidischer Vektorraum mit induzierter Norm $|| \cdot ||$. Sei T ein endlich–dimensionaler Untervektorraum von X mit Orthogonalbasis $\{p_1, \ldots, p_n\}$. Sei weiter $f \in X$.
Dann ist die Bestapproximation p^* an f in T gegeben durch

$$p^* = \sum_{k=1}^{n} \frac{(f, p_k)}{(p_k, p_k)} p_k.$$

Wir wollen dieses Korollar nun für unser Ausgangsproblem der polynomialen Approximation nutzen. Sei dazu X der Vektorraum der stetigen Funktionen auf einem endlichen Intervall, versehen mit einem Skalarprodukt $(\cdot, \cdot)$, und $\mathcal{P}_n$ der Untervektorraum der Polynome vom Grad $\leq n$. Wir suchen das Polynom, das eine vorgegebene stetige Funktion f auf dem Intervall bzgl. der induzierten Norm am besten approximiert.
Um den Satz anwenden zu können, benötigen wir zunächst eine orthonormale Basis von $\mathcal{P}_n$.

Definition 8.10 *(**orthogonale und orthonormale Polynome**)*
Durch Anwendung des Gram–Schmidtschen Orthogonalisierungsverfahrens auf die Monome $1, x, x^2, \ldots$ erhält man eine Folge von paarweise orthonormalen Polynomen p_n, $n \geq 0$. Es gilt grad $p_n = n$. Die p_n heißen orthonormale Polynome des Skalarprodukts $(\cdot, \cdot)$. $\{p_0 \ldots, p_n\}$ ist Basis des Polynomraums $\mathcal{P}_n$. p_n steht senkrecht auf $\mathcal{P}_{n-1}$ für $n > 0$.
Verzichtet man auf die Normierung und ersetzt sie durch eine andere Forderung, so heißen die Polynome orthogonale Polynome.

In den Übungen haben Sie bereits nachgewiesen, dass die **Tschebyscheff–Polynome** orthogonale Polynome zum Skalarprodukt

$$(f, g) = \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} f(x) g(x) \, \mathrm{dx}$$

sind.
Als Übung berechnen wir die ersten orthogonalen Polynome zum Standardskalarprodukt für die stetigen Funktionen auf $[-1,1]$

$$(f,g) = \int_{-1}^{1} f(x)g(x)\,\mathrm{dx}.$$

Sie heißen **Legendre–Polynome**. Wir fordern die Normierung $p_n(1) = 1$.

$$p_0(x) = 1,\ ||p_0(x)|| = \sqrt{2}.$$
$$p_1(x) = x - \frac{1}{2}(x,p_0)p_0 = x.$$
$$\tilde{p}_2(x) = x^2 - \frac{1}{2}(x,p_0)p_0 = x^2 - \frac{1}{3}.\ p_2(x) = \frac{1}{2}(3x^2-1)\ \text{(Normierung)}$$

Orthogonale Polynome spielen in der angewandten Mathematik eine große Rolle, insbesondere in der Approximation und bei der Lösung gewöhnlicher Differentialgleichungen. Allgemein betrachtet man das Skalarprodukt

$$(f,g)_w = \int_a^b w(x)f(x)g(x)\mathrm{dx}\,.$$

mit positiven Funktionen w. Einige Beispiele mit zugehörigen orthogonalen Polynomen:

$[a,b]$	$w(x)$	Bezeichung
$[-1,1]$	1	Legendre–Polynome P_k
$[-1,1]$	$(1-x^2)^{-1/2}$	Tschebyscheff–Polynome 1. Art T_k
$[-1,1]$	$(1-x^2)^{1/2}$	Tschebyscheff–Polynome 2. Art U_k
$[-1,1]$	$(1-x)^\alpha(1+x)^\beta$	Jakobi–Polynome$P_k^{(\alpha,\beta)}$
$(-\infty,\infty)$	$e^{-x^2/2}$	Hermitesche Polynome H_k
$(0,\infty)$	e^{-x}	Laguerresche Polynome L_k

Die Eigenschaften dieser Polynome sind gut untersucht, wir bemerken

Satz 8.11 *Sei (p_n) System von orthogonalen Polynomen zum Skalarprodukt $(f,g)_w$ im Vektorraum der stetigen Funktionen auf $[a,b]$. Dann hat $p_n \in \mathcal{P}_n$ genau n Nullstellen in (a,b).*

Beweis: Seien $x_1,\ldots,x_m$ die Vorzeichenwechsel von p_n, also $m \le n$. Sei

$$q(x) := \prod_{k=1}^{m}(x - x_k) \in \mathcal{P}_m.$$

Dann hat $q(x)p_n(x)$ konstantes Vorzeichen. Angenommen, $m < n$. Da p_n dann senkrecht steht auf den Polynomen aus $\mathcal{P}_m$, gilt

$$0 = (p_n, q) = \int_a^b w(x)p_n(x)q(x)\mathsf{dx}\,.$$

Wegen $w > 0$ ist also $p_n \cdot q = 0 \lightning$. □

Als Beispiel für die Approximation berechnen wir nun noch die Bestapproximation des Cosinus auf $[-1, 1]$ durch Polynome vom Grad ≤ 2 bezüglich des Standard–Skalarprodukts. Wir setzen

$$p^* = \sum_{k=0}^{2} \alpha_k p_k,\ \alpha_k = \frac{(cos, p_k)}{(p_k, p_k)}.$$

Dann gilt

$$\begin{aligned}
\alpha_0 &= \frac{\int_{-1}^{1} \cos(x)1\mathsf{dx}}{\int_{-1}^{1} 1\mathsf{dx}} = \sin(1) \sim 0.84.\\
\alpha_1 &= \frac{\int_{-1}^{1} \cos(x)x\mathsf{dx}}{\int_{-1}^{1} x^2\mathsf{dx}} = 0.\\
\alpha_2 &= \frac{\int_{-1}^{1} \cos(x)\frac{1}{2}(3x^2-1)\mathsf{dx}}{\int_{-1}^{1} \frac{1}{4}(3x^2-1)^2\mathsf{dx}} = \frac{-4\sin(1) + 6\cos(1)}{2/5} \sim -0.31
\end{aligned}$$

Insgesamt erhalten wir die Approximation

$$\cos(x) \sim 0.99656 - 0.46526x^2,$$

die sich leicht von der abgeschnittenen Taylor–Reihe unterscheidet.

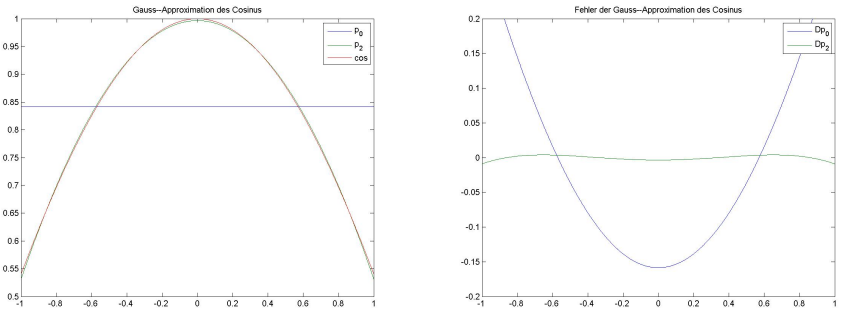

Abbildung 8.4: Gauss–Approximation des Cosinus

Klick für Bild gausscosdemo
Klick für Matlab Figure gausscosdemo
Klick für Bild gausscosdemodiff
Klick für Matlab Figure gausscosdemodiff

Für unser Standardbeispiel der Approximation des Sinus auf dem Intervall $I = [0, \pi/2]$ berechnen wir zunächst die orthogonalen Polynome p_k mit dem Gram–Schmidt–Verfahren auf I exakt in Maple bezüglich eines vorgegebenen Skalarprodukts und approximieren dann

$$p^* = \sum_{k=0}^{3} \frac{(f, p_k)}{(p_k, p_k)} p_k.$$

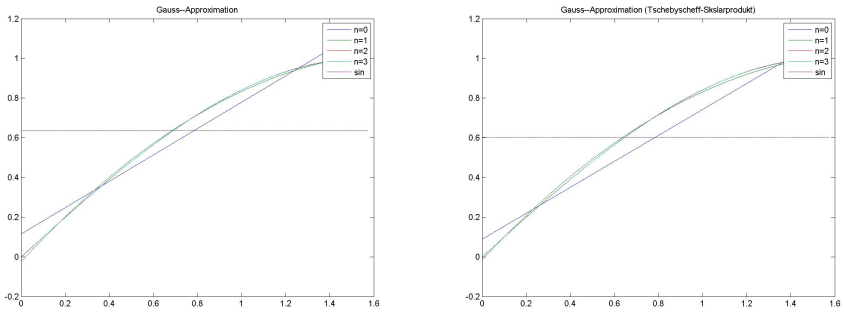

Abbildung 8.5: Gauss–Approximation. Links Legendre, rechts Tschebyscheff.

Klick für Bild gaussappro
Klick für Matlab Figure gaussappro
Klick für Bild gaussaprotscheb
Klick für Matlab Figure gaussaprotscheb

```
function [ output_args ] = gaussappro( input_args )
%GAUSSAPPRO
%Computes orthogonal polynomials with respect to a given scalar product
%and the best approximation in the Gauss sense to sin(x) on (0,pi/2)
%Attention: Maple toolbox may be required!
syms x;
```

Listing 8.3: Gauss–Approximation und orthogonale Polynome (Approximation/gaussappro.m)

Klicken für den Quellcode von Approximation/gaussappro.m

	maximaler Fehler
Gauss (Legendre)	0.0027
Gauss (Tschebyscheff)	0.0015

Wir erhalten für das Standard–Skalarprodukt also denselben Fehler wie für die Regression, tatsächlich liefern die beiden Verfahren ein identisches Ergebnis. Durch Verwendung des Skalarprodukts, das die Tschebyscheff-Polynome definiert, können wir den maximalen Fehler noch einmal um den Faktor 2 drücken.
Die berechneten Polynome sind

$$p_L^*(x) = -0.00225845161 + 1.027169448x - 0.0699435763x^2 - 0.1138685452x^3$$

$$p_T^*(x) = -0.001244756730 + 1.023967362x - 0.06858728910x^2 - 0.1133771946x^3$$

Die Erklärung dafür, dass die Tschebyscheff–Polynome ein besseres Ergebnis liefern: Die Approximation wird häufig zum Rand hin schlechter (das kann man analytisch zeigen), durch die Gewichtung sorgen wir dafür, dass der Rand besser approximiert wird.
Dies sieht zunächst sehr gut aus - hat aber einen Pferdefuß: Alle über Skalarprodukte definierten Normen tun eigentlich nicht das, was wir wollen. Als Beispiel betrachten wir die folgende (unstetige) Funktion:

$$f(x) = \begin{cases} 1 & ||x|| \leq 0.1 \\ 0 & \text{sonst} \end{cases}$$

auf $[-1, 1]$.
Angenommen, wir wollen diese Funktion durch eine Konstante approximieren. Dann gibt es verschiedene Strategien: Falls wir den durchschnittlichen Fehler minimieren wollen (dies tut die Gauss–Approximation), so sollten wir die Konstante nah an 0 wählen. Dadurch erhalten wir zwar für $x = 0$ einen großen Fehler, aber nur auf einem kleinen Gebiet. Tatsächlich liefert die Gauss–Approximation dieses Ergebnis. Wollen wir dagegen den maximalen Fehler minimieren (wie es etwa ein CPU–Architekt tun würde), würden wir natürlich $1/2$ als Konstante wählen. Dies ist die Idee der Tschebyscheff–Approximation.

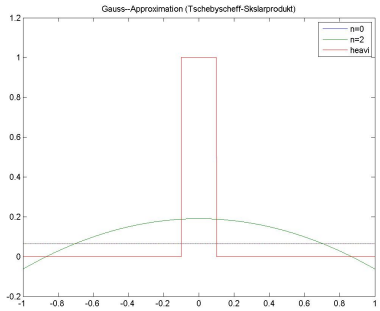

Abbildung 8.6: Approximation einer Treppenfunktion mit Gauss

Klick für Bild gaussapprotschebheavi
Klick für Matlab Figure gaussapprotschebheavi

Bemerkung: Wir haben hier der Einfachheit halber durch Polynomräume approximiert. Dies wurde tatsächlich in den Algorithmen und Sätzen gar nicht verwendet.

Gauss–Approximation lässt sich auf beliebigen Unterräumen durchführen. Eine beliebte Wahl ist die rationale Approximation, dort wählt man Unterräume, die auch Quotienten von Polynomen enthalten.

8.3 Tschebyscheff–Approximation

In diesem Abschnitt wählen wir als Norm im Vektorraum X der stetigen Funktionen auf einem abgeschlossenen Intervall $I = [a, b]$ die Unendlich–Norm $|| \cdot ||_\infty$. Wir haben bereits gesehen, dass die Unendlich–Norm nicht strikt ist, wir können also nicht erwarten, eindeutige Bestapproximationen zu erhalten. Bei der Approximation bezüglich der Unendlich–Norm versuchen wir, p^* so zu wählen, dass der Maximalabstand zwischen f und p minimal wird, also

$$||f - p^*||_\infty \leq ||f - p||_\infty \, \forall p \in \mathcal{P}_n.$$

p^* heißt Tschebyscheff–Approximation an f in $\mathcal{P}_n$. Wir schauen uns diese Bestapproximation zunächst am Beispiel des letzten Kapitels an. Wir wählen wieder

$$f(x) = \begin{cases} 1 & ||x|| \leq 0.1 \\ 0 & \text{sonst} \end{cases}$$

auf $[-1, 1]$ und $n = 0$. Natürlich ist die Bestapproximation dann die konstante Funktion

$$p^*(x) = 1/2$$

mit dem Maximalabstand $1/2$.

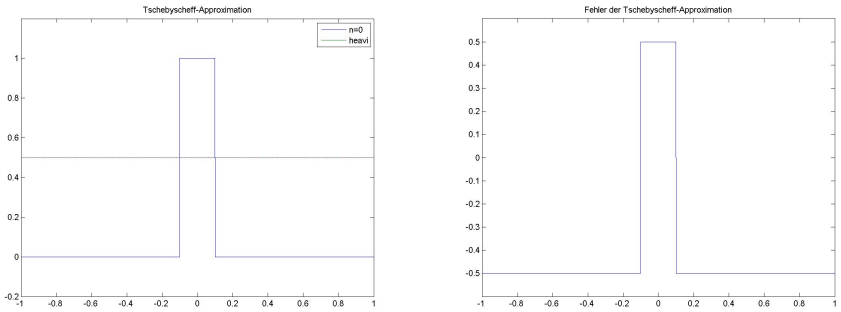

Abbildung 8.7: Tschebyscheff–Approximation vom Grad 0

Klick für Bild tscheb1
Klick für Matlab Figure tscheb1
Klick für Bild tscheb1b
Klick für Matlab Figure tscheb1b

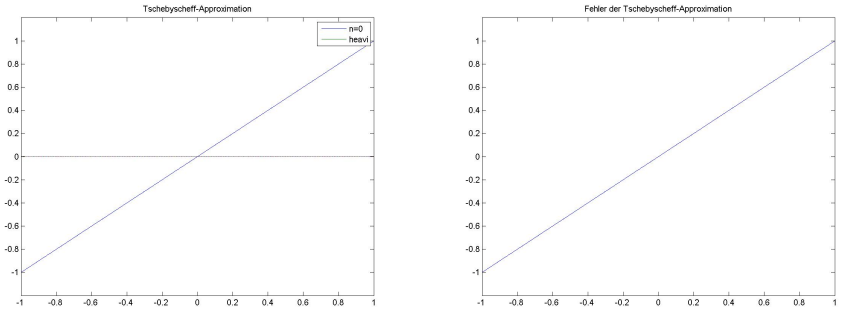

Abbildung 8.8: Tschebyscheff–Approximation vom Grad 0 ($f(x) = x$)

Klick für Bild tscheb1c
Klick für Matlab Figure tscheb1c
Klick für Bild tscheb1d
Klick für Matlab Figure tscheb1d

```
function [ output_args ] = tscheb1( input_args )
%TSCHEB1

N=1000;
x=(0:N)/N*2-1;
```

```
y=double(heaviside(0.1-abs(x)));
```

Listing 8.4: Tschebascheff–Approximation einfaches Beispiel (Approximation/tscheb1.m)

Klicken für den Quellcode von Approximation/tscheb1.m

Wenn wir den Fehler der Tschebyscheff–Approximation betrachten, fällt auf, dass er sein Betragsmaximum mit wechselnden Vorzeichen annimmt. Dies ist natürlich für $n = 0$ immer der Fall, man überlegt sich schnell, dass die beste Approximation die Konstante $(\max + \min)/2$ ist, und an den Stellen, an denen f Maximum oder Minimum annimmt, ist der Abstand sein Betragsmaximum mit wechselnden Vorzeichen. Gleichzeitig charakterisiert das die Bestapproximation eindeutig.
Wir betrachten nun wieder den Sinus auf $[0, \pi/2]$ und versuchen, die Bestapproximation zu vom Grad 1 zu schätzen.

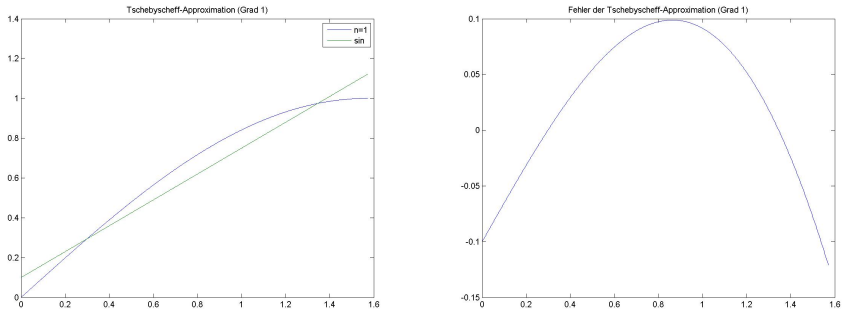

Abbildung 8.9: Tschebyscheff–Approximation vom Grad 1 (geraten)

Klick für Bild tscheb2
Klick für Matlab Figure tscheb2
Klick für Bild tscheb2b
Klick für Matlab Figure tscheb2b

```
function [ output_args ] = tscheb2( input_args )
%TSCHEB2

N=1000;
x=(0:N)/N*pi/2;
y=sin(x);
```

Listing 8.5: Tschebyscheff–Approximation vom Grad 1 (Approximation/tscheb2.m)

Klicken für den Quellcode von Approximation/tscheb2.m

Zumindest angenähert sehen wir auch hier, dass der Fehler der von uns geratenen Approximation sein Betragsmaximum mit unterschiedlichen Vorzeichen annimmt — nicht ganz genau, es ist ja nur geraten...
Wir vermuten daher, dass eine Funktion eine Bestapproximation bezüglich der Unendlich–Norm ist, wenn der Fehler sein Betragsmaximum mit wechselnden Vorzeichen (ausreichend oft) annimmt. Dies ist tatsächlich der Fall. Wir zeigen zunächst

Lemma 8.12
Sei $I = [a, b]$, $f : I \mapsto \mathbb{R}$ stetig. Sei $n \geq 0$ ganz, $p \in \mathcal{P}_n$.
Seien $x_k \in I$, $k = 0 \ldots n+1$, und $x_k < x_{k+1}$. Die Funktion

$$f - p$$

habe alternierende Vorzeichen in den x_k, d.h.

$$\operatorname{sgn}(f(x_k) - p(x_k)) = (-1)^k \sigma,\ \sigma \in \{-1, 1\}.$$

Dann gilt

$$\min_{k=0}^{n+1} |f(x_k) - p(x_k)| \leq d(f, \mathcal{P}_n).$$

Wir wenden das Lemma zunächst auf unsere geratene Approximation an. Der Fehler nimmt dort an drei Stellen die Werte -0.1, 0.1 und -0.12 an. Jedes Polynom vom Grad ≤ 1 hat also mindestens den Maximalabstand 0.1 zu f.
Beweis: Angenommen, es gibt ein $q \in \mathcal{P}_n$, das f besser approximiert als das Minimum. Dann gilt für alle $k = 0 \ldots n+1$

$$\begin{aligned}
\sigma(-1)^k(f(x_k) - p(x_k)) &= |f(x_k) - p(x_k)| \\
&> ||f - q||_\infty \\
&\geq |f(x_k) - q(x_k)| \\
&\geq \sigma(-1)^k(f(x_k) - q(x_k)).
\end{aligned}$$

Damit gilt

$$\sigma(-1)^k(q(x_k) - p(x_k)) > 0,\ j = 0 \ldots n+1.$$

Die Funktion $q - p$ hat also $n+1$ Vorzeichenwechsel und ist stetig. Sie hat also auch mindestens $n+1$ Nullstellen. Da p und q aber in $\mathcal{P}_n$ liegen, ist damit $q - p$ ein Polynom vom Grad $\leq n$ mit $n+1$ Nullstellen, also das Nullpolynom und damit $p = q$ im Widerspruch dazu, dass $q - p$ alternierendes Vorzeichen hat. □

Korollar 8.13 *(**Alternantensatz**)*
Sei $p^* \in \mathcal{P}_n$, $f : I \mapsto \mathbb{R}$ stetig. Die Funktion $f - p^*$ nehme für aufsteigend geordnete, unterschiedliche Argumente $x_k \in I$ ihr Betragsmaximum mit alternierendem Vorzeichen an, also

$$f(x_k) - p^*(x_k) = \sigma(-1)^k ||f - p^*||_\infty,\ \sigma \in \{-1, 1\}.$$

Dann ist p^* Bestapproximation an f in $\mathcal{P}_n$.

Beweis: Nach Lemma 8.12 gilt

$$||f - p^*|| \leq d(f, \mathcal{P}_n).$$

□

Die Folge von Argumenten x_k aus dem Lemma heißt **Alternante** zu $f - p^*$. Damit können wir also feststellen, ob ein Polynom p eine Bestapproximation ist. Es gilt auch die Umkehrung: Jede Tschebyscheff–Bestapproximation besitzt eine Alternante.

Satz 8.14 *(Existenz einer Alternante)*
Sei p^* die Bestapproximation an f in $\mathcal{P}_n$ bezüglich $|| \cdot ||_\infty$. Dann besitzt $f - p^*$ eine Alternante, d.h.

$$\exists x_0 < \ldots < x_{n+1},\ \sigma : (f - p)(x_k) = \sigma(-1)^k ||f - p^*||_\infty,\ |\sigma| = 1.$$

Beweis: Dies ist Satz 17 in Meinardus [1964].
Beweisskizze:
Sei $f \neq p^*$. Sei D die Menge aller Werte, für die $f - p^*$ sein Betragsmaximum annimmt, also

$$D := \{x \in I : |f(x) - p^*(x)| = ||f - p^*||_\infty\}.$$

Wir betrachten die Anzahl der Vorzeichenwechsel von $f - p^*$ auf D. Sind es mindestens $(n+1)$, so gibt es eine Alternante. Angenommen, dies sei nicht der Fall. Dann gibt es Werte $x_k \in I$ mit

$$(f - p^*)(x_k) = 0,\ k = 1 \ldots m,\ m \leq n,$$

die die Bereiche mit gleichem Vorzeichen trennen, denn D ist abgeschlossen.
Wir setzen

$$q(x) := \prod_{k=1}^{m} (x - x_k) \in \mathcal{P}_n.$$

Dann wechseln q und $(f - p^*)$ bezogen auf D an denselben Stellen ihr Vorzeichen. Ohne Einschränkung sei das Vorzeichen immer gleich. Wir betrachten

$$p(x) := p^*(x) + \epsilon q(x) \in \mathcal{P}_n$$

für ein kleines $\epsilon > 0$. Für $x \in D$ (und auch in einer kleinen Umgebung $\tilde{D}$) gilt dann

$$|f(x) - p(x)| = |(f(x) - p^*(x)) - \epsilon q(x)| < |f(x) - p^*(x)|.$$

Für ϵ hinreichend klein werden die Maxima von $f - p$ in $\tilde{D}$ angenommen, dies ist ein Widerspruch. □

Häufig sind die Intervallenden a und b Teil der Alternante. Mit dieser Annahme lässt sich manchmal die Alternante auch analytisch berechnen. Als Beispiel berechnen wir

Beispiel 8.15 *(Tschebyscheff–Approximation von x^2 durch lineare Funktionen)*
Sei $I = [0, 1]$, $f : I \mapsto \mathbb{R}$, $f(x) = x^2$. Sei $p^*(x) = \alpha x + \beta$ die Tschebyxscheff–Approximation an f. Sei x_0, x_1, x_2 die Alternante und $d = ||f - p^*||_\infty$. Wir machen den Ansatz $x_0 = 0$ und $x_2 = 1$. Dann gilt

$$\begin{aligned} p^*(0) - f(0) = \beta &= D \\ p^*(x_1) - f(x_1) = \alpha x_1 + \beta - x_1^2 &= -D \\ p^*(1) - f(1) = \alpha + \beta - 1 &= D \end{aligned} \quad .$$

für $D = d(f, \mathcal{P}_n)$ oder $D = -d(f, \mathcal{P}_n)$. Zusätzlich muss die Differenz am inneren Punkt x_1 ihr Maximum oder Minimum annehmen, die Ableitung muss also dort verschwinden, und damit

$$p'(x_1) - f'(x_1) = \alpha - 2x_1 = 0.$$

Wir erhalten sofort

$$\alpha = 1,\ x_1 = \frac{1}{2},\ \beta = D = -\frac{1}{8}$$

und damit ist

$$p^*(x) = x - \frac{1}{8}$$

die beste lineare Approximation an x^2. Die Alternante ist $(0, 1/2, 1)$, an allen diesen Stellen nimmt die Differenz ihren Maximalabstand $1/8$ mit alternierendem Vorzeichen an.

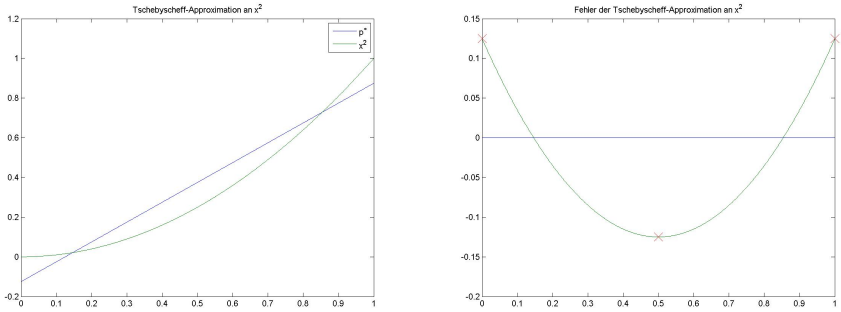

Abbildung 8.10: Tschebyscheff–Approximation an x^2

Klick für Bild remezbeispiel
Klick für Matlab Figure remezbeispiel
Klick für Bild remezbeispiel2
Klick für Matlab Figure remezbeispiel2

```
function remezbeispiel
%REMEZBEISPIEL
N=1000;
x=(0:N)/N;
alternante=[0 0.5 1];
plot(x,x-1/8,x,x.^2);
```

Listing 8.6: Beispiel zur Tschebyscheff–Approximation (Approximation/remezbeispiel.m)

Klicken für den Quellcode von Approximation/remezbeispiel.m

Als Anwendung beweisen wir

Satz 8.16 *(Optimalität der Tschebyscheff–Polynome)*
Sei $n > 0$ und T_n das Tschebyscheff–Polynom vom Grad n.

$$p(x) = \frac{1}{2^{n-1}} T_n(x)$$

hat unter allen Polynomen vom Grad n mit Höchstkoeffizient 1 die kleinste Unendlich–Norm auf dem Intervall $[-1, 1]$.

Beweis: In den Übungen wurde bereits gezeigt, dass p den Höchstkoeffizienten 1 hat. Also ist

$$\frac{1}{2^{n-1}} \cos(n \arccos x) = p(x) = x^n - v(x)$$

mit $v \in \mathcal{P}_{n-1}$. Wenn p minimale Norm haben soll unter allen Polynomen dieser Form, muss v die Bestapproximation von x^n in $\mathcal{P}_{n-1}$ sein.
Wir setzen

$$x_k = \cos(\frac{k\pi}{n}),\ k = 0 \dots n.$$

Dann gilt

$$\begin{aligned}(x_k)^n - v(x_k) &= \frac{1}{2^{n-1}} T_n(x_k) \\ &= \frac{1}{2^{n-1}} \cos(k\pi) \\ &= \frac{1}{2^{n-1}} (-1)^k.\end{aligned}$$

Also nimmt die Differenzfunktion mit wechselnden Vorzeichen an $(n+1)$ Stellen ihr Betragsmaximum an, ist also Alternante und v ist Bestapproximation. □

Wir wollen nun noch einen iterativen Algorithmus herleiten, der die Tschebyscheff–Approximation berechnet. Der Beweis zu Satz 8.14 gibt dazu eine Idee: Wir starten mit einem Polynom p_0. Falls $f - p_0$ eine Alternante hat, so sind wir fertig. Falls nicht, so können wir wie im Beweis eine neue Funktion p_1 konstruieren, die einen kleineren Maximalabstand zu f hat als p_0 usw.
Dieser Algorithmus ist leider wenig praktikabel. Die Untersuchung, ob $f - p_0$ eine Alternante besitzt bzw. die Berechnung der Menge D aus dem Beweis wird schnell aufwändig.
Statt dessen nutzen wir den Remez–Algorithmus. Er geht nicht vom Polynom aus und berechnet die zugehörige Alternante, sondern versucht, iterativ eine Alternante und dadurch die zugehörige Bestapproximation zu bestimmen.
Wenn $x_0, \dots, x_{n+1}$ eine Alternante zur Approximation von f durch $p \in \mathcal{P}_n$ bilden, so haben sie zu der vorgegebenen Funktion f an diesen Stellen den gleichen Abstand $D = ||f - p||_\infty$.
Wir starten also zunächst mit irgendeiner geordneten Startverteilung von Punkten $x_0, \dots, x_{n+1}$ aus dem Intervall I. Dann berechnen wir ein Polynom p, das zu f an diesen Punkten den gleichen Abstand D mit alternierendem Vorzeichen hat. Das folgende Lemma zeigt, dass dieses Polynom eindeutig bestimmt werden kann durch Lösen eines linearen Gleichungssystems in $(n+2)$ Variablen. Falls $|D| = ||f - p||_\infty$, so sind wir fertig, wir haben eine Alternante gefunden. Falls nicht, verschieben wir die Punkte (im Beispiel wird sofort klar, wie), und machen mit der neuen Verteilung weiter.
Zunächst aber

Lemma 8.17
Sei $x_0 < \ldots < x_{n+1}$. Dann gibt es genau ein Polynom $p \in \mathcal{P}_n$ und ein $D \in \mathbb{R}$, so dass

$$p(x_k) - f(x_k) = (-1)^k D \,\forall\, k = 0 \ldots n+1.$$

Beweis: D und die Koeffizienten von p erfüllen ein lineares Gleichungssystems. Die Annahme, dass dieses zwei verschiedene Lösungen hat, führt sofort zum Widerspruch, also ist das Gleichungssystem eindeutig lösbar. □

Korollar 8.18 *Eine Bestapproximation p^* ist eindeutig durch ihre Alternante charakterisiert. Bei gegebener Alternante kann p^* durch Lösen eines linearen Gleichungssystems in $(n+2)$ Variablen bestimmt werden.*

Korollar 8.19 *Die Bestapproximation in $\mathcal{P}_n$ bezüglich $||\cdot||_\infty$ ist eindeutig.*

Beweis: Die Existenz wurde bereits gezeigt. Seien p_1 und p_2 zwei Bestapproximationen. Dann ist wie in 8.6 auch

$$p^* = \frac{1}{2}(p_1 + p_2)$$

eine Bestapproximation. Nach Satz 8.14 besitzt $f - p_*$ eine Alternante $x_0, \ldots, x_{n+1}$. Dann ist dies aber auch eine Alternante von $f - p_1$ und $f - p_2$, die die Bestapproximation eindeutig charakterisiert. □

Falls in Lemma 8.17 $|D| = ||p - f||_\infty$, so sind wir fertig, denn dann ist nach dem Alternantensatz p eine Bestapproximation. Falls nicht, so gilt $|D| < ||p - f||$. Dies motiviert den folgenden Algorithmus:

1. Wähle x_k gleichverteilt im Intervall $[a, b]$.
2. Berechne D und p nach Lemma 8.17.
3. Falls $|D| = ||p - f||_\infty$, so ist p eine Bestapproximation.
4. Falls $D < ||p - f||_\infty$, so verschiebe die x_k leicht so, dass der Abstand zwischen p und f an diesen Stellen größer wird.
5. Gehe zurück zu 2.

Man kann zeigen, dass unter Bedingungen der Remez–Algorithmus konvergiert gegen die Tschebyscheff–Approximation. Wir schauen nur kurz auf ein Beispiel. Wir betrachten wieder den Sinus auf $[0, \pi/2]$ und suchen eine lineare Approximation

(wie oben). Im ersten Schritt wählen wir die x_k gleichverteilt und berechnen die zugehörige Approximationsfunktion wie im Lemma.

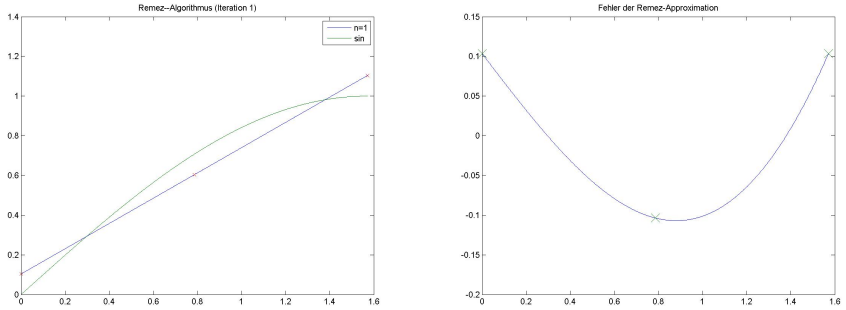

Abbildung 8.11: Erste Iteration des Remez–Algorithmus

Klick für Bild Remeziter1
Klick für Matlab Figure Remeziter1
Klick für Bild Remeziterdiff1
Klick für Matlab Figure Remeziterdiff1

Wir sehen, dass wir noch keine Alternante erreicht haben. Am mittleren Interpolationspunkt (x_1) wird noch nicht das Maximum des Abstands angenommen. Wir verschieben also nur diesen Punkt etwas nach rechts, wiederholen den Algorithmus mit dieser Anordnung und erhalten

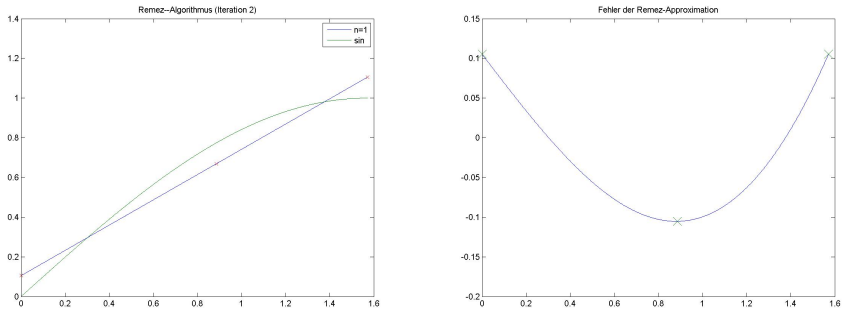

Abbildung 8.12: Zweite Iteration des Remez–Algorithmus

Klick für Bild Remeziter2
Klick für Matlab Figure Remeziter2
Klick für Bild Remeziterdiff2
Klick für Matlab Figure Remeziterdiff2

Tatsächlich bekommen wir eine Alternante. Das zugehörige Polynom ist eine Tschebyscheff–Approximation. Die folgenden Programme sind einfache Implementationen des Remez–Algorithmus.

```
function [ output_args ] = remezalgo( n )
%REMEZALGO Very simple implementation of the Remez algorithm
if (nargin<1)
    n=3;
end
differenz=1e-2;
```

Listing 8.7: Simple Implementation des Remez–Algorithmus mit dem Standardbeispiel (Approximation/remezalgo.m)

Klicken für den Quellcode von Approximation/remezalgo.m

```
function [ output_args ] = remezalgo( n )
%REMEZALGO Very simple implementation of the Remez algorithm
%Application to the approximation of sin(x)/x by a polynomial in x*x
%(see Abramowitz/Stegun, 4.3.96).
if (nargin<1)
    n=3;
```

Listing 8.8: Remez–Algorithmus angewandt auf Abramowitz and Stegun [1965] 4.3.96 (Approximation/remezalgoabramo.m)

Klicken für den Quellcode von Approximation/remezalgoabramo.m

Maple besitzt eine Implementation des Remez–Algorithmus. Wir nutzen sie, um für unser Standardbeispiel der Approximation des Sinus mit kubischen Polynomen den Fehler der Tschebyscheff–Interpolation anzugeben.

	maximaler Fehler
Tschebyscheff	0.0014

```
function [ output_args ] = mapleremez( input_args )
%MAPLEREMEZ
N=1000;
x=(0:N)/N*pi/2;
y=zeros(1,N+1);
maple('with(numapprox)')
```

Listing 8.9: Remez–Algorithmus in Maple (Approximation/mapleremez.m)

Klicken für den Quellcode von Approximation/mapleremez.m

Das berechnete Polynom ist

$$p^*(x) = -0.0013670643 + 1.025256091 * x - 0.0706895833 * x^2 - 0.1125059808 * x^3.$$

Zumindest in diesem Fall konnten wir die bereits durch die Tschebyscheff–Polynome gelieferte Gauss–Approximation also leider nur noch marginal verbessern. Das bereits für die Gauss–Approximation Gesagte gilt auch hier: Wir sind keineswegs auf Polynome beschränkt. Die meisten Sätze gelten für alle Funktionenräume, bei denen die Interpolationsaufgabe eindeutig lösbar ist (Haarsche Systeme). Die Maple–Routinen etwa lassen gleich eine rationale Approximation zu. Die Standard–Tafelwerke (etwa Abramowitz and Stegun [1965]) geben hochgenaue Remez–Approximationen für die elementaren Funktionen an.

8.4 Der Approximationssatz von Weierstrass

Der letzte Abschnitt hat uns gezeigt, dass wir die Bestapproximation p_n^* bezüglich der Unendlichnorm an eine stetige Funktion f in $\mathcal{P}_n$ berechnen können. Es stellt sich die Frage, ob man durch ausreichend hohe Polynomgrade f beliebig genau approximieren kann, d.h. ob

$$||f - p_n^*||_\infty \mapsto 0,\ n \mapsto \infty.$$

Dies ist äquivalent zu der Frage, ob die Polynome dicht liegen im Raum der stetigen Funktionen. Diese Frage wurde von Weierstrass beantwortet – in Münster kommt man nicht darum herum, dies zumindest zu erwähnen. Beweise finden sich zuhauf in der Literatur (und gehören eigentlich in andere Vorlesungen), wir skizzieren hier nur kurz die Beweisidee. Im Folgenden sei immer X der Vektorraum der stetigen Funktionen auf $I = [a, b]$ (ohne Einschränkung $a = 0$ und $b = 1$) und $|| \cdot ||$ die Unendlichnorm.

Definition 8.20 *(**monotone Operatoren**)*
Sei $L : X \mapsto X$ ein linearer Operator. L heißt monoton, falls für alle $f, g \in X$

$$(f(t) \leq g(t) \forall t \in I) \Rightarrow ((L(f))(t) \leq (L(g))(t) \forall t \in I).$$

Als Beispiel für einen positiven Operator betrachten wir den Bernsteinoperator.

Definition 8.21 *(**Bernsteinoperator**)*
Sei X der Raum der stetigen Funktionen auf dem Intervall $I = [0, 1]$. Der Operator

$$B_n : X \mapsto \mathcal{P}_n,\ (B_n(f))(t) = \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) t^k (1-t)^{n-k},\ n \geq 1$$

heißt Bernsteinoperator.

Lemma 8.22 *(Eigenschaften des Bernsteinoperators)*

1. *B_n ist monoton.*
2. *Sei $p_k(x) = x^k$.*

$$(B_n(p_0)) = p_0$$
$$(B_n(p_1)) = p_1.$$
$$(B_n(p_2))(t) = p_2(t) + \frac{t - t^2}{n}.$$

$B_n(p_k)$ konvergiert also gleichmäßig gegen p_k, $k = 0, 1, 2$.

Beweis:

1. Sei $f \leq g$, dann ist insbesondere $f(k/n) \leq g(k/n)$. Der Binomialkoeffizient ist nichtnegativ, ebenso $t^k(1-t)^{n-k}$, also ist

$$(B_n(f))(t) = \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) t^k(1-t)^{n-k} \leq \sum_{k=0}^{n} \binom{n}{k} g\left(\frac{k}{n}\right) t^k(1-t)^{n-k} = (B_n(g))(t).$$

2. Es gilt

$$1 = (t + (1-t))^n = \sum_{k=0}^{n} \binom{n}{k} t^k(1-t)^{n-k} = (B_n(p_0))(t).$$

Weiter gilt

$$\begin{aligned}(B_n(p_1))(t) &= \sum_{k=0}^{n} \binom{n}{k} \frac{k}{n} t^k(1-t)^{n-k} \\ &= \sum_{k=1}^{n} \binom{n-1}{k-1} t^k(1-t)^{n-k} \\ &= t\sum_{k=0}^{n-1} \binom{n-1}{k} t^k(1-t)^{n-1-k} \\ &= t\end{aligned}$$

und schließlich

$$\begin{aligned}(B_n(t(1-t)))(t) &= \sum_{k=0}^{n} \binom{n}{k} \frac{k}{n} \frac{n-k}{n} t^k(1-t)^{n-k} \\ &= \frac{n(n-1)}{n^2} t(1-t) \sum_{k=0}^{n-2} \binom{n-2}{k} t^k(1-t)^{n-2-k} \\ &= (1 - \frac{1}{n}) t(1-t).\end{aligned}$$

□

Der entscheidende Hilfssatz ist

Lemma 8.23 *(**Lemma von Korovkin**)*
Sei $L_n : X \mapsto X$ eine Folge linearer, monotoner Operatoren. Es sei $p_k(t) = t^k$. Falls

$$||L_n(p_k) - p_k||_\infty \mapsto 0,\ k = 0, 1, 2,$$

so gilt bereits

$$||L_n(f) - f||_\infty \mapsto 0$$

für alle $f \in X$.

Die Voraussetzungen dieses Lemmas sind für den Bernsteinoperator erfüllt. Falls wir es zeigen können, liefert also der Bernsteinoperator für jede stetige Funktion f eine Polynomfolge, die gleichmäßig gegen f konvergiert.
Beweis: Der Beweis findet sich z.B. in Meinardus [1964], Seite 6, und deutlich ausführlicher online im Skript Oberle [2007], Kapitel 4.
Wir folgen dem eleganten Beweis von Meinardus. Sei zunächst $\epsilon > 0$ fest gewählt. f ist stetig auf einem Kompaktum, also gleichmäßig stetig. Damit gibt es ein $\delta > 0$, so dass

$$|f(t) - f(x)| \leq \epsilon \, \forall |t - x| \leq \delta.$$

Für $|t - x| \geq \delta$ gilt

$$|f(t) - f(x)| \leq 2||f|| \leq \frac{2||f||}{\delta^2}(t - x)^2,$$

insgesamt also sicherlich

$$|f(t) - f(x)| \leq \epsilon + \frac{2||f||}{\delta^2}(t - x)^2.$$

Wir betrachten die linke und rechte Seite als Funktionen in t (setzen also x konstant) und nutzen aus, dass L_n monoton und linear ist. Dann gilt wegen

$$|z| \leq u \Leftrightarrow -u \leq z \leq u$$

auch

$$|L_n(f)(t) - f(x)(L_n(1))(t)| \leq \epsilon(L_n(1))(t) + \frac{2||f||}{\delta^2}(L_n((t - x)^2))(t).$$

Unabhängig von t und x konvergieren nach Voraussetzung $L_n(1)$ und $L_n((t - x)^2)$ gleichmäßig gegen 1 bzw. $(t - x)^2$. Sei $\epsilon' > 0$. Dann gilt

$$|(L_n(f))(t) - f(x)| \leq \epsilon + \frac{2||f||}{\delta^2}(t - x)^2 + \epsilon'$$

für $n > n_0(\epsilon')$. Ausgewertet für $t = x$ gilt damit

$$|(L_n(f))(x) - f(x)| \leq \epsilon + \epsilon'.$$

□

Wir folgern nun aus 8.22 und dem Lemma von Korovkin

Satz 8.24 *(**Satz von Weierstrass**)*

$$B_n(f) \mapsto f \, \forall f \in X.$$

Insbesondere liegen wegen $B_n(f) \in \mathcal{P}_n$ die Polynome dicht in X.

Der Wert der Approximation $B_n(f)$ an f liegt ausschließlich in diesem theoretischen Ergebnis. Numerisch liefert die Approximation wesentlich schlechtere Werte als die Tschebyscheff–Approximation (die ja optimal ist).

Kapitel 9

Grundzüge der linearen und nichtlinearen Optimierung

Viele Aufgaben der angewandten Mathematik führen auf Probleme, bei denen wir einen Wert suchen, für den eine gegebene Zielfunktion ihr Maximum oder Minimum annimmt. Einige Beispiele haben wir bereits kennengelernt, etwa

Beispiel 9.1

kleinste Quadrate–Lösungen (4.2): Hier konnten wir mit Hilfe der Gaussschen Normalgleichungen 4.4 die Lösung direkt angeben.

Minimum Norm–Lösungen: Hier haben wir den kleinsten Vektor v mit der Nebenbedingung, dass v kleinste Quadrate–Lösung ist, berechnet.

Krylovraumverfahren (6.1): Hier haben wir die Lösung des Minimierungsproblems numerisch durch Gradientenverfahren berechnet.

Uns interessieren natürlich die Probleme mit numerischen Lösungsverfahren. Grob lassen sich die Optimierungsaufgaben in mindestens zwei Kategorien einteilen.

1. **Unbeschränkte Optimierung**: Dies sind Aufgaben der Form

   $$\min_{x \in V} f(x)$$

   wobei V einen kompletten Raum darstellt. Dies ist bei den kleinste Quadrate–Lösungen und den Krylovraumverfahren der Fall.

2. **Optimierung mit Nebenbedingungen**: In der Praxis ist der Raum, in dem wir die Kandidaten für die Minimierung suchen, fast nie unbeschränkt, sondern unterliegt Einschränkungen, die aus der Anwendung stammen. Das einfachste Beispiel sind die Minimum Norm–Lösungen, bei denen wir den Raum auf

die Menge der kleinsten Quadrate–Lösungen einschränken. Wir haben also Aufgaben der Form

$$\min_{x \in V} f(x) \text{ unter } g(x) = 0 \text{ und } h(x) \leq 0.$$

Die verwendeten (iterativen) numerischen Methoden unterscheiden sich deutlich. In der unbeschränkten Optimierung könnten wir etwa Gradientenmethoden verwenden, oder Nullstellen der Ableitung von f mit Hilfe des Newtonverfahrens suchen. Bei der beschränkten Optimierung müssen wir hingegen immer sicherstellen, dass die Nebenbedingungen von den Kandidaten noch erfüllt werden. Wir schauen auf den klassischen, in der Anwendung immer noch häufig auftretenden Spezialfall der linearen Optimierung. Hier sind sowohl die Nebenbedingungen g und h wie auch die Zielfunktion h linear. Es ist sofort klar, dass erst die Nebenbedingungen die Existenz eines Minimums garantieren.

9.1 Lineare Optimierung

Lineare Optimierungsaufgaben stammen häufig aus der Chemie oder den Wirtschaftswissenschaften. Die einfachsten Prototypen sind von der Form:
In einem Chemiekonzern werden zwei Chemikalien X und Y hergestellt. Bei der Herstellung von X fallen pro Liter $S_{1,X}, S_{2,X}, \ldots$ mg. Für die Schadstoffe gelten die Grenzwerte $S_1, S_2, \ldots$. Weiter werden pro Liter $R_{1,X}, R_{2,X}, \ldots$ kg Rohstoffe benötigt, entsprechend für Y. Zusätzlich gibt es eine Maschine, die nur einmal zur Verfügung steht, aber 24 Stunden laufen muss, sie kann pro Stunde entweder die Menge M_X von X oder die Menge M_Y von X herstellen.
Der Konzern erwirtschafte pro kg der Chemikalie einen Gewinn von G_X bzw G_Y. Welcher Produktionsplan garantiert den höchsten Gewinn?
Dies ist leicht in einen mathematische Formulierung zu bringen:
Maximiere

$$f(X, Y) = G_X X + G_Y X$$

unter den Nebenbedingungen

$$\begin{aligned} X, Y &\geq 0 \\ S_{k,X} \cdot X + S_{k,Y} \cdot Y &\leq S_k,\ k = 1, \ldots, N \\ (1/M_X) \cdot X + (1/M_Y) \cdot Y &= 24 \end{aligned}$$

Mit

$$x = (X, Y)$$

und offensichtlicher Definition für die restlichen Variablen formulieren wir dies um zu

$$\max x^t G \text{ unter } Ax \leq b,\ x \geq 0.$$

Die offensichtliche Art, die Gleichung einzubauen, wäre, einfach entweder X oder Y komplett aus den Gleichungen zu eliminieren. Alternativ könnten wir die Gleichung $z = C$ auch durch $Z - C \geq 0$ und $C - Z \geq 0$ modellieren.
Für ein kurzes Beispiel wählen wir

$$A = \begin{pmatrix} 20 & 10 \\ 4 & 5 \\ 6 & 15 \end{pmatrix},\ b = \begin{pmatrix} 8000 \\ 2000 \\ 4500 \end{pmatrix},\ G = \begin{pmatrix} 16 \\ 32 \end{pmatrix}$$

In unseren zwei Variablen ist das Problem leicht lösbar. Wir zeichnen zunächst den zulässigen Bereich.
TODO
Unser Gesamtgewinn ist $f(x) = G^t x$. Wir betrachten die Produktionsplankombinationen, die den gleichen Gewinn C garantieren. Diese erfüllen die Gleichung

$$Y(G, X) = G/32 - X/2.$$

Wir suchen also diejenige Gerade, für die G möglichst groß ist, aber noch so, dass die zugehörige Gerade mit dem zulässigen Bereich noch mindestens einen Punkt gemeinsam hat. Alle Geraden liegen parallel, wir müssen also nur eine einzeichnen und sie möglichst weit nach oben verschieben. Anschaulich ist klar, dass der Maximalpunkt entweder eine Ecke ist, oder (falls eine begrenzende Gerade parallel zur Gewinngeraden verläuft) eine Kante – in diesem Fall enthält sie sogar zwei Ecken. In jedem Fall wird der Maximalgewinn in einem Eckpunkt des zulässigen Gebiets angenommen. Wir können uns also in unseren Betrachtungen auf die Untersuchung der Eckpunkte des zulässigen Gebiets beschränken.
Diese grafissche Vorgehensweise ist natürlich nur im $\mathbb{R}^2$ durchführbar. Wir betrachten die **Normalform**:
Sei $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $f(x) := c^t x$ und

$$M := \{x \in \mathbb{R}^n : x_j \geq 0,\ j = 1 \dots n \text{ und } Ax = b\}$$

die Menge der zulässigen Punkte. Bestimme $x_M \in \mathbb{R}^n$, so dass

$$f(x_m) \geq f(x) \forall\, x \in M.$$

Dies deckt alle linearen Aufgaben ab. Ist etwa x_k unbeschränkt, so ersetzen wir $x_k = y_k - z_k$ und wir können ungestraft die Positivität der neuen Variablen annehmen.

Falls wir eine Ungleichungnebenbedingung der Form $(Ax)_j \leq b_j$ haben, so führen wir eine neue (Schlupf–) Variable u_j ein und schreiben

$$(Ax)_j + u_j = b_j$$

was mit der Bedingung $u_j \geq 0$ äquivalent zur alten Bedingung ist. Insgesamt erhalten wir in diesen neuen Variablen dann ein Gleichungssystem (unter Umständen mit wesentlich mehr Variablen) in Normalform, das nur nichtnegative Variable und Gleichungen (keine Ungleichungen) enthält.
Offensichtlich macht die Minimierungsaufgabe nur Sinn, wenn $n >$ Rang A.
M ist konvex. Die Ecken von M definieren wir geometrisch mit

Definition 9.2 *(Ecken des zulässigen Gebiets)*
$x \in M$ heißt Ecke von M, wenn es nicht als Konvexkombination zweier Punkte x_1 und x_2 von M dargestellt werden kann, die verschieden von x sind.

Wir bemerken ohne Beweis

Satz 9.3 *Sei M konvex und beschränkt. Dann ist M die konvexe Hülle seiner Ecken.*

Beweisidee: Per Induktion über die Dimension des zugrunde liegenden Raums. Sei $x \in M$. Falls x auf dem Rand von M liegt (einer $(n-1)$–dimensionalen Untermannigfaltigkeit), so ist x Konvexkombination der Ecken des Randes nach Induktionsvoraussetzung. Falls nicht, so ist x Konvexkombination zweier Punkte $x^{(1)}$ und $x^{(2)}$. Die Gerade durch diese Punkte schneidet den Rand in zwei Punkten, die wieder nach Induktionsvoraussetzung Konvexkombination von Ecken des Randes sind, und damit ist auch x Konvexkombination von Punkten des Randes.
Damit gilt sofort der

Satz 9.4 *(Eckensatz)*
Sei M beschrönkt. Dann gibt es eine Ecke von M, die Lösung der Optimierungsaufgabe ist.

Beweis: f ist stetig auf einem Kompaktum, also wird das Maximum angenommen. Sei z Lösung der Optimierungsaufgabe. z ist Konvexkombination von Ecken von M, also

$$z = \sum_k \lambda_k x^{(k)},\ 0 < \lambda_k,\ \sum_k \lambda_k = 1.$$

Damit gilt aber auch

$$f(z) = \sum_k \lambda_k f(x^{(k)})$$

und damit $f(x^{(k)}) = f(z)$. □

Dies motiviert sofort einen Algorithmus: Berechne alle Ecken von M und die zugehörigen Werte von f. Die Ecke, die das Maximum liefert, löst die Optimierungsaufgabe.
Um dies zu realisieren, benötigen wir zunächst

Satz 9.5 *(Charakterisierung der Ecken)*
x ist genau dann Ecke von M, wenn es eine Indexmenge $I \subset \{1 \dots n\}$ mit $|I| = m$ gibt, so dass

1. $x \geq 0$.
2. $Ax = b$.
3. $x_i = 0 \, \forall i \notin I$.
4. *Die Spalten a_i, $i \in I$, sind linear unabhängig.*

I heißt Basis der Ecke, $\{1 \dots n\} \setminus I$ heißt Nichtbasis.

Schauen wir kurz auf unser grafisches Beispiel. Dort hatten wir nur Ungleichungen, die wir durch Schlupfvariable realisieren, d.h. wir haben die Variablen x_1, x_2, u_1, u_2 und u_3. Die Ecken lagen jeweils auf zwei begrenzenden Geraden. Die dritte Gleichung war dort nicht erfüllt. Zu den Ecken unserer Grafik gehört also jeweils ein Lösungstupel, bei dem höchstens x_1, x_2 und ein u_k nicht verschwinden, also gerade drei ($= m$).
Beweis:

1. Es gelten 1-4. Angenommen, x sei keine Ecke. Dann ist x Konvexkombination zweier verschiedener Punkte $x^{(1)}$ und $x^{(2)}$. Wegen $x_i = 0$, $i \notin I$, gilt auch $x_i^{(1)} = x_i^{(2)} = 0$ (denn alle Komponenten der Vektoren sind nichtnegativ), und damit $x_i = x_i^{(1)} = x_i^{(2)}$ für $i \notin I$.
   Die restlichen Unbekannten lösen das Gleichungssystem

   $$b = Ax = \sum_{i \in I} x_i a_i$$

   und entsprechend für $x^{(1)}$ und $x^{(2)}$. Da die a_i linear unabhängig sind und $|I| = m$, gilt auch $x_i^{(1)} = x_i^{(2)}$ für $i \in I$, also gilt $x^{(1)} = x^{(2)}$ im Widerspruch zur Annahme.

2. Sei x eine Ecke von M. Sei I die Indexmenge der Koordinaten mit $x_i > 0$. Angenommen, die zugehörigen a_i seien linear abhängig. Dann gibt es eine nichttriviale Linearkombination

   $$A\alpha = \sum_i \alpha_i a_i = 0$$

mit $a_i = 0$ für $i \notin I$. Sei $x^\epsilon = x + \epsilon\alpha$. Dann gilt $Ax^\epsilon = b$. Für ausreichend kleines $|\epsilon|$ bleibt die Lösung also in M. Damit wäre aber x Linearkombination von $x \pm \epsilon\alpha$ und damit keine Ecke. Also sind die zugehörigen a_i linear unabhängig, insbesondere sind es höchstens m. Falls es weniger sind, ergänzen wir I entsprechend.

□

9.2 Simplex–Verfahren

Zur Herleitung des Simplex–Verfahrens betrachten wir zunächst nur den einfachsten Fall. In der Normalform stamme jede Gleichung aus einer Ungleichung, die wir mit Hilfe einer Schlupfvariablen y_k umgewandelt haben, $k = 1 \dots m$. Weiter sei $b_k > 0$, $k = 1 \dots m$. Dann ist $(0, b)^t$ eine Ecke von M. Ihre Basis ist $y_1 \dots y_m$, ihre Nichtbasis ist $x_1 \dots x_n$.
Die Idee des Simplex–Verfahrens ist, ausgehend von dieser Ecke eine Folge von zulässigen Ecken zu konstruieren, bei denen der Wert des Zielfunktionals ansteigt. Hierbei tauschen wir jeweils eine Variable aus der Basis mit einer aus der Nichtbasis. Es gilt für alle Elemente aus M

$$Ax + y = b, \text{ also } \begin{pmatrix} A & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = b$$

oder in der p. Zeile

$$\sum_k (a_k)_p x_k + y_p = b_p. \tag{9.1}$$

Wir wollen nun x_s anstelle von y_p in die Basis bringen, d.h. wir suchen eine neue Ecke $(\tilde{x}, \tilde{y})$, so dass $\tilde{y}_p = 0$ und $\tilde{x}_s \neq 0$ und nach wie vor $\tilde{x}_k = 0$ für $k \neq s$.
Diese ist leicht bestimmt: Wir setzen $(\tilde{x}, \tilde{y})$ in 9.1 ein und erhalten wegen $\tilde{y}_p = 0$

$$\tilde{x}_s = \frac{b_p}{(a_s)_p}.$$

Für $k \neq p$ ergibt sich entsprechend

$$\tilde{y}_k = b_k - (a_s)_k \tilde{x}_s.$$

Damit haben wir bereits die neue Ecke gefunden. Dies war aber nur deshalb so leicht, weil unser Gleichungssystem eine so einfache Form hatte, die wir jetzt natürlich zerstört haben. Wir stellen diese Form wieder her, indem wir die Positionen von x_s und y_p im Vektor vertauschen (was wir uns natürlich merken müssen).

Dadurch steht im Vektor die Nichtbasis wieder oben, die Basis unten. In der Matrix müssen wir die zugehörigen Spalten vertauschen, wir erhalten

$$(a_1 \dots a_{s-1} e_p a_{s+1} \dots a_n e_1 \dots e_{p-1} a_s e_{p+1} \dots e_m).$$

Wir haben durch die Vertauschung also die einfache Form der Matrix zerstört. Diese stellen wir nun durch Zeilenoperationen wieder her.

1. Damit im rechten Teil der Matrix auf der Hauptdiagonalen wieder eine 1 steht, multiplizieren wir die $p.$ Gleichung mit $1/(a_s)_p$.
2. Damit im rechten Teil der Matrix in der $p.$ Spalte die Elemente außerhalb der Hauptdiagonalen verschwinden, ziehen wir für $k \neq p$ das $(a_s)_k$–fache der $p.$ Gleichung von der $k.$ Gleichung ab.

Hier wie im Folgenden machen wir natürlich stillschweigend die Annahme, dass das Pivotelement nicht verschwindet. Darauf kann man verzichten (Stichwort: Berechnung von entarteten Lösungen).
Damit haben wir hier die Form fast wiederhergestellt: Wir müssen noch bei der Auswahl von s und p sicherstellen, dass die rechte Seite positiv bleibt.
Bleibt noch die Zielfunktion. Sie war vor der Vertauschung natürlich eine Funktion in den ersten n Variablen des Vektors, das wollen wir auch beibehalten. Wieder gilt mit 9.1

$$(a_s)_p x_s = b_p - y_p - \sum_{k \neq s} (a_k)_p x_k.$$

Wir können also x_s in der Zielfunktion durch y_p ersetzen. Es gilt

$$f(x,y) = c^t x = \tilde{c} \begin{pmatrix} x_1 \\ \vdots \\ x_{s-1} \\ y_p \\ x_{s+1} \\ \vdots \\ x_n \end{pmatrix} + c_s \frac{b_p}{(a_s)_p}$$

mit

$$\tilde{c}_s = -\frac{1}{(a_s)_p},\ \tilde{c}_k = c_k - \frac{(a_k)_p}{(a_s)_p} (k \neq s).$$

Der Funktionswert des Zielfunktionals an der neuen Ecke ist damit insbesondere

$$c_s \frac{b_p}{(a_s)_p}.$$

Falls dieser Wert positiv ist für ein Pärchen (s, p), so steigt der Funktionswert in dieser benachbarten Ecke an und wir müssen nur noch sicherstellen, dass die neue Ecke die Voraussetzung erfüllt, dass die rechte Seite positiv ist.
Mit dieser neuen Ecke können wir nun den Algorithmus fortführen. Üblicherweise wird dies in einem Tableau (dem Simplex–Tableau) durchgeführt.

Kapitel 10

Ausblick

Übersicht über weiterführende Vorlesungen/Themen der Angewandten Mathematik.

Literaturverzeichnis

M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables : [is an Outgrowth of a Conference on Mathematical Tables Held at Cambridge, Mass., on 1954*. Applied mathematics series. Dover Publ., 1965. ISBN 9780486612720. URL http://people.math.sfu.ca/~cbm/aands/abramowitz_and_stegun.pdf.

D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer-Verlag Berlin Heidelberg, 2007. ISBN 9783540724506. URL http://books.google.de/books?id=s1M-jAqi1sYC.

J. G. F. Francis. The qr transformation a unitary analogue to the lr transformation—part 1. *The Computer Journal*, 4(3):265–271, 1961. doi: 10.1093/comjnl/4.3.265. URL http://comjnl.oxfordjournals.org/content/4/3/265.abstract.

J. G. F. Francis. The qr transformation—part 2. *The Computer Journal*, 4(4):332–345, 1962. URL http://comjnl.oxfordjournals.org/content/4/4/332.abstract.

Gene Golub and Frank Uhlig. The qr algorithm: 50 years later its genesis by john francis and vera kublanovskaya and subsequent developments. *IMA Journal of Numerical Analysis*, 29(3):467–485, 2009. doi: 10.1093/imanum/drp012. URL http://imajna.oxfordjournals.org/content/29/3/467.abstract.

A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987. ISBN 9780898713961. URL http://books.google.de/books?id=WwMDNLxrwocC.

Martin H. Gutknecht and Beresford N. Parlett. From qd to LR, or, how were the qd and LR algorithms discovered? *IMA J. Numer. Anal.*, 31(3):741–754, 2011. doi: 10.1093/imanum/drq003. URL http://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CDgQFjAA&url=http%3A%2F%2Fwww.sam.math.ethz.ch%2F~mhg%2Ftalks%2FqdLRlongHO.pdf&ei=kT_QUJ6tDM3JswbopoDYBQ&usg=AFQjCNEO_009PVIL7orytkMjte6KQPIWZg&sig2=pDTqxytaJzeay1RfnQKYFg&bvm=bv.1355534169,d.Yms.

M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Mathematische Leitfäden. Teubner, 2006. ISBN 9783835100909. URL http://books.google.de/books?id=tKrhTUmYNEoC.

John Harrison. Formal verification of ia-64 division algorithms. In *Proceedings of the 13th International Conference on Theorem Proving in Higher Order Logics*, TPHOLs '00, pages 233–251, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67863-8. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.7123&rep=rep1&type=pdf.

K. R. James. Convergence of matrix iterations subject to diagonal dominance. *SIAM Journal on Numerical Analysis*, 10(3):pp. 478–484, 1973. ISSN 00361429. URL http://www.jstor.org/stable/2156114.

T. Kato. *Perturbation Theory of Linear Operators*. Classics in mathematics. Springer, 1995. URL http://books.google.de/books?id=zzNqMgEACAAJ.

K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Applied Math.*, 2:164–168, 1944.

D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030. URL http://www.jstor.org/stable/2098941.

P. J. McKenna. Large torsional oscillations in suspension bridges revisited: Fixing an old approximation. *The American Mathematical Monthly*, 106(1):pp. 1–18, 1999. ISSN 00029890. URL http://scholar.google.de/scholar_url?hl=de&q=http://actuarialscience.math.uconn.edu/~mckenna/2410f09/monthly1.pdf&sa=X&scisig=AAGBfm1Wn1ubV1hU3qHMj1bRcNul9-GvDQ&oi=scholarr&ei=99zKUJMWyNWyBom4gbgC&ved=0CDUQgAMoADAA.

G. Meinardus. *Approximation von Funktionen und ihre numerische Behandlung*. Springer Tracts in Natural Philosophy. New York, 1964. URL http://books.google.de/books?id=1Fs0o90hn_AC.

J.M. Muller. *Elementary Functions: Algorithms and Implementation*. Computer Science. Birkhäuser Boston, 2005. ISBN 9780817643720. URL http://books.google.de/books?id=g3AlWip4R38C.

F. Natterer. *The Mathematics of Computerized Tomography*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2001. ISBN 9780898714937. URL http://books.google.de/books?id=gjSO1hLbcD0C.

F. Natterer and F. Wübbeling. *Mathematical Methods in Image Reconstruction*. Monographs on Mathematical Modeling and Computation, No 5 Series. Society for Industrial & Applied, 2001. ISBN 9780898714722. URL http://books.google.de/books?id=u8W9I32wNRMC.

H.J. Oberle. *Skript zur Vorlesung Approximation*. 2007. URL http://www.math.uni-hamburg.de/home/oberle/skripte/approximation.html.

Thomas Risse. Cordic-algorithmen verbinden mathematik, computer-architektur und anwendungen. *Global J. of Engng. Educ.*, 8(3), 2004. URL http://www.wiete.com.au/journals/GJEE/Publish/TOCVol8No3.html.

H. Rutishauser. Solution of eigenvalue problems with the LR transformation. In *Applied Mathematics Series*, volume 49, pages 47–81. 1958.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003. ISBN 9780898715347. URL http://books.google.de/books?id=Uoe7xBOhS5AC.

R. Schaback and H. Wendland. *Numerische Mathematik*. Springer-Lehrbuch. Springer, 2004. ISBN 9783540213949. URL http://books.google.de/books?id=hH0Y09xuJjcC.

J. Steppeler, R. Hess, U. Schättler, and L. Bonaventura. Review of numerical methods for nonhydrostatic weather prediction models. *Meteorology and Atmospheric Physics*, 82:287–301, 2003. ISSN 0177-7971. doi: 10.1007/s00703-001-0593-8. URL http://scholar.google.de/scholar_url?hl=de&q=http://dvgu.ru/meteo/library/30820287.pdf&sa=X&scisig=AAGBfm3oRnh-l7aCUJCqMOVvhjiLQLlUjQ&oi=scholarr&ei=rte1UPpkpoziBOf9gbgL&ved=0CDIQgAMoATAA.

Abbildungsverzeichnis

Listings