Vorlesungsmitschrieb

Numerik II

Prof. Dr. Christian Lubich

im Sommersemester 2007
überarbeitet im Sommersemester 2009
an der Eberhard-Karls-Universtität Tübingen

gesetzt von Markus Klein mit LaTeX

Letzte Änderung: 12. Februar 2010

Vorwort

Dieses Scriptum entstand als Live-Mitschrieb im Sommersemester 2007 und mit Korrekturen im Sommersemester 2009 bei Prof. Dr. Christian Lubich an der Eberhard-Karls-Universität Tübingen. Ich danke an dieser Stelle Frau Petra Arnold und Frau Karin Schaller, die zahlreiche Fehler gefunden haben.
Es erhebt keinen Anspruch auf Vollständigkeit oder Richtigkeit. Es ist *nicht* durch Prof. Lubich autorisiert.

Bei Fragen, Wünschen oder Verbesserungsvorschlägen freue ich mich über jede E-Mail an
studium@kleiner-markus.de.
Vielen Dank!

Inhaltsverzeichnis

1 Schnelle Fourier-Transformation

1.1 Diskrete Fourier-Transformation

Wir betrachten endliche Folgen $x = (x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$ perdiodisch fortgesetzt auf beliebigen ganzzahlige Indizes (bei Bedarf). Wir setzen also $x_k = x_\ell$, falls $k \equiv \ell \mod N$.

1.1 Definition (diskrete Fouriertransformation)
Die Abbildung $\mathcal{F}_N : \mathbb{C}^N \to \mathbb{C}^N$ wird definiert durch $\mathcal{F}_N x = \hat{x}$ mit

$$\hat{x}_k = \sum_{j=0}^{N-1} w_N^{k \cdot j} x_j$$

wobei $w_N = e^{\imath \cdot \frac{2\pi}{N}}$ die primitive N-te komplexe Einheitswurzel ist, d.h. $w_N^N = 1$.

1.2 Bemerkung
Ist klar, um was es sich bei N handelt, schreiben wir auch nur w für die Einheitswurzel.

1.3 Bemerkung (Rechenaufwand)
Zur direkten Berechnung der diskreten Fouriertransformation benötigt man etwa N^2 Operationen (an Multiplikationen und Additionen).

Wir werden später sehen, daß die schnelle Fouriertransformation (FFT) etwa $N \log_2 N$ Operationen benötigt, falls $N = 2^L$ ist.

1.4 Hilfssatz (Orthogonalitätsrelation der diskreten Fouriertransformation)
Es gilt:

$$\sum_{k=0}^{N-1} w_N^{k\ell} \bar{w}_N^{km} = \begin{cases} N, & \ell \equiv m \mod N \\ 0, & \text{sonst} \end{cases}$$

Beweis
Sei $\bar{w} = w^{-1}$. Für $l \equiv m$ ist

$$\sum_{k=0}^{N-1} 1 = N$$

sonst ergibt sich

$$\sum_{k=0}^{N-1} w^{k\ell} w^{-km} = \sum_{k=0}^{N-1} w^{k(\ell-m)} = \frac{1-(w^{l-m})^N}{1-w^{\ell-m}} = 0$$

□

1.5 Satz (Parseval-Gleichung)
Sei $\mathbb{C}^N$ mit der euklidische Norm versehen. Dann gilt:

$$\left\| \frac{1}{\sqrt{N}} \mathcal{F}_N x \right\| = \|x\| \quad \forall x \in \mathbb{C}^N$$

d.h. dies ist eine Isometrie/unitäre Abbildung.

Ausgeschrieben heißt dies:

$$\frac{1}{N} \sum_{k=0}^{N-1} |\hat{x}_k|^2 = \sum_{j=0}^{N-1} |x_j|^2$$

Beweis
Wir rechnen

$$\begin{aligned}
\|\hat{x}\|^2 &= \sum_{k=0}^{N-1} \hat{x}_k \bar{\hat{x}}_k \\
&= \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} w^{k\ell} x_l \cdot \sum_{m=0}^{N-1} \bar{w}^{km} \bar{x}_m \\
&= \sum_{\ell} \sum_{m} x_\ell \bar{x}_m \underbrace{\sum_{k} w^{k\ell} \bar{w}^{km}}_{=N\cdot\delta_{lm}} \\
&= N \cdot \sum_{\ell=0}^{N-1} x_\ell \bar{x}_\ell \\
&= N \cdot \|x\|^2
\end{aligned}$$

Womit die Gleichheit gilt. □

1.6 Notation
Wir haben die Abbildung $\mathcal{F}_N : \mathbb{C}^N \to \mathbb{C}^N$ linear mit der Matrix $(w_k^{kj})_{k,j=0}^{N-1}$. Analog dazu führen wir die Abbildung ein:
$\bar{\mathcal{F}}_N : \mathbb{C}^N \to \mathbb{C}^N$ mit der Matrix $(\bar{w}_N^{kj})_{k,j=0}^{N-1} = (w_N^{-kj})$

1.7 Satz (inverse diskrete Fouriertransformation)
Es ist

$$\mathcal{F}_N^{-1} = \frac{1}{N}\bar{\mathcal{F}}_N$$

bzw.

$$x_j = \frac{1}{N}\sum_{k=0}^{N-1} \bar{w}_N^{kj}\hat{x}_k$$

für $j = 0, \ldots, N-1$

Beweis
Wir wissen aus dem Hilfssatz, daß $\mathcal{F}_N \cdot \bar{\mathcal{F}}_N = N \cdot I_N$ oder direkt:

$$\sum_{k=0}^{N-1} w_N^{-kj} \cdot \sum_{\ell=0}^{N-1} w_N^{k\ell} x_\ell = \sum_{\ell=0}^{N-1} x_\ell \sum_{k=0}^{N-1} w_N^{-kj} w_N^{k\ell} = N x_j$$

□

1.8 Definition (punktweises Produkt)
Wir betrachten nun die Multiplikation von Folgen $x, y \in \mathbb{C}^N$ als punktweise Multiplikation:

$$(x \cdot y)_k = x_k y_k$$

1.9 Definition (Faltungs-Multiplikation)
Wir definieren für Folgen $x, y \in \mathbb{C}^N$, die N-periodisch fortgesetzt sind, die sog. Faltungsmultiplikation $x * y \in \mathbb{C}^N$:

$$(x * y)_k = \sum_{j=0}^{N-1} x_{k-j} y_j$$

1.10 Satz (Faltungssatz)
Die Fouriertransformation überführt die Faltung in ein punktweises Produkt, d.h.

$$\mathcal{F}_N(x * y) = (\mathcal{F}_N x) \cdot (\mathcal{F}_N y)$$

Beweis
Wir betrachten die m-te Komponente der linken Seite:

$$(\mathcal{F}_N(x * y))_m = \sum_{k=0}^{N-1} w^{mk} \sum_{j=0}^{k-1} x_{k-j} y_j$$

Für eine Indexverschiebung: $k - j =: \ell$ erhalten wir:

$$= \sum_{j=0}^{N-1} \sum_{\ell=-j}^{N-1-j} w^{m(\ell+j)} x_\ell y_j$$

da die Folgen N-periodisch sind, erhalten wir:

$$\begin{aligned} &= \sum_{j=0}^{N-1} \sum_{\ell=0}^{N-1} w^{m(\ell+j)} x_\ell y_j \\ &= \sum_{j=0}^{N-1} w^{mj} y_j \sum_{\ell=0}^{N-1} w^{m\ell} x_\ell \\ &= \hat{y}_m \hat{x}_m \\ &= (\mathcal{F}_N y) \cdot (\mathcal{F}_N x) \end{aligned}$$

womit die Behauptung folgt. □

1.11 Korollar (Rechenregeln für die Faltung)
Da das punktweise Produkt kommutativ und assoziativ ist, ist die Faltung ebenfals kommutativ und assoziativ.

1.12 Korollar (direkte Berechnung der Faltung)
Es ist

$$x * y = \frac{1}{N} \bar{\mathcal{F}}_N \left(\mathcal{F}_N x \cdot \mathcal{F}_N y \right)$$

was uns eine geschickte Berechnung für die Faltung ermöglicht, wenn man die Fouriertransformation bereits kennt.

1.13 Bemerkung (Betrachtung des Rechenaufwands für die Faltung)
Eine direkte Berechnung der Faltung braucht etwa N^2 Multiplikationen und Additionen.

Mit der FFT benötigen wir je $N \cdot \log_2 N$ Operationen, für die punktweise Multiplikation benötigt man lediglich N Operationen. Für die inverse FFT benötigen wir ebenfalls $N \cdot \log_2 N$ Operationen.

Damit benötigt man mit der FFT lediglich $3N \log_2 N + 2N$ Operationen.

1.2 Schnelle Fouriertransformation (FFT)

Wir haben einen Vektor $x = (x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$ gegeben und möchten $\hat{x} = \mathcal{F}_N x$ berechnen.

1.14 Satz (Reduktionsformel)
Wir teilen den Vektor x auf in zwei Vektoren in u und v, wobei u die geraden und v die ungeraden Indizes sind, also

$$x = (u_0, v_0, u_1, v_1, \ldots, u_{N/2-1}, v_{N/2-1}) \in \mathbb{C}^N$$

Dann gilt für $k = 0, \ldots, N/2 - 1$:

$$\begin{aligned}(\mathcal{F}_N x)_k &= (\mathcal{F}_{N/2} u)_k + w_N^k (\mathcal{F}_{N/2} v)_k \\ (\mathcal{F}_N x)_{k+N/2} &= (\mathcal{F}_{N/2} u)_k - w_N^k (\mathcal{F}_{N/2} v)_k\end{aligned}$$

Beweis
Wir berechnen den k-ten Eintrag von $\mathcal{F}_N x$:

$$\begin{aligned}(\mathcal{F}_N x)_k &= \sum_{\ell=0}^{N-1} w_N^{k\ell} x_\ell \\ &= \sum_{j=0}^{N/2-1} w_N^{k \cdot 2j} u_j + \sum_{j=0}^{N/2-1} w_N^{k(2j+1)} v_j\end{aligned}$$

Hierbei ist zu beachten, daß $W_N^2 = W_{N/2}$ und damit ergibt sich:

$$\begin{aligned}&= \sum_{j=0}^{N/2-1} w_{N/2}^{kj} u_j + w_N^k \sum_{j=0}^{N/2-1} w_{N/2}^{kj} v_j \\ &= (\mathcal{F}_{N/2} u)_k + w_N^k (\mathcal{F}_{N/2} v)_k\end{aligned}$$

Wegen der Periodizität gilt dies für alle k und damit folgt aus der Bedingung

$$w_N^{k+N/2} = w_N^k \cdot w_N^{N/2} = -w_N^k$$

die zweite Gleichung. □

1.15 Bemerkung
Falls $\mathcal{F}_{N/2} u$ und $\mathcal{F}_{N/2} v$ bekannt sind, braucht man $\frac{N}{2}$ Multiplikationen und N Additionen zur Berechnung von $\mathcal{F}_N x$.

1.16 Historische Notiz
Diese Formel geht auf Cooley-Tukey, 1965 zurück.

Ähnliche Ideen fand man ebenfalls bei Danilson und Lanzos, 1942, Runge 1925, Gauß und Caesar „divide et impere“.

1.17 Bemerkung (zum Algorithmus)
Falls $N = 2^L$ ist, so kann man den Vektor L-mal zerlegen, bis man Vektoren der Länge 1 erhält.

Dabei erhält man einen Rechenaufwand von $L \cdot \frac{N}{2}$ Multiplikationen. Dabei ist $L = \log_2 N$.

1.18 Satz (Rechenaufwand der FFT)
$\mathcal{F}_N x$ läßt sich für $N = 2^L$ berechnen mit $\frac{1}{2} N \log_2 N$ komplexen Multiplikationen und $N \log_2 N$ komplexen Additionen.

1.19 Bemerkung (Reihenfolge der Elemente)
Die Reihenfolge der Elemente wird bei der Durchführung des Algorithmus vertauscht.

Wir betrachten nun die Binärdarstellung der Elemente, um einen Trick zu erhalten, dies zu sehen:

Input dezimal	Input binär	Output binär	Output dezimal
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Wir erhalten also die Reihenfolge durch Spiegeln der Binärzahlen.

1.20 Bemerkung (Vergleich zur direkten Berechnung)
Wir halten nun verschiedene konkrete Werte, die für den Vorteil der FFT sprechen, in einer Tabelle fest:

N	N^2	$N \log N$	Quotient
$2^5 = 32$	10^3	160	6,4
$2^{10} \approx 10^3$	10^6	10^4	100
$2^{20} \approx 10^6$	10^{12}	$2 \cdot 10^7$	50.000

1.3 Fourier-Reihen

1.21 Definition (Fouriertransformierte)
Sei $(c_n)_{n\in\mathbb{Z}}$ eine absolut summierbare Folge komplexer Zahlen, d.h. $\sum_{n\in\mathbb{Z}} |c_n| < \infty$.

$$\hat{c}(t) = \sum_{n=-\infty}^{\infty} c_n e^{\imath nt} \quad t \in \mathbb{R}$$

heiße die *Fouriertransformierte* der Folge $(c_n)_{n\in\mathbb{Z}}$.

1.22 Proposition (Eigenschaften der Fouriertransformierten)
Die Fouriertransformierte $\hat{c}(t)$ hat die folgenden Eigenschaften:

1. $\hat{c}$ ist 2π-periodisch.
2. $\hat{c}$ ist stetig.
3. Es gilt die Orthogonalitätsrelation
$$\frac{1}{2\pi}\int_0^{2\pi} e^{-\imath nt} e^{\imath mt}\, dt = \begin{cases} 1, & m = n \\ 0, & m \neq n \end{cases}$$
4. Es gilt die Umkehrformel:
$$c_n = \frac{1}{2\pi}\int_0^{2\pi} e^{-\imath nt}\hat{c}(t)\, dt$$
5. Es gilt ebenfalls wie bei der diskreten Fouriertransformation die Parseval-Gleichung:
$$\sum_{n=-\infty}^{\infty} |c_n|^2 = \frac{1}{2\pi}\int_0^{2\pi} |\hat{c}(t)|^2\, dt$$
6. Seien $(c_n), (d_n)$ absolut summierbar. Dann gilt für die Faltung:
$$(c * d)_n := \sum_{j=-\infty}^{\infty} c_{n-j} d_j$$
als Faltung definiert und es gilt der Faltungssatz:
$$\widehat{(c * d)}(t) = \hat{c}(t) \cdot \hat{d}(t)$$

Beweis

2. $\sum_{n=-N}^{N} c_n e^{\imath nt} \to \hat{c}(t)$ ist gleichmäßig konvergent für $N \to \infty$ wegen
$$\left| \hat{c}(t) - \sum_{n=-N}^{N} c_n e^{\imath nt} \right| \leq \sum_{|n|>N} |c_n| \to 0$$
und diese Konvergenz ist unabhängig von t und damit ist $\hat{c}$ stetig. □

1.23 Definition (Fourierkoeffizient)
Sei f eine 2π-periodische stetige Funktion gegeben. Dann ist für $n \in \mathbb{Z}$

$$c_n = \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-\imath n t} \, dt$$

der n-te Fourier-Koeffizient von f und wir bezeichnen $c_n = \hat{f}(n)$.

1.24 Satz (Berechnung der Fourierkoeffizenten und Abschätzung)
Sei f 2π-periodisch und p-mal diffbar, $f^{(p)}$ absolut integrierbar (es würde genügen, wenn $f \in W^{p,1}$ ist). Dann gilt:

1. Der n-te Fourier-Koeffizient von $f^{(p)}$ ist $(\imath n)^p \cdot c_n$.
2. $c_n = \mathcal{O}(|n|^{-p})$, d.h. $|c_n| \leq M \cdot |n|^{-p}$ mit
$$M = \frac{1}{2\pi} \int_0^{2\pi} |f^{(p)}(t)| \, dt < \infty$$

Beweis

1. Wir erhalten über (mehrfache) partielle Integration:
$$\begin{aligned} \frac{1}{2\pi} \int_0^{2\pi} f^{(p)}(t) e^{-\imath n t} \, dt &= -\frac{1}{2\pi} \int_0^{2\pi} f^{(p-1)}(t)(-\imath n) e^{-\imath n t} \, dt \\ &= (\imath n)^p \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-\imath n t} \, dt \end{aligned}$$
wobei die Randterme aufgrund der Periodizität wegfallen.
2. Durch den obigen Beweis erhalten wir mit Betragsbetrachtungen:
$$|c_n| \cdot |n|^p = \left| \frac{1}{2\pi} \int_0^{2\pi} f^{(p)}(t) e^{-\imath n t} \, dt \right| \leq \frac{1}{2\pi} \int_0^{2\pi} |f^{(p)}(t)| \, dt$$

□

1.25 Bemerkung
Insbesondere gilt, daß (c_n) absolut summierbar ist, falls $f \in \mathcal{C}^2$ (oder auch $f \in W^{2,1}$).

Mit größeren Aufwand kann man auch sagen, daß $f \in \mathcal{C}^1$ genügen würde.

1.26 Satz (Faltungssatz)
Seien f, g stetig und 2π-periodisch. Dann ist die Faltung von f und g, definiert durch

$$(f * g)(t) = \frac{1}{2\pi} \int_0^{2\pi} f(t-\tau) g(\tau)\, d\tau$$

wieder 2π-periodisch und stetig.

Für die Fourierkoeffizienten der Faltung gilt:

$$\widehat{f * g}(n) = \hat{f}(n) \cdot \hat{g}(n), \quad n \in \mathbb{Z}$$

Beweis
Die Periodizität ist klar und die Stetigkeit sind klar nach Analysis II.

Wir rechnen nun

$$\begin{aligned}\widehat{(f * g)}(n) &= \frac{1}{2\pi} \int_0^{2\pi} \frac{1}{2\pi} \int_0^{2\pi} f(t-\tau) g(\tau)\, d\tau \cdot e^{-\imath n t}\, dt \\ &= \left(\frac{1}{2\pi}\right)^2 \int_0^{2\pi} \int_0^{2\pi} f(t-\tau) e^{-\imath n (t-\tau)} \cdot g(\tau) e^{-\imath n \tau}\, dt\, d\tau\end{aligned}$$

für $s := t - \tau$:

$$\begin{aligned}&= \frac{1}{2\pi} \int_0^{2\pi} f(s) e^{-\imath n s}\, ds \cdot \frac{1}{2\pi} \int_0^{2\pi} g(\tau) e^{-\imath n \tau}\, d\tau \\ &= \hat{f}(n) \cdot \hat{g}(n)\end{aligned}$$

womit die Behauptung gilt. □

1.27 Bemerkung (Verallgemeinerung)
Die Fouriertransformation ist auch für $n \in \mathbb{R}$ definiert und hierfür gilt ebenfalls der Faltungssatz.

1.28 Bemerkung (Ausblick)
Läßt sich eine stetige Funktion f aus deren Fourierkoeffizienten zurückgewinnen? Gilt also $f(t) = \sum_{n=-\infty}^{\infty} c_n e^{\imath n t}$?

Wir werden sehen, daß ohne zusätzliche Voraussetzungen an f die Folge (c_n) nicht absolut summierbar ist. Selbst wenn (c_n) absolut summierbar ist, gilt dann $f(t) = \hat{c}(t)$? Wir werden im Folgenden sehen, daß dies richtig ist.

1.29 Satz (Fejer, 1904)
Sei $f : \mathbb{R} \to \mathbb{C}$ stetig und 2π-periodisch mit Fourierkoeffizienten

$$(c_n)_{n\in\mathbb{Z}} = \frac{1}{2\pi}\int_0^{2\pi} e^{-\imath nt} f(t)\,dt$$

Dann gilt:

$$\sum_{k=-n}^{n}\left(1-\frac{|k|}{n+1}\right)c_k e^{\imath kt} \to f(t)$$

gleichmäßig in t.

Beweis

1. Wir betrachten die Faltung mit dem Faltungssatz

   $$(e^{\imath n\tau} * f)(t) = \frac{1}{2\pi}\int_0^{2\pi} e^{\imath n(t-\tau)} f(\tau)\,d\tau = e^{\imath nt} c_n$$

   Damit gilt wegen Linearität

   $$\sum_{j=-n}^{n}\left(1-\frac{|j|}{n+1}\right)e^{\imath jt}c_j := (K_n * f)(t)$$

   mit dem sog. Fejer-Kern K_n, für den gilt:

   $$K_n(t) = \sum_{j=-n}^{n}\left(1-\frac{|j|}{n+1}\right)e^{\imath jt}$$

2. Wir zeigen, daß die Reduktionsformel gilt:

   $$K_n(t) = \frac{1}{n+1}\left(\frac{\sin\left(\frac{n+1}{2}t\right)}{\sin\left(\frac{t}{2}\right)}\right)^2$$

   Dies zeigen wir über die Definition des Sinus, indem wir verwenden, daß gilt:

   $$\sin^2\left(\frac{t}{2}\right) = \frac{1}{2}(1-\cos t) = -\frac{1}{4}e^{-\imath t} + \frac{1}{2} - \frac{1}{4}e^{\imath t}$$

   Durch direkte Rechnung ergibt sich dann die Identität, denn es gilt:

   $$\begin{aligned}\left(-\frac{1}{4}e^{\imath t} + \frac{1}{2} - \frac{1}{4}e^{\imath t}\right)\sum_{j=-n}^{n}\left(1-\frac{|j|}{n+1}\right)e^{\imath jt} &= \frac{1}{n+1}\left(-\frac{1}{4}e^{-\imath(n+t)t} + \frac{1}{2} - \frac{1}{4}e^{\imath(n+1)t}\right)\\ &= \frac{1}{n+1}\sin^2\left(\frac{(n+1)t}{2}\right)\end{aligned}$$

3. Wir betrachten nun einige Eigenschaften:

   a) Es ist wegen der Orthogonalitätsrelation:

$$\frac{1}{2\pi}\int_0^{2\pi} K_n(t)\,dt = \sum_{j=-n}^{n}\left(1-\frac{|j|}{n+1}\right)\underbrace{\frac{1}{2\pi}\int_0^{2\pi} e^{\imath jt}\,dt}_{=\delta_{j0}} = 1$$

   b) $K_n(t) \geq 0 \quad \forall t\,\forall n$ wegen der Rekuktionsformel.

   c) $\forall \delta \in (0,\pi)$ ist

$$\lim_{n\to\infty}\int_\delta^{2\pi-\delta} K_n(t)\,dt = 0$$

   wegen

$$K_n(t) \leq \frac{1}{n+1}\frac{1}{\left(\sin\left(\frac{\delta}{2}\right)\right)^2} \to 0$$

4. Es ist:

$$(K_n * f)(t) - f(t) = \frac{1}{2\pi}\int_0^{2\pi} K_n(\tau)\big(f(t-\tau)-f(t)\big)\,d\tau$$

   und damit:

$$\int_0^{2\pi} K_n(\tau)\big(f(t-\tau)-f(t)\big)\,d\tau = \underbrace{\int_{-\delta}^{\delta} K_n(\tau)\big(f(t-\tau)-f(t)\big)\,d\tau}_{=:I_1} + \underbrace{\int_\delta^{2\pi-\delta} K_n(\tau)\big(f(t-\tau)-f(t)\big)\,d\tau}_{=:I_2}$$

   und damit

$$I_1 \leq \max_{|\tau|\leq\delta}\big|f(t-\tau)-f(t)\big| \cdot \frac{1}{2\pi}\int_{-\delta}^{\delta} K_n(\tau)\,d\tau \leq \frac{1}{2\pi}\int_{-\pi}^{\pi} K_n(\tau)\,d\tau = 1$$

   Den anderen Teil schätzen wir wie folgt ab:

$$I_2 \leq \frac{1}{2\pi}\cdot 2\max_{0\leq\theta\leq 2\pi}|f(\theta)| \cdot \underbrace{\int_\delta^{2\pi-\delta} K_n(\tau)\,d\tau}_{\to 0}$$

Damit gilt für $n \to \infty$:

$$\limsup_{n\to\infty} |(K_n * f)(t) - f(t)| \leq \max_{|\tau|\leq\delta} |f(t-\tau) - f(t)| \quad \forall \delta \in (0, \pi)$$

und für $\delta \to 0$ geht der Term gegen 0 wegen der Steitigkeit. Weil f gleichmäßig stetig ist, konvergiert dies auch gleichmäßig.

Also konvergiert $K_n * f(t) \to f(t)$ gleichmäßig in t. □

1.30 Bemerkung
Es sei festzuhalten, daß

$$\sum_{k=-n}^{n} \left(1 - \frac{|k|}{n+1}\right) c_k e^{\imath nt} = \frac{1}{n+1} \sum_{m=0}^{n} \left(\sum_{k=-m}^{m} c_k e^{\imath kt} \right)$$

das arithmetische Mittel aller Partialsummen ist, wobei die Partialsumme nicht unbedingt konvergieren muß.

1.31 Satz (Eindeutigkeitssatz)
Seien f, g 2π-periodisch, stetig mit denselben Fourier-Koeffizienten. Dann ist $f = g$.

Beweis
Es ist

$$f(t) = \lim_{n\to\infty} \sum_{j=-n}^{n} \left(1 - \frac{|j|}{n+1}\right) c_j e^{\imath jt} = g(t) \quad \forall t$$

□

1.32 Satz (Darstellung mittels Fourierkoeffienzten)
Sei f 2π-periodisch und stetig. Falls die Fourier-Koeffizienten absolut summierbar sind, dann gilt:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{\imath nt} \quad \forall t$$

Beweis
f und die Reihe haben dieselben Fourierkoeffizenten und damit stimmen nach dem Eindeutigkeitssatz beide Funktionen überein. □

1.33 Bemerkung
Dies ist insbesondere erfüllt, wenn f einmal stetig diff'bar ist ($f \in W^{1,1}$ würde auch genügen).

1.34 Satz (Interpretation des Satzes von Fejer)
Jede stetige 2π-periodische Funktion kann gleichmäßig durch trigonometrische Polynome approximiert werden.

BEWEIS
Dies ergibt sich direkt aus der Aussage des Satzes von Fejer. □

1.35 Satz (Weierstraßscher Approximationssatz)
Jede stetige Funktion auf einem kompakten Intervall $g : [a, b] \to \mathbb{R}$ kann gleichmäßig durch Polynome approximiert werden, d.h.

$$\forall \varepsilon > 0 \, \exists \text{ Polynom } p : \quad \max_{x \in [a,b]} |p(x) - g(x)| < \varepsilon$$

BEWEIS
Wir nehmen o.B.d.A. an, daß $[a, b] = [-1, 1]$ und setzen $f(t) = g(x)$ für $x = \cos t$, genauer $t = \arccos x \in [0, \pi]$.

Weiter setzen wir f als gerade Funktion 2π-periodisch fort, also $f(-t) = f(t)$. f ist stetig.

Wir sehen, daß bei geraden Funktionen $c_{-n} = c_n$ ist, denn:

$$c_{-n} = \frac{1}{2\pi} \int_0^{2\pi} e^{\imath n t} f(t)\, dt = \frac{1}{2\pi} \int_{-2\pi}^{0} e^{-\imath n \tau} \underbrace{f(-\tau)}_{=f(\tau)}\, d\tau = \frac{1}{2\pi} \int_0^{2\pi} e^{-\imath n \tau} \underbrace{f(-\tau)}_{=f(\tau)}\, d\tau = c_n$$

aufgrund von Variablentransformationen und der 2π-Periodizität. Weiter wissen wir mit dem Satz von Fejer:

$$\sum_{j=-n}^{n} \left(1 - \frac{|j|}{n+1}\right) c_j e^{\imath j t} = c_0 + 2 \sum_{k=1}^{n} \left(1 - \frac{k}{n+1}\right) c_k \cos(kt)$$

Wir wissen, daß der linke Term gleichmäßig gegen $f(t)$ konvergiert.
Weil $\cos(k \arccos x) = T_k(x)$ das k-te Tschebyscheff-Polynom ist, gilt damit:

$$c_0 + 2 \sum_{k=1}^{n} \left(1 - \frac{k}{n+1}\right) c_k T_k(x) \to g(x)$$

Falls (c_k) absolut summierbar ist, dann ist dies gerade gleichmäßig konvergent. □

1.36 Bemerkung
Die Konvergenz im letzten Satz kann beliebig langsam sein. Sie wird schneller, wenn die Koeffizienten rasch abfallen (dies ist inbesondere der Fall, wenn die Funktion oft diff'bar ist).

1.4 Approximation von Fourierkoeffizienten, trigonometrische Interpolation

1.37 Bemerkung
Wir möchten den Zusammenhang zwischen der diskreten und kontinuierlichen Fouriertransformation klären:

Wir haben die Fourierkoeffizienten für eine 2π-periodische, stetige Funktion f:

$$\hat{f}(n) = c_n = \frac{1}{2\pi} \int_0^{2\pi} e^{-\imath nt} f(t)\, dt$$

Wir approximieren das Integral durch die Trapezregel mit $h = \frac{2\pi}{N}$ und erhalten:

$$\hat{f}_N(n) = \frac{1}{2\pi} \left(\frac{h}{2} e^{-\imath n0} f(0) + h e^{-\imath nh} f(h) + \ldots + h e^{-\imath n(N-1)h} f((N-1)h) + \frac{h}{2} e^{-\imath nNh} f(Nh) \right)$$

Wegen der Periodizität ist $\frac{h}{2} e^{-\imath n0} f(0) = \frac{h}{2} e^{-\imath Nh} f(Nh)$ (Also fällt die Trapezregel hier mit der Rechtecksregel zusammen). Damit erhalten wir insgesamt:

$$\hat{f}_N(n) = \frac{1}{N} \sum_{j=0}^{N-1} \underbrace{e^{-\imath nj \frac{2\pi}{N}}}_{= w_N^{-nj}} f(t_j) = \frac{1}{N} \sum_{j=0}^{N-1} w_N^{-nj} f(t_j)$$

wobei $t_j = jh = j\frac{2\pi}{N}$. Dieses $\hat{f}_N(n)$ ist damit N-periodisch als Vektor.

Damit ist

$$\hat{f}_N = \mathcal{F}_N^{-1}(f(t_j))_{j=0}^{N-1}$$

was wir mit FFT berechnen können.

1.38 Satz (Aliasing-Formel)
Sei $(\hat{f}(n))_{n\in\mathbb{Z}}$ absolut summierbar. Dann ist

$$\hat{f}_N(n) - \hat{f}(n) = \sum_{0 \neq \ell \in \mathbb{Z}} \hat{f}(n + \ell N)$$

Beweis
Wir wissen aus dem letzten Kapitel, daß bei der absoluten Summierbarkeit der Fourierkoeffizienten gilt:

$$f(t) = \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{\imath k t}$$

Wir betrachen nun:

$$\begin{aligned}
\hat{f}_N(n) &= \frac{1}{N} \sum_{j=0}^{N-1} w_N^{-nj} \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{\imath k \cdot j \frac{2\pi}{N}} \\
&= \sum_{k=-\infty}^{\infty} \hat{f}(k) \underbrace{\frac{1}{N} \sum_{j=0}^{N-1} w_N^{-nj} w_N^{kj}}_{= \begin{cases} 1, & k = n \mod N \\ 0, & \text{sonst} \end{cases}} \\
&= \sum_{\ell=-\infty}^{\infty} \hat{f}(n + \ell N)
\end{aligned}$$

Für $\ell = 0$ ist dies gerade $\hat{f}(n)$ und damit ergibt sich die Behauptung. □

1.39 Korollar (Fehlerabschätzung für den numerischen Fourierkoeffizienten)
Sei f p-fach stetig diff'bar (mit $p \geq 2$) und 2π-periodisch.
Dann gilt für $|n| \leq \frac{N}{2}$:

$$\left| \hat{f}_N(n) - \hat{f}(n) \right| \leq C \cdot N^{-p}$$

mit

$$C = \frac{1}{2\pi} \int_0^{2\pi} |f^{(p)}(t)| \, dt$$

Beweis
Wir wissen aus dem vorherigen Abschnitt, daß

$$|\hat{f}(n)| \leq C |n|^{-p}$$

andererseits ist mit der Aliasing-Formel:

$$\left| \hat{f}_N(n) - \hat{f}(n) \right| \leq \sum_{\ell \neq 0} |\hat{f}(n + \ell N)| \leq \sum_{\ell \neq 0} C \cdot |n + \ell N|^{-p}$$

Es ist $|n + \ell N| \geq \frac{N}{2}$ für $|n| \leq \frac{N}{2}$.
Also ist $|n + \ell N| < N$ für höchstens ein $\ell \neq 0$. Dies ist:

$$\ell = \begin{cases} -1, & n > 0 \\ 1, & n < 0 \end{cases}$$

Zu jedem Intervall $I = [N, 2N), [2N, 3N), \ldots$ und $(-2N, -N], (-3N, -2N], \ldots$ gibt es genau ein ℓ so, daß $n + \ell N \in I$. Damit ergibt sich insgesamt:

$$\sum_{\ell \neq 0} |n + \ell N|^{-p} \leq \left(\frac{N}{2}\right)^{-p} + 2 \sum_{m=1}^{\infty} (mN)^{-p} \leq c_p \cdot N^{-p}$$

mit der Konstante c_p, der sich aus den konvergierenden Summen ergibt. Damit ist $C = c_p M$ und die Behauptung folgt. □

1.40 Bemerkung
Für den Spezialfall $n = 0$ ergibt sich für die Gitterweite $h = \frac{2\pi}{N}$:

$$h \sum_{j=0}^{N-1} f(t_j) - \frac{1}{2\pi} \int_0^{2\pi} f(t)\, dt = \mathcal{O}(h^p)$$

Die Ordnung der Trapezregel hängt also von der Glattheit der Funktion ab, d.h. die Trapezregel ist sehr genau für glatte und periodische Integranden.

1.41 Satz (trigonometrische Interpolation)
Das trigonometrische Polynom

$$f_N(t) = {\sum_{n=-\frac{N}{2}}^{\frac{N}{2}}}' \hat{f}_N(n) e^{\imath n t}$$

interpoliert f in den Punkten $t_j = j\frac{2\pi}{N}$ mit $j = 0, 1, \ldots, N-1$.

Beweis
Wir wissen, daß wegen der Periodizität gilt (insbesondere ist $e^{\imath n t_j} = w_N^{nj}$):

$$f_N(t_j) = \sum_{n=0}^{N-1} \hat{f}_N(n) w_N^{nj}$$

Dabei haben wir insbesondere verwendet, daß der linke und rechte Randterm gleich sind, wodurch die $\sum'$ verschwindet.

Diese Formel bedeutet, daß

$$(f_N(t_j))_{j=0}^{N-1} = \mathcal{F}_N \left(\hat{f}_N(n)\right)_{n=0}^{N-1} = \mathcal{F}_N \mathcal{F}_N^{-1}(f(t_j))_{j=0}^{N-1} = (f(t_j))_{j=0}^{N-1}$$

Damit interpoliert $f_N(t)$ bereits die Gitterpunkte t_j. □

1.42 Notation
$\sum'$ bedeutet, daß der erste Term und der letzte Term jeweils mit dem Faktor $\frac{1}{2}$ genommen werden, d.h. für unsere Summe ergibt sich gerade:

$$\sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' \hat{f}_N(n) := \frac{1}{2}\hat{f}_N\left(-\frac{N}{2}\right) + \hat{f}_N\left(-\frac{N}{2}+1\right) + \ldots + \hat{f}_N\left(\frac{N}{2}-1\right) + \frac{1}{2}\hat{f}_N\left(\frac{N}{2}\right)$$

Dies ist die Interpolation in den Knoten $t_j = j\frac{2\pi}{N}$.

Es wäre ebenso erlaubt, stattdessen eine der folgenden Summen zu bilden:

$$\sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}}, \qquad \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1}, \qquad \sum_{n=-\frac{N}{2}+k}^{\frac{N}{2}-1+k}$$

und mit dieser Methode könnten wir ebenso den Satz beweisen.

Wir werden jedoch sehen, daß unsere Variante aus Symmetriegründen die am wenigsten oszillierenste ist.

1.43 Algorithmus (trigonometrische Interpolation)
Wir nehmen f als 2π-periodisch. Dann gehen wir wie folgt vor:

1. Wir berechnen $f(t_j)$ für $j = 0, \ldots, N-1$ und $t_j = j \cdot \frac{2\pi}{N}$.
2. Wir berechnen mit der FFT $\hat{f}_N = \mathcal{F}_N^{-1}(f(t_j))_{j=0}^{N-1}$ mit einem Aufwand von $\mathcal{O}(N \log N)$ Operationen.
3. Wir erhalten das trigonometrische IPP wie im vorhergehenden Satz.

1.44 Satz (Fehlerabschätzung bei der trigonometrischen Interpolation)
Falls $(\hat{f}(n))_n$ absolut summierbar ist, gilt:

$$|f_N(t) - f(t)| \leq 2 \cdot \sum_{|n| \geq \frac{N}{2}}{}' \left|\hat{f}(n)\right|$$

Beweis
Wir verwenden die Aliasing-Formel und erhalten:

$$\begin{aligned}|f_N(t) - f(t)| &= \left| \sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' \hat{f}_N(n) e^{\imath nt} - \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{\imath nt} \right| \\ &= \left| \sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' (\hat{f}_N(n) - \hat{f}(n)) e^{\imath nt} - \sum_{|n|\geq\frac{N}{2}}{}' \hat{f}(n) e^{\imath nt} \right|\end{aligned}$$

mit der Aliasing-Formel:

$$\begin{aligned}&= \left| \sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' \left(\sum_{\ell\neq 0} \hat{f}(n+\ell N) \right) e^{\imath nt} - \sum_{|n|\geq\frac{N}{2}}{}' \hat{f}(n) e^{\imath nt} \right| \\ &\leq \sum_{|m|\geq\frac{N}{2}}{}' |\hat{f}(m)| + \sum_{|n|\geq\frac{N}{2}}{}' |\hat{f}(n)| \\ &\leq 2 \cdot \sum_{|n|\geq\frac{N}{2}}{}' |\hat{f}(n)|\end{aligned}$$

womit der Satz folgt. □

1.45 Bemerkung
Falls $f \in \mathcal{C}^p$ mit $p \geq 2$. Dann gilt bekanntlich, daß $\hat{f}(n) = \mathcal{O}(|n|^{-p})$.
Also ist $|f_N(t) - f(t)| = \mathcal{O}(N^{-p})$

1.5 Inverses Faltungsproblem, Regularisierung, Filter

1.46 Motivation (Problemstellung)
Ein Eingangssignal $u = u(x)$ für $x \in \mathbb{R}$ geht in einen Apparat und es wird ein Signal b gemessen, was nicht mehr dem Eingangssignal entspricht.

Wir treffen die folgende Annahme:

1. $u \mapsto b$ sei linear.
2. $u \mapsto b$ sei verschiebungsinvariant

Wir wissen, daß jene Abbildungen Faltungen sind.

1.47 Bemerkung
Diese Problemstellung tritt in vielen Bereichen, etwa der Bild- bzw. Signalverarbeitung auf.

1.48 Konstruktion (Modell des Apparats)
Wir betrachten das folgende Modell:

$$\int_{-\infty}^{\infty} a(x-y)u(y)\,dy + \varepsilon(x) = b(x)$$

wobei $u(y)$ das unbekannte Eingangssignal ist, $b(x)$ das beobachtete Signal und der Term $a(x-y)$ sei die Apparatfunktion, während der Term $\varepsilon(x)$ die Störungen des Modells beschreibt.
In $\varepsilon(x)$ gehen Meßfehler, Rundungsfehler und Modellfehler ein. Dieses Rauschen ist natürlich unbekannt. Wir kennen jedoch evtl. eine Schranke für die Störungen so, daß $|\varepsilon(x)| \leq M$ punktweise oder im quadratischen Mittel, d.h. $\int |\varepsilon(x)|^2 \leq M$ oder ähnlich, ist.

Wir möchten aus dem beobachteten Signal das Eingangssignal u rekonstruieren.

1.49 Bemerkung (Reduktion des Problems)
Im Allgemeinen haben a, u, b einen kompakten Träger so, daß $\operatorname{supp} a = \overline{\{x : a(x) \neq 0\}}$.
Nach Variablentransformation kann man ohne Einschränkungen erhalten, daß

$$\operatorname{supp} a \subseteq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \qquad \operatorname{supp} u \subseteq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

Damit ist

$$\operatorname{supp}(b-\varepsilon) \subseteq [-\pi, \pi]$$

denn falls $b(x) \neq \varepsilon(x)$. Dann $\exists y : a(x-y) \cdot u(y) \neq 0$ und dann ist $x-y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ und damit $x \in [-\pi, \pi]$.

Wir setzen nun außerdem a, u, ε, b 2π-periodisch auf $\mathbb{R}$ fort.

Damit haben wir unser Problem auf ein neues reduziert:

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} a(x-y)u(y)\,dy + \varepsilon(x) = b(x)$$

für $x \in [-\pi, \pi]$, wofür wir kurz schreiben können:

$$a * u + \varepsilon = b$$

bzw. für den linearen Operator $Au = a * u$:

$$Au + \varepsilon = b$$

1.50 Bemerkung
Der „übliche“ Lösungsweg besteht darin, daß z.B. das Lineare Gleichungssystem $Au + \varepsilon = b$ in $\mathbb{R}^n$ so gelöst wird, daß die Störung vernachlässigt wird und wir $Av = b$ mit den Standard-Methoden lösen.

Dann gilt für den Fehler $v - u = A^{-1}\varepsilon$, sofern $\|A^{-1}\varepsilon\| \ll \|v\|$, was nicht unbedingt gegeben sein muß, falls die Matrix schlecht konditioniert ist.
Leider ist dies hier der Fall.

1.51 Erinnerung (Faltungssatz)
Es ist $\widehat{a * u}(n) = \hat{a}(n)\hat{u}(n)$

1.52 Bemerkung (Schlechte Kondition des Problems)
Für die Fourierkoeffizienten gilt:

$$\hat{a}(n)\hat{u}(n) + \hat{\varepsilon}(n) = \hat{b}(n) \quad n \in \mathbb{Z}$$

Damit ist

$$\hat{u}(n) = \frac{\hat{b}(n)}{\hat{a}(n)} - \frac{\hat{\varepsilon}(n)}{\hat{a}(n)} = \hat{v}(n) - \frac{\hat{\varepsilon}(n)}{\hat{a}(n)}$$

Dabei löst $\hat{v}(n)$ die Gleichung $\hat{a}(n)\hat{u}(n) = \hat{b}(n)$.
Da aber die Fourierkoeffizienten von ε etwa konstant bleiben, aber die Fourierkoeffizeinten von a rasch abfallen (falls a glatt ist), wird der hintere Ausdruck den Term dominieren, wodurch sich dann ergibt, daß

$$u(x) = \sum_{n=-\infty}^{\infty} \hat{u}(n)e^{\imath nx} = \sum_{n=-\infty}^{\infty} \hat{v}(n)e^{\imath nx} - \sum_{n=-\infty}^{\infty} \frac{\hat{\varepsilon}(n)}{\hat{a}(n)} e^{\imath nx}$$

Wir erhalten für große n eine katastrophale Fehlerverstärkung, also ist das Problem tatsächlich schlecht konditioniert.
Man spricht auch von einem schlecht gestellten Problem oder einem sog. ill-posed problem.

1.53 Bemerkung (Anderer Zugang (Minimierungsproblem))
Wir möchten ja nicht $Au = b$ lösen, sondern wir möchten verlangen, daß $\|Au - b\| \leq \|\varepsilon\|$ wird, d.h. falls $\|\varepsilon\| \approx \delta$ bekannt ist, fordern wir, daß $\|Au - b\| \leq \delta$

Im Allgemeinen gibt es unendlich viele solche u, die dies erfüllen. Wir möchten unter all diesen das u wählen mit $\|u''\|$ minimal (wie etwa beim kubischen Spline).
Alternativ könnte man auch $\|u\|$ minimieren. Wir suchen also ein möglichst „glattes“ oder möglichst „kleines“ u und erhalten damit ein Minimierungsproblem.

1.54 Bemerkung (zur Norm)
Wir möchten festlegen, daß wir in diesem Fall, falls nicht anders gesagt, als Norm die L^2 Norm betrachten, also:

$$\|f\| = \|f\|_{L^2} := \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} |f(x)|^2 \, dx \right)^{\frac{1}{2}}$$

1.55 Satz (Allgemeine Form des Minimierungsproblem)
Wir möchten $\|Lu\|$ minimal erhalten und fordern nur, daß L linear ist, z.B. $Lu = u''$. Zusätzlich soll die Nebenbedingung $\|Au - b\| \leq \delta$ gelten.

Das Minimum wird angenommen für

$$\|Au - b\| = \delta$$

Beweis
Angenommen, dies gelte nicht, d.h. $\|Lu\|$ sei minimal für u mit $\|Au - b\| < \delta$.
Wir betrachten nun für ein $\varrho > 0$ mit $\tilde{u} = (1 - \varrho)u$. Weil L linear ist, gilt:

$$\|L\tilde{u}\| = (1 - \varrho)\|Lu\| < \|Lu\|$$

und für es gilt:

$$\|A\tilde{u} - b\| = \|(1 - \varrho)(Au - b) - \varrho b\| \leq \underbrace{(1 - \varrho)\|Au - b\|}_{<\delta} + \varrho\|b\| \leq \delta$$

womit für ein genügend kleines ϱ auf einen Widerspruch geführt ist. □

1.56 Bemerkung
In der Praxis hat man meist zusätzliche Einschränkungen, z.B. daß man nur Lösungen, die monoton oder positiv sind, berücksichtigt.

1.57 Bemerkung
Es ist eine sinnvolle Annahme, daß $\|b\| > \delta$, sonst wäre das beobachtete Signal schwächer als das Rauschen.
Daraus folgt direkt, daß $u \neq 0$ ist, wodurch das Problem nicht trivial gelöst werden kann.

1.58 Definition (Tychonoff-Regularisierung)
Wir betrachten ein festes $\alpha > 0$ als sog. Regularisierungsparameter.
Wir lösen dann das Minimierungsproblem ohne Nebenbedingung, d.h.

$$\|Au - b\|^2 + \alpha\|Lu\|^2 = \min$$

Für $\alpha \to 0$ ergibt sich, daß $\|Au - b\| = 0$. Damit wird aber $\|Lu\|$ beliebig groß.
Für $\alpha \to \infty$ ergibt sich, daß $\|Lu\| = 0$. Damit wird $\|Au - b\|$ beliebig groß.

Das optimale α ist so zu wählen so, daß $\|Au - b\| \approx \|\varepsilon\|$, falls dies bekannt ist, sonst muß man dies sehr lange ausprobieren, bis „das Bild schön wird“.

1.59 Bemerkung
Wir klären nun den Zusammenhang zwischen dem Minimierungs- und dem Regularisierungsproblem:

Wir beschränken uns auf ein endlich-dimensionales Problem, also $b \in \mathbb{R}^n$ und $u \in \mathbb{R}^n$. Dann sind A, L Matrizen und $\|\cdot\|$ sei die euklidische Norm. Dann ist

$$\|Lu\|^2 = u^T L^T L u, \qquad \|Au - b\|^2 = (Au-b)^T(Au-b) = u^T A^T A u - 2u^T A^T b + b^T b$$

Die allgemeine Lösung $f(u) = \min$ und $g(u) = 0$ erfüllt:

$$f'(u) + g'(u)^T \lambda = 0, \qquad g(u) = 0$$

wobei das λ ein Lagrange-Multiplikator ist. Hier erhalten wir dann die beiden Bedingungen:

$$2L^T L u + (2A^T A u - 2A^T b)^T \cdot \lambda = 0 \quad \lambda \in \mathbb{R}, \qquad \|Au - b\|^2 - \delta^2 = 0$$

also gilt: $(L^T L + \lambda A^T A)u = A^T b$. Für ein festes λ ist dies die Lösung eines Minimisierungsproblems ohne Nebenbedingungen:

$$\|Lu\|^2 + \lambda \|Au - b\|^2 = \min, \text{ also für } \alpha = \frac{1}{\lambda} \text{ gilt dann: } \|Au - b\|^2 + \alpha \|Lu\|^2 = \min$$

1.60 Satz (Tychonoff-Regularisierung)
Wir setzen $Au = a * u$ und nehmen $Lu = u^{(p)}$.
Die Lösung des Minimierungsproblems für $\alpha > 0$ als gegebenen Regularisierungsparameters

$$\|a * u - b\|_{L^2}^2 + \alpha \|u^{(p)}\|_{L^2}^2 = \min$$

für u eine 2π-periodisch mit quadratisch-integrierbarer p-ten Ableitung ist gegeben durch die Fourier-Koeffizienten

$$\hat{u}(n) = \begin{cases} \frac{|\hat{a}(n)|^2}{|\hat{a}(n)|^2 + \alpha n^{2p}} \cdot \frac{\hat{b}(n)}{\hat{a}(n)}, & \hat{a}(n) \neq 0 \\ 0, & \hat{a}(n) = 0 \end{cases}$$

Dabei definieren wir den Regularisierungsfilter $\Phi_\alpha(n)$:

$$\Phi_\alpha(n) := \frac{|\hat{a}(n)|^2}{|\hat{a}(n)|^2 + \alpha n^{2p}}$$

Beweis
Nach der Parseval-Formel gilt ist das Regularisierungsproblem äquivalent zu

$$\sum_{n=-\infty}^{\infty} \underbrace{\left(\left| \hat{a}(n)\hat{u}(n) - \hat{b}(n) \right|^2 + \alpha n^{2p} |\hat{u}(n)|^2 \right)}_{=:\Lambda_n} = \min$$

Dabei haben wir den Faltungssatz benutzt und die Tatsache, daß $\widehat{u^{(p)}}(n) = (\imath n)^p \hat{u}(n)$.
Der Ausdruck wird minimal, falls jeder einzelne Summand minimal wird.

Für den n-ten Summanden, Λ_n, rechnen wir:

$$\begin{aligned}\Lambda_n &= |\hat{a}(n)|^2 \cdot |\hat{u}(n)|^2 - \hat{a}(n)\hat{u}(n)\overline{\hat{b}(n)} - \overline{\hat{a}(n)\hat{u}(n)}\hat{b}(n) + |\hat{b}(n)|^2 + \alpha n^{2p}|\hat{u}(n)|^2 \\ &= \underbrace{|\hat{a}(n)|^2 + \alpha n^{2p}}_{=:r} \cdot \underbrace{|\hat{u}(n)|^2}_{=:z^2} - 2\,\mathrm{Re}\left(\underbrace{\overline{\hat{u}(n)}}_{=:\bar{z}} \underbrace{\overline{\hat{a}(n)} \cdot \hat{b}(n)}_{=:s} \right) + |\hat{b}(n)|^2\end{aligned}$$

Damit muß gelten:

$$r|z|^2 - 2\,\mathrm{Re}(\bar{z}s) = \min \text{ bzw. für } q = \frac{s}{r} \text{ gilt dann: } |z|^2 - 2\,\mathrm{Re}(\bar{z}q) = \min$$

aber es gilt wegen quadratischer Ergänzung:

$$|z|^2 - 2\,\mathrm{Re}(\bar{z}q) \geq |z|^2 - 2|z| \cdot |q| + |q|^2 - |q|^2 \geq -|q|^2$$

d.h.

$$\hat{u}(n) = \frac{|\hat{a}(n)|^2}{|\hat{a}(n)|^2 + \alpha n^{2p}} \frac{\hat{b}(n)}{\hat{a}(n)}$$

□

1.6 Numerische Ent-Faltung, Glätten von Messdaten

1.61 Motivation (Problemstellung)
Wir haben die gleichen Voraussetzungen wie im letzten Paragraphen und wir haben das Problem $a * u + \varepsilon = b$ gegeben, wobei a, u und b 2π-periodisch sind. und $\|\varepsilon\|_{L^2} \approx \delta$.
Wir haben die Bedingungen, daß

$$\|u^{(p)}\|_{L^2} = \min, \qquad \|a * u - b\|_{L^2} \leq \delta$$

b wird nun in diskreten Punkten $x_j = \frac{2\pi}{N} j$ gemessen.
Wir ersetzen also b durch das trigonometrische Interpolationspolynom b_N und führen dabei die Bedingungen auf die folgenden zurück:

$$\|u_N^{(p)}\|_{L^2} = \min, \qquad \|a * u_N - b_N\|_{L^2} \leq \delta$$

1.62 Definition (Regularisierung)
Wir wählen $\alpha > 0$ und erhalten das Regularisierungsproblem, daß

$$\|a * u_N - b_N\|_{L^2}^2 + \alpha \|u_N^{(p)}\|_{L^2}^2 = \min$$

unter allen trigonometrischen Polynomen

$$u_N(x) = {\sum_{n=-\frac{N}{2}}^{\frac{N}{2}}}' \hat{u}_N(n) e^{inx}$$

Wir wissen dann aus dem Satz aus Abschnitt 5, daß

$$\hat{u}_N(n) = \Phi_\alpha(n) \frac{\hat{b}_N(n)}{\hat{a}_N(n)}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2} - 1$$

1.63 Algorithmus (Praktische Berechnung)
Wir haben $b(x_j)$ für $j = 0, \ldots, N-1$ gegeben sowie $a(x)$ bzw. $\hat{a}(n)$.

1. Wir berechnen mit der FFT wie im Paragraph 4
$$\left(\hat{b}_N(n)\right)_{n=-\frac{N}{2}}^{\frac{N}{2}-1} = \frac{1}{N} \bar{\mathcal{F}}_N \left(b(x_j)\right)_{j=0}^{N-1}$$
mit insgesamt $N \log_2 N$ Operationen.
2. Die Fourierkoeffizienten der Apparatefunktion $\hat{a}(n)$ sind entweder gegeben (oft ist sogar nur $\hat{a}(n)$ und nicht a) oder wir approximieren $\hat{a}_M(n)$ mit $M \geq N$, evtl. sogar $M \gg N$.
3. Dann berechnen wir:
$$\hat{u}_N(n) = \frac{\overline{\hat{a}(n)} \hat{b}(n)}{|\hat{a}(n)|^2 + \alpha n^{2p}}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2} - 1$$
4. Danach berechnen wir mit $N \log_2 N$ Operationen die diskrete Fouriertransformation über FFT:
$$\left(u_N(x_j)\right)_{j=0}^{N-1} = \mathcal{F}_N \left(\hat{u}_N(n)\right)_{n=-\frac{N}{2}}^{\frac{N}{2}-1}$$

1.64 Bemerkung (Wahl des Regularisierungsparameter)
Wir wollen wissen, wie man zum Regularisierungsparameter α kommt bzw. wie man ihn relativ gut approximiert.

Falls die Schätzung (dieser Term wird auch Streuung genannt)

$$\delta \approx \left(\frac{1}{N} \sum_{j=0}^{N-1} |\varepsilon(x_j)|^2 \right)^{\frac{1}{2}} = \|\varepsilon\|_{L^2}$$

bekannt ist, dann beginnen wir mit irgendeinem α und berechnen (mit der Parsevalformel):

$$d_\alpha^2 = \|a * u_N - b_N\|_{L^2}^2 = \sum_{N=-\frac{N}{2}}^{\frac{N}{2}}{}' |\hat{a}(n)\hat{u}_N(n) - \hat{b}_N(n)|^2 = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' (1 - \Phi_\alpha(n))^2 \, |\hat{b}_N(n)|^2$$

Wir wählen dann α so, daß $d_\alpha \approx \delta$.
Wir beachten, daß $\alpha \mapsto d_\alpha$ monoton wachsend ist, d.h. wir können diesen Vorgang relativ einfach iterieren.

Bei einem optimalen α berechnen wir $\hat{u}_N(n)$ und mit FFT dann $u_N(x_j)$.

1.65 Bemerkung
Falls die Streuung unbekannt ist, dann macht das Verfahren keinen Sinn. Hier gibt es jedoch statistische Verfahren, die einen optimalen Regularisierungsparameter zur Wahl von α finden.

Dies kann man in der Literatur unter „generalized cross validation“ finden.

1.66 Bemerkung (Glätten von Daten)
Wir haben gemessene Werte $b(x_j)$ und eine Streuung des Meßfehlers, die etwa δ groß ist.
Wir suchen also ein trigonometrisches Polynom u_N mit

$$\|u_N - b_N\|_{L^2}^2 = \frac{1}{N} \sum_{j=0}^{N-1} |u_N(x_j) - b(x_j)|^2 \leq \delta^2 = \min, \qquad \|u_N''\|_{L_2} = \min$$

womit wir nach unserer Formel dann dies direkt berechnen können.

Für den Spezialfall $\hat{a}(n) = 1 \quad \forall n$ ist dann $a * u = u$, was der Faltung mit der Dirac-Masse entspricht.

Wegen $p = 2$ ergibt sich dann:

$$\hat{u}_N(n) = \frac{1}{1 + \alpha n^4} \hat{b}_N(n)$$

Wir beachten, daß die hochfrequenten Anteile von $\hat{b}_N(n)$ (für die der Betrag von n groß ist) durch diese Formel herausgefiltert werden, was eine Glättung bewirkt.

Durch die inverse Fouriertransformation erhalten wir schließlich das gewünschte u_N.

1.67 Bemerkung (Vorgehen bei nicht-periodischen Daten)
Falls die Daten nicht periodisch sind, so ziehen wir hiervon eine Ausgleichsgerade ab derart, daß diese um ein Niveau schwanken und setzen diese dann periodisch fort.

1.68 Bemerkung
Eine andere Möglichkeit wäre, dies mit einem Spline zu glätten statt mit Fouriertransformationen.

1.69 Bemerkung (Differentiation gestörter Daten)
Wir möchten die Ableitung von Daten suchen, d.h. $u = b'$, also ist

$$\int_0^x u(t)\,dt = b(x) - b(0)$$

Ist $\hat{u}(n) = \imath n\hat{b}$, also

$$\underbrace{\frac{1}{\imath n}}_{=:\hat{a}(n)}\hat{u}(n) = \hat{b}(n)$$

Damit erhalten wir ein neues Minimierungsproblem:

$$\sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1}\left|\frac{1}{\imath n}\hat{u}(n) - \hat{b}_N(n)\right|^2 \leq \delta^2, \qquad \|u''\|_{L^2} = \min$$

Für den Spezialfall $\hat{a}(n) = \frac{1}{\imath n}$ ergibt sich dann:

$$\hat{u}_N(n) = \frac{n^{-2}}{n^{-2} + \alpha n^4}\imath n\hat{b}_N(n) = \frac{\imath n}{1 + \alpha n^6}\hat{b}_N(n)$$

2 Eigenwert-Probleme

Wir möchten für eine gegebene $n \times n$ Matrix A ($\in \mathbb{R}^{n\times n}$ oder $\in \mathbb{C}^{n\times n}$) Eigenwerte $\lambda \in \mathbb{C}$ und $0 \neq v \in \mathbb{C}^n$ suchen so, daß $Av = \lambda v$.

2.1 Grundlagen

2.1 Motivation

1. Es gibt vielfältige Anwendungen, in denen Eigenwerte verlangt werden.

   In der Mechanik (Physik) interessiert man sich z.B. für die Eigenschwingung von Membranen. Wenn $u(x)$ die Auslenkung auf einem Gebiet Ω ist, dann fordert man, daß

   $$-\Delta u = \lambda u \quad \text{in } \Omega, \qquad u = 0 \quad \text{auf } \partial\Omega$$

   Für ein Gitter über Ω erhalten wir die Diskretisierung

   $$Av = \lambda v$$

2. Suchmaschinen: „The 25.000.000.000 Dollar eigenvalue problem“, auf das wird später bei der Betrachtung des Google-Mechanismus eingehen werden.

2.2 Wiederholung (Charakteristisches Polynom)
Es ist $Av = \lambda v \iff \det(A - \lambda \cdot I) = \chi_A(\lambda) = 0$.
Man könnte daran denken, zuerst das charakteristische Polynom zu berechnen und dann dessen Nullstellen, um an Eigenwerte zu kommen.

2.3 Beispiel (Beispiel für schlechte Kondition)
Sei $A = \operatorname{diag}(10, 11, \ldots, 16)$ eine 7×7 Matrix.
Hier ist klar, was die Eigenwerte sind und das char. Polynom ist:

$$\chi_A(\lambda) = -\lambda^7 + 91\lambda^6 - 3535\lambda^5 + \ldots - 31813200\lambda + 57657600$$

Falls man die Nullstellen von χ_A mit einem „guten“ Algorithmus berechnet
(z.B. `C02AEE` der `Nag`-Bibliothek), ergibt sich bei Rechnung in einfacher Genauigkeit (d.h. `eps` $= 10^{-8}$):

$$9,952, \qquad 11,31 \pm \imath 0,47, \qquad 13,65, \qquad 14,26 \pm \imath 0,51, \qquad 16,23$$

Dies entspricht nicht den wirklichen Eigenwerten. Das Problem ist, daß die Berechnung der Nullstellen eines Polynoms ausgehend von dessen Koeffizienten ein schlecht konditioniertes Problem ist.

2.4 Bemerkung (Theoretische Betrachtung der schlechten Kondition)
Wir haben ein Polynom

$$p(\lambda) = \sum_{k=0}^{n} a_k \lambda^k, \qquad \bar{a}_k = a_k(1+\varepsilon_k), \qquad |\varepsilon_k| \leq \texttt{eps}$$

und wir betrachten ein gestörtes Polynom

$$p(\lambda, \texttt{eps}) = \sum_{k=0}^{n} \left(a_k + \texttt{eps} \cdot \underbrace{a_k \frac{\varepsilon_k}{\texttt{eps}}}_{=:b_k} \right) \lambda^k = p(\lambda) + \texttt{eps} \cdot q(\lambda)$$

mit $q(\lambda) = \sum_{k=0}^{n} b_k \lambda^k$ mit $|b_k| \leq |a_k|$.

Wir untersuchen nun die Nullstellen $\lambda(\varepsilon)$ von $p(\lambda, \varepsilon) = p(\lambda) + \varepsilon q(\lambda)$ in Abhängigkeit von ε.

Sei $\lambda(0) = \lambda^*$ eine einfache Nullstelle von p. Wir betrachten $p(\lambda(\varepsilon), \varepsilon) = 0$ für alle ε mit $|\varepsilon| \leq \varepsilon_0$ und wissen, daß dies nach dem Satz über implizite Funktionen differenzierbar ist und damit ist:

$$\underbrace{\frac{\partial p}{\partial \lambda}(\lambda(\varepsilon), \varepsilon)}_{=p'(\lambda(\varepsilon))\lambda'(\varepsilon)+\mathcal{O}(\varepsilon)} + \underbrace{\frac{\partial p}{\partial \varepsilon}(\lambda(\varepsilon), \varepsilon)}_{=q(\lambda(\varepsilon))} = 0$$

Damit ist:

$$\lambda'(0) = -\frac{q(\lambda^*)}{p'(\lambda^*)}, \qquad \lambda(\varepsilon) \approx \lambda^* + \varepsilon \lambda'(0)$$

Damit ist der relative Fehler:

$$\frac{|\lambda(\varepsilon) - \lambda^*|}{|\lambda^*|} \approx \frac{|\lambda(0)' \cdot \varepsilon|}{|\lambda^*|} = |\varepsilon| \cdot \underbrace{\left| \frac{q(\lambda^*)}{\lambda^* p'(\lambda^*)} \right|}$$

Der markierte hintere Ausdruck kann unter Umständen sehr groß werden.

In unserem Beispiel ist für $\lambda^* = 10$ dann $|q(\lambda^*)| \approx 10^9$ und $|p'(\lambda^*)| = 720$ und damit ist der relative Fehler bereits bei $\varepsilon \cdot 10^5$, womit wir bei Maschinengenauigkeit bereits sehr viele Stellen verlieren.

Daher ist es numerisch nicht sinnvoll, die Koeffizienten des char. Polynom zu berechnen.

2.5 Proposition (Erhalten der Eigenvektoren bei Ähnlichkeitstransformationen)
Falls $B = T^{-1}AT$ ist, dann ist

$$Av = \lambda v \iff TBT^{-1}v = \lambda v \iff B(T^{-1}v) = \lambda(T^{-1}v)$$

Damit haben B und A dieselben Eigenwerte und v ist genau dann ein Eigenvektor von A, wenn $T^{-1}v$ ein Eigenvektor von B ist.

2.6 Definition (unitäre Matrix)
$U \in \mathbb{C}^{n\times n}$ heißt unitär, wenn $U^*U = I$, wobei $U^* = \bar{U}^T$, also $U^{-1} = U^*$.
Falls U reell ist, dann ist U orthogonal, d.h. $U^* = U^T = U^{-1}$.

2.7 Satz (Schursche Normalform)
Sei $A \in \mathbb{C}^{n\times n}$. Dann gibt es ein U unitär so, daß

$$U^*AU = \begin{pmatrix} \lambda_1 & \star & \dots & \star \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \star \\ 0 & \dots & 0 & \lambda_n \end{pmatrix}$$

also eine obere Dreiecksform hat. Diese Form heißt Schursche Normalform.

Beweis
$\chi_A(\lambda)$ hat eine Nullstelle $\lambda_1 \in \mathbb{C}$, welche der Eigenwert von A ist.
Dann gibt es also einen Eigenvektor $0 \neq v_1 \in \mathbb{C}^n$ so, daß $Av_1 = \lambda_1 v_1$. Wir können ohne Einschränkung annehmen, daß $\|v_1\|_2 = 1$. Wir konstruieren nun (nach Gram-Schmidt) eine Matrix $V_1 = (v_1, v_2, \dots, v_n)$ so, daß $v_2, \dots, v_n$ so gewählt sind, daß $v_1, v_2, \dots, v_n$ eine ONB des $\mathbb{C}^n$ sind, d.h. $V_1^*V_1 = I$ eine unitäre Matrix.

Wir betrachten nun

$$AV_1 = (Av_1, Av_2, \dots, Av_n) = V_1 \begin{pmatrix} \lambda_1 & \star \\ 0 & \hat{A} \end{pmatrix}$$

Wir verfahren nun mit der Matrix $\hat{A}$ genauso. Wir erhalten dann:

$$A \underbrace{V_1 \cdot \ldots \cdot V_n}_{=:U} = AU = \begin{pmatrix} \lambda_1 & \star & \dots & \star \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \star \\ 0 & \dots & 0 & \lambda_n \end{pmatrix} \underbrace{V_n^* \cdot \ldots \cdot V_1^*}_{=U^*}$$

Damit folgt durch Invertieren und die Tatsche, daß die Inversen von unitären Matrizen wieder unitär sind, die Behauptung. □

2.8 Historische Notiz
Dieser Satz geht auf Schur, 1909, zurück.

2.9 Definition (normale Matrix)
Eine Matrix A heißt *normal*, wenn $AA^* = A^*A$.
Dies ist insbesondere der Fall, wenn $A = A^*$, d.h. A ist hermitesch bzw. A ist symmetrisch, wenn A reell ist.

Für $A = -A^*$ heißt A schiefhermitesche.

2.10 Satz (Hermite)
Sei $A \in \mathbb{C}^{n\times n}$ normal. Dann gibt es ein U unitär so, daß $U^*AU = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ eine Diagonalmatrix ist.

Beweis
Nach der Schurschen Normalform gibt es U unitär mit $U^*AU =: R$ eine obere Dreiecksmatrix.

Falls A normal ist, dann ist R normal, denn (aufgrunddessen, daß A normal ist):

$$R^*R = U^*A^*UU^*AU = U^*A^*AU = U^*AA^*U = U^*AUU^*A^*U = RR^*$$

Jede normale obere Dreiecksmatrix ist bereits notwendigerweise diagonal. Dazu betrachten wir, daß $R^*R = RR^*$ gilt:

$$\begin{pmatrix} \lambda_1 & r_{12} & \ldots & r_{1n} \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ 0 & \ldots & 0 & \lambda_n \end{pmatrix} \cdot \begin{pmatrix} \bar\lambda_1 & 0 & \ldots & 0 \\ \bar r_{12} & \bar\lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \bar r_{11} & \ldots & \bar r_{n-1,n} & \bar\lambda_n \end{pmatrix}$$
$$= \begin{pmatrix} \bar\lambda_1 & 0 & \ldots & 0 \\ \bar r_{12} & \bar\lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \bar r_{11} & \ldots & \bar r_{n-1,n} & \bar\lambda_n \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & r_{12} & \ldots & r_{1n} \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ 0 & \ldots & 0 & \lambda_n \end{pmatrix}$$

Es ist (erste Zeile mal erste Spalte):

$$|\lambda_1|^2 + |r_{12}|^2 + |r_{13}|^2 + \ldots + |r_{1n}|^2 = |\lambda_1|^2 \Rightarrow r_{12} = \ldots = r_{1n} = 0$$

dies führt man induktiv fort und erhält die Behauptung. □

2.11 Satz (Jordan-Normalform)
Zu jedem $A \in \mathbb{C}^{n\times n}$ gibt es eine invertierbare Matrix T mit

$$T^{-1}AT = J = \begin{pmatrix} J_{n_1}(\lambda_1) & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & J_{n_k}(\lambda_k) \end{pmatrix}$$

mit den Jordanblöcken

$$J_i(\lambda) = \begin{pmatrix} \lambda & 0 & \dots & \dots & 0 \\ 1 & \lambda & \ddots & & \vdots \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & \lambda \end{pmatrix}$$

Beweis
Lineare Algebra II. □

2.12 Bemerkung
Auf der Nebendiagonalen eines Jordanblocks können auch Einträge $\varepsilon \neq 0$ stehen anstelle der Einsen.

2.2 Kondition des Eigenwertproblems

2.13 Motivation (Problemstellung)
Sei $A = (a_{ij})$ eine gegebene Matrix. Wegen Rundungsfehlern können wir nur mit einer gestörten Matrix $\tilde{A} = (\tilde{a}_{ij})$ arbeiten, also $\tilde{a}_{ij} = a_{ij}(1 + \varepsilon_{ij})$ mit $|\varepsilon_{ij}| \leq \texttt{eps}$.

Damit können wir schreiben:

$$\tilde{A} = A + \texttt{eps} \cdot C, \qquad |c_{ij}| \leq |a_{ij}|$$

Wir betrachten nun $A(\varepsilon) = A + \varepsilon C$ für ein kleines ε.

Wir möchten für die Eigenwerte eine Abschätzung finden wie:

$$|\lambda(\varepsilon) - \lambda(0)| \leq \text{const} \cdot \varepsilon$$

Ist dies überhaupt möglich, wenn ja, mit welcher Konstante?

Wir werden mit dem folgenden Satz sehen, daß dies abhängt von einer Konstanten, die von A abhängt.

2.14 Definition (Konditionszahl eines Eigenwerts)
$\frac{1}{|u^*v|}$ heißt die Konditionszahl des Eigenwerts λ.

2.15 Bemerkung
Wir werden im folgenden Satz sehen, daß je höher die Konditionszahl ist, desto schlechter verhält sich die „Stetigkeit“ der Eigenwerte.

2.16 Satz (Fehlerabschätzung bei einem einfachen Eigenwert)
Sei λ eine einfache Nullstelle von des charakteristischen Polynoms, χ_A.

Für ein kleines ε erfüllt ein Eigenwert von $A(\varepsilon) = A + \varepsilon C$ die folgende Relation:

$$\lambda(\varepsilon) = \lambda + \varepsilon \frac{u^*Cv}{u^*v} + \mathcal{O}(\varepsilon^2)$$

wobei v ein Eigenvektor zu A ist, d.h. $Av = \lambda v$, und u^* ist ein Linkseigenvektor von A, also $u^*A = \lambda u^* \iff A^*u = \bar{\lambda}u$.

Wir wählen u^* und v normiert, also $\|u\|_2 = \|u^*\|_2 = \|v\|_2 = 1$.

Beweis

1. Es ist $\chi_A(\lambda) = 0$ und $\chi'_A(\lambda) \neq 0$. Der Satz über implizite Funktionen liefert lokal die Existienz von ε so, daß $\chi_{A(\varepsilon)}(\lambda(\varepsilon)) = 0$ ist.
   Damit ist $\varepsilon \mapsto \lambda(\varepsilon)$ stetig diffbar. Es ist also (mit der Taylorentwicklung):
   $$\lambda(\varepsilon) = \lambda + \varepsilon\lambda'(0) + \mathcal{O}(\varepsilon^2)$$
2. Es ist $Av = \lambda v$ mit $\|v\|_2^2 = v^*v = 1$. Dann ist
   $$(A(\varepsilon) - \lambda(\varepsilon)I)v(\varepsilon) = 0, \qquad \|v(\varepsilon)\|_2^2 = 1, \qquad v(0) = v$$
   mit dem Satz über implizite Funktionen erhalten wir auch hier: $v(\varepsilon) = v + \varepsilon v'(0) + \mathcal{O}(\varepsilon^2)$.
3. Es ist $(A(\varepsilon) - \lambda(\varepsilon)I)v(\varepsilon) = 0$, d.h.
   $$(A + \varepsilon C)(v + \varepsilon v' + \mathcal{O}(\varepsilon^2)) = (\lambda + \varepsilon\lambda' + \mathcal{O}(\varepsilon^2))(v + \varepsilon v' + \mathcal{O}(\varepsilon^2))$$
   Wir multiplizieren dies aus und vergleichen die Potenzen von ε:
   $$\varepsilon^0 : Av = \lambda v, \qquad \varepsilon^1 : Cv + Av' = \lambda v' + \lambda' v$$
   d.h. wir erhalten: $(A - \lambda I)v' = -Cv + \lambda' v$. Durch Linksmultiplikation mit u^* erhalten wir wegen $u^*(A - \lambda I)$ schließlich:
   $$0 = -u^*Cv + \lambda' u^*v \Rightarrow \lambda' = \frac{u^*Cv}{u^*v}$$

□

2.17 Beispiel

1. Falls A normal ist, dann sind die Links- und die Rechtseigenvektoren identisch, weil

   $$U^*AU = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \iff U^*A = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)U^*$$

   und es ist $AU = U \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, d.h. wir erhalten, daß

   $$\frac{1}{|u^*v|} = 1$$

   falls u und v normiert sind.

   In diesem Fall ist das Problem gut konditioniert.

   Wir könnten hier sogar zeigen, daß nicht nur

   $$|\lambda(\varepsilon) - \lambda| \leq |\varepsilon| \cdot |v^*Cv|$$

   gilt, sondern sogar:

   $$|\lambda(\varepsilon) - \lambda| \leq |\varepsilon| \|C\|_2$$

2. Falls A nicht normal ist, kann u^*v beliebig klein sein, z.B.

   $$A = \begin{pmatrix} 1 & \alpha \\ 0 & 2 \end{pmatrix}, \qquad v = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad u^* = \frac{1}{\sqrt{1+\alpha^2}} \begin{pmatrix} 1 & -\alpha \end{pmatrix}$$

   Für ein sehr großes α ist u fast der zweite Einheitsvektor und es ist dann:

   $$u^*v = \frac{1}{\sqrt{1+\alpha^2}} \to 0$$

   Hier wäre das Problem sehr schlecht konditioniert.

2.18 Bemerkung (Betrachtung für mehrfache Eigenwerte)
Wir können den Beweis nicht übertragen, da die Voraussetzungen an den Satz für implizite Funktionen nicht gewährleistet sind.

Wir betrachten hier als Beispiel die Matrix mit mehrfachen Jordan-Blöcken:

$$A = \begin{pmatrix} \lambda & 1 & 0 & \ldots & 0 \\ 0 & \lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ldots & \ddots & \lambda & 1 \\ 0 & \ldots & \ldots & 0 & \lambda \end{pmatrix}$$

welche ein Jordan-Kästen der Größe $n \times n$ darstellt.
Wir betrachten das charakteristische Polynom von $A + \varepsilon C$. Dieses erfüllt:

$$\chi_{A+\varepsilon C}(\lambda) = (\lambda - x)^n + \varepsilon(-1)^{n+1}c_{n1} + \mathcal{O}(\varepsilon^2) + \mathcal{O}(\varepsilon(\lambda - x))$$

Ist nun λ ein Eigenwert von $A+\varepsilon C$, dann gilt (unter Vernachlässigung der verschwindenden Terme):

$$(\lambda - x)^n = \varepsilon(-1)^n c_{n1} + \ldots \quad \Rightarrow \quad \lambda(\varepsilon) = \lambda + \varepsilon^{\frac{1}{n}} \cdot |c_{n1}|^{\frac{1}{n}} \cdot e^{\imath \frac{k2\pi}{2n}}$$

wobei $e^{\imath \frac{k2\pi}{2n}}$ die n-te Einheitswurzel ist.

Das Problem ist also nicht gut konditioniert. Hier ist es schwer, Eigenwerte zu berechnen.

2.19 Algorithmus (QR-Algorithmus (einfache Version))
Wir möchten von einer Matrix A alle Eigenwerte berechnen. Wir betrachten dazu die einfache Iteration:

1. $A_0 := A$
2. Wir berechnen dann für alle $k = 0, 1, 2, \ldots$: $A_k = Q_k R_k$ mit der QR-Zerlegung und setzen anschließend $A_{k+1} := R_k Q_k$.

Im Wesentlichen gilt dann, daß $A_k \to R$ konvergiert, wobei das R eine rechte obere Dreiecksmatrix auf Schurscher Normalform ist, denn es gilt:

$$A = Q^* R Q \quad \text{mit } Q^* = \ldots \cdot Q_3 \cdot Q_2 \cdot Q_1 \cdot Q_0$$

2.20 Bemerkung
Wir werden im Folgenden Hilfsmittel kennenlernen, mit denen wir die Konvergenz beweisen können.

2.3 Potenzenmethode

2.21 Algorithmus (Algorithmus zur Potenzenmethode)
Zur Berechnung von einzelnen Eigenwerten und Eigenvektoren einer Matrix A betrachten wir das folgende Verfahren:

Wir betrachten die Folge der Vektoren $y_0 \in \mathbb{R}^n$ oder $y_0 \in \mathbb{C}^n$ beliebig und setzen:

$$y_{k+1} := A y_k$$

für $k = 1, 2, \ldots$, d.h. $y_k = A^k y_0$.

2.22 Definition (Rayleigh-Quotient)
Sei A eine Matrix und y_k wie im Algorithmus zur Potenzenmethode. Dann definieren wir den Rayleigh-Quotient durch:

$$\frac{y_k^* A y_k}{y_k^* y_k}$$

2.23 Satz (Konvergenz des Verfahrens)
Sei A diagonalisierbar und $T^{-1}AT = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ und $T = (v_1 | \ldots | v_n)$ und $Av_i = \lambda v_i$.

Es gelte außerdem, daß $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$.

Falls $y_n = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$ mit $\alpha_1 \neq 0$, dann gilt für $y_{k+1} = Ay_k$:

1. Es gilt für y_k:
$$y_k = \lambda_1^k \left(\alpha_1 v_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right)$$
Wir beachten, daß dann $\frac{1}{\lambda_1^k} y_k$ gegen den Eigenvektor v_1 konvergiert, der zum größten Eigenwert gehört.
2. Für den Rayleigh-Quotient gilt:
$$\frac{y_k^* A y_k}{y_k^* y_k} = \lambda_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right)$$
Falls A normal ist, dann gilt:
$$\frac{y_k^* A y_k}{y_k^* y_k} = \lambda_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^{2k} \right)$$

Beweis

1. Sei $y_0 = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$ mit $\alpha_0 \neq 0$. Dann ist:
$$y_1 = Ay_0 = \alpha_1 \lambda_1 v_1 + \alpha_2 \lambda_2 v_2 + \ldots + \alpha_n \lambda_n v_n$$
und iterativ erhalten wir:
$$\begin{aligned} y_k &= A^k y_0 \\ &= \alpha_1 \lambda_1^k v_1 + \alpha_2 \lambda_2^k v_2 + \ldots + \alpha_n \lambda_n^k v_n \\ &= \lambda_1^k \left(\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \ldots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k v_n \right) \end{aligned}$$

2. Es ist (Wir beachten, daß wir o.B.d.A. $\|v_1\| = 1_2$ wählen):
$$y_k^* y_k = |\lambda_1|^{2k} |\alpha_1|^2 \underbrace{v_1^* v_1}_{=1} + 2 \operatorname{Re}\left(\bar{\alpha}_1 \alpha_2 \bar{\lambda}_1^k \lambda_2^k \right) + |\alpha_2|^2 |\lambda_2|^2 v_2^* v_2 + \ldots$$

Dann ist:

$$y_k^* A y_k = y_k^* y_{k+1} = |\alpha_1|^2 |\lambda_1|^{2k} \lambda_1 + 2 \cdot \mathrm{Re}\left(\bar{\alpha}_1 \bar{\lambda}_1^k \alpha_2 \lambda_2^{k+1} v_1^* v_2\right) + |\alpha_2|^2 |\lambda_2|^{2k} \lambda_2 + \dots$$

Dann ist:

$$\frac{y_k^* A y_k}{y_k^* y_k} = \frac{|\alpha_1|^2 \cdot |\lambda_1|^{2k} \lambda_1 \left(1 + \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)\right)}{|\alpha_1|^2 |\lambda_1|^{2k} \left(1 + \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)\right)} = \lambda_1 \left(1 + \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)\right)$$

Bei normalen Matrizen fällt der Term $v_1^* v_2 = 0$ weg (da die Eigenvektoren orthogonal sind) und es folgt auch für diese Matrizen die Behauptung. □

2.24 Beispiel
Wir haben

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

und es ist $\lambda_1 = 2 + \sqrt{2} = 3,4142$ der größte Eigenwert.
Wir bekommen für einen Startvektor y_0 die folgende Iteration:

$$y_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \qquad y_1 = \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}, \qquad y_2 = \begin{pmatrix} 10 \\ 14 \\ 10 \end{pmatrix}, \dots$$

und der Rayleight-Quotient ist dann:

$$\frac{y_1^* A y_1}{y_1^* y_1} = \frac{y_1^* y_2}{y_1^* y_1} = \frac{116}{34} \approx 3,4117$$

2.25 Bemerkung
Um einen Overflow zu verhindern (die Zahlen werden immer größer als die Maschinenzahl), setzen wir (wobei $(z_{k+1})_{\max}$ die betragsgrößte Komponente von z_{k+1} ist):

$$z_{k+1} := A y_k, \qquad y_{k+1} := \frac{1}{(z_{k+1})_{\max}} z_{k+1}$$

2.26 Bemerkung (Anwendung der Potenzenmethode (Google))
Was Google ausmacht ist ein Algorithmus, der eine geeignete Reihenfolge der Ergebnisse liefert, der sog. Page-Rank-Algorithmus, bei dem es darum geht, die Wichtigkeit von Webseiten zu charakterisieren.
Google bestimmt den Rang $r(P)$ einer Seite P dadurch, daß

$$r(P) = \sum_{Q \in B_p} \frac{r(Q)}{|Q|}$$

wobei $B_p = \{\text{alle Seiten, die auf } P \text{ verweisen }\}$, und wobei $|Q|$ die Anzahl der Verweise von Q (egal auf welche Seite!) sind.

Dies ist eine rekursive Definition. Wir iterieren dann:

$$r_{k+1}(P) = \sum_{Q \in B_p} \frac{r_k(Q)}{|Q|}$$

und wir setzen

$$y_k := \begin{pmatrix} r_k(P_1) \\ \vdots \\ r_k(P_n) \end{pmatrix}$$

Dann ist mit der Potenzenmethode $y_{k+1} = Ay_k$ mit $A = (a_{ij})$, definiert durch

$$a_{ij} = \begin{cases} \frac{1}{|P_j|}, & \text{falls } P_j \text{ verweist auf } P_i \\ 0, & \text{sonst} \end{cases}$$

Die Spaltensumme ist 1, damit ist 1 der betragsgrößte Eigenwert.

Zu diesem Problem gibt es einen Artikel 25,000,000,000 *Dollar eigenvalue problem: The linear algebra behind Google* von Bryan und Leise (2005) und Langville und Meyer (2006) in SIAM Review.

2.27 Motivation (Konvergenzgeschwindigkeit)
Die Potenzenmethode konvergiert langsam, falls $\left|\frac{\lambda_2}{\lambda_1}\right| \approx 1$ ist, was uns sehr traurig stimmt. Dies bringt uns zur inversen Potenzenmethode, der sog. *Wielandt-Iteration*

2.28 Algorithmus (Inverse Potenzenmethode (Wielandt-Iteration))
Sei eine Approximation μ an einen gesuchten Eigenwert λ_1 bekannt, welcher nicht notwendigerweise der größte Eigenwert sein muß. Sei nun

$$|\mu - \lambda_1| \ll |\mu - \lambda_j| \quad \forall j = 2, \ldots, n$$

und damit ist dann auch:

$$\frac{1}{|\mu - \lambda_1|} \gg \frac{1}{|\mu - \lambda_j|}$$

Aber $\frac{1}{\mu - \lambda_j}$ sind die Eigenwerte der Matrix $(\mu I - A)^{-1}$, d.h. wir wenden die Potenzenmethode auf $(\mu I - A)^{-1}$ an (ohne diese Matrix zu berechnen). Wir berechnen nur das LGS dieser Matrix.

Sei also y_0 ein Startvektor und wir lösen dann im k-ten Schritt:

$$(\mu I - A)y_{k+1} = y_k \quad k = 0, 1, 2, \ldots$$

Wir benötigen hier nur eine LR-Zerlegung für alle Iterationsschritte (da die Matrix für jeden Schritt dieselbe ist).

2.29 Bemerkung (Schätzung des Eigenwerts und Eigenvektors)
Wir können z.B. mit der normalen Potenzenmethode den Eigenwert und Eigenvektor relativ gut schätzen.

2.30 Beispiel
Sei

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

Sei $\mu = 3,41$ und wir wählen $y_0 = \begin{pmatrix} 1 \\ 1,4 \\ 1 \end{pmatrix}$. Wir erhalten:

$$\frac{y_1^*(\mu I - A)^{-1} y_1}{y_1^* y_1} = \frac{y_1^* y_2}{y_1^* y_1} = -237,3288707 \approx \frac{1}{\mu - \lambda_1}$$

womit wir dann $\lambda_1 \approx 3,414213562$ erhalten und alle angegebenen Stellen mit dem größten Eigenwert, $2 + \sqrt{2}$, übereinstimmen.

2.4 Simultane Iteration und QR-Algorithmus

2.31 Motivation
Im Folgenden sei A eine reelle Matrix, für deren Eigenwerte gilt, daß

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$$

d.h. insbesondere, daß A diagonalisierbar ist und falls A reell ist, sind alle Eigenwerte auch reell.

Wir möchten nun auch den zweiten, dritten, usw. Eigenwert berechnen.

2.32 Wiederholung (Potenzenmethode)
Sei y_0 beliebig und $y_{k+1} = Ay_k$. Wir wissen, daß gilt:

$$y_k = \lambda_1^k \left(\alpha_1 v_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right)$$

wobei wir annehmen, daß v_1 ein Eigenvektor ist mit $\|v_1\|_2 = 1$.
Damit ist $\frac{y_k}{\|y_k\|} \to v_1$, falls $\lambda_1 > 0$ und es ist: $(-1)^k \frac{y_k}{\|y_k\|} \to v_1$, falls $\lambda_1 < 0$.
Allgemein also $(\operatorname{sgn} \lambda_1)^k \frac{y_k}{\|y_k\|} \to v_1$.

2.33 Algorithmus (Erweiterung des Algorithmus)
Sei q_0 beliebig mit $\|q_0\|_2 = 1$.
Wir betrachten die Iteration $Aq_k = \lambda_1^{(k+1)} q_{k+1}$ mit $\|q_{k+1}\|_2 = 1$ und $\operatorname{sgn} \lambda_1^{(k+1)} = \operatorname{sgn} q_k^T A q_k$. Falls k sehr groß ist, ist dies bereits $\operatorname{sgn} \lambda_1$.
Wir wissen, daß $q_k \to v_1$ konvergiert und $\lambda_1^{(k)} \to \lambda_1$ mit der Konvergenzrate $\left| \frac{\lambda_2}{\lambda_1} \right|$.

2.34 Idee (Berechnung des nächsten Eigenwerts)
Seien nun λ_1, v_1 bekannt und wir möchten λ_2, v_2 berechnen.
Wir betrachten das orthogonale Komplement zu $\mathbb{R}v_1$:

$$V = \{u \in \mathbb{R}^n | v_1^T u = 0\}$$

Wir wissen $\dim V = n - 1$. Weiter betrachten wir die folgenden Abbildung:

$$L_1 := P \circ A_{|V} : V \to \mathbb{R}^n \to V$$

wobei P eine orthogonale Projektion: $\mathbb{R}^n \to V$ ist.

Wir haben also $\mathbb{R}^n \ni q = \alpha v_1 + u$, wobei $u \in V$ ist. Dann ist $v_1^T q = \alpha$, womit wir α erhalten, wobei wir beachten, daß $\|v_1\| = 1$ ist.

Dann ist $Pq = u = q - \alpha v_1 = (I - v_1 v_1^T) q$. Damit ist für $u \in V$:

$$L_1(u) = (I - v_1 v_1^T) Au$$

Mit dem nächsten Hilfssatz erhalten wir, daß L_1 gerade die restlichen Eigenwerte von A hat.

2.35 Hilfssatz (Eigenwerte von L1)
Die Eigenwerte von L_1 sind $\lambda_2, \ldots, \lambda_n$.

Beweis
Nach der Schurschen Normalform wissen wir, daß sich für die Matrix A ergibt:

$$U^*AU = \begin{pmatrix} \lambda_1 & \star & \star \\ 0 & \ddots & \star \\ 0 & 0 & \lambda_n \end{pmatrix} = R$$

Wobei $U = (u_1|\dots|u_n)$ eine unitäre Matrix mit $u_1 = v_1$ ist.

Die Vektoren $(u_2, \dots, u_n)$ bilden eine ONB von V. Dann ist

$$\begin{aligned} L_1(u_i) &= (I - v_1 v_1^T) A u_i \\ &= (I - v_1 v_1^T)\,(v_1 r_{1,i} + \dots + u_{i-1} r_{i-1,i} + u_i \lambda_i) \\ &= u_2 r_{2,i} + \dots + u_{i-1} r_{i-1,i} + u_i \lambda_i \end{aligned}$$

Aus dieser Formel können wir die Darstellungsmatrix von L_1 bezüglich der Basis $(u_2, \dots, u_n)$ ablesen, welche folgendermaßen aussieht:

$$\begin{pmatrix} \lambda_2 & \star & \star \\ 0 & \ddots & \star \\ 0 & 0 & \lambda_n \end{pmatrix}$$

Und diese Matrix hat gerade die Eigenwerte $\lambda_2, \dots, \lambda_n$. □

2.36 Algorithmus (Berechnung des zweiten Eigenwerts)
Um λ_2 zu berechnen, wenden wir die Potenzenmethode auf L_1 an:
Sei p_0 beliebig mit $\|p_0\|_2 = 1$ und $p_0 \perp v_1$, weil $p_0 \in V$. Dann gilt mider Potenzenmethode:

$$\underbrace{(I - v_1 v_1^T) A p_k}_{= A p_k - (v_1^T A p_k) v_1} = \lambda_2^{(k+1)} p_{k+1} \qquad \text{mit } \|p_{k+1}\|_2 = 1, \qquad \operatorname{sgn} \lambda_2^{(k+1)} = \operatorname{sgn} \underbrace{p_k^T L_1(p_k)}_{= p_k^T A p_k}$$

Damit ist $\lambda_2^{(k)} \to \lambda_2$ und $p_k \to u_2$ usw. und wir erhalten die Schursche Normalform.

2.37 Algorithmus (Abänderung des Algorithmus)
Wir berechnen nicht zuerst λ_1 und v_1, sondern vernwenden die *simultane Iteration* für λ_1 und λ_2, wodurch wir den folgenden Algorithmus erhalten:

Seien q_0, p_0 beliebig und $\|q_0\|_2 = \|p_0\|_2 = 1$ und $q_0 \perp p_0$. Dann berechnen wir:

$$A q_k = \lambda_1^{(k+1)} q_{k+1} \|q_{k+1}\|_2 = 1 \operatorname{sgn} \lambda_1^{(k+1)} = \operatorname{sgn} q_k^T A q_k$$

und es ist

$$Ap_k = (q_{k+1}^T Ap_k)q_{k+1} = \lambda_2^{(k+1)} p_{k+1}\|p_{k+1}\| = 1 \operatorname{sgn} \lambda_2^{(k+1)} = \operatorname{sgn} p_k^T Ap_k$$

wobei q_{k+1} v_1 darstellt und wir haben dann wieder, daß $p_{k+1} \perp q_{k+1}$ gilt.

2.38 Bemerkung (Alternative Schreibweise für die simultante Iteration)
Wir schreiben alternativ, aber äquivalent zu oben:

$$A \underbrace{(q_k, p_k)}_{=U_k} = \underbrace{(q_{k+1}, p_{k+1})}_{=U_{k+1}} \underbrace{\begin{pmatrix} \lambda^{(k+1)} & \alpha_{k+1} \\ 0 & \lambda_2^{(k+1)} \end{pmatrix}}_{=R_{k+1}}$$

Dies ist gerade die QR-Zerlegung einer Matrix.

2.39 Algorithmus (Verallgemeinerung des Algorithmus)
Wir wählen für U_0 eine beliebige orthogonale Matrix (z.B. $U_0 = I$).
Wir betrachten die Iteration:

$$AU_k = U_{k+1}R_{k+1}$$

Dies stellt gerade die QR-Zerlegung dar, dann konvergiert

$$R_k \to \begin{pmatrix} \lambda_1 & \star & \star \\ 0 & \ddots & \star \\ 0 & 0 & \lambda_n \end{pmatrix}$$

und es konvergiert $u_k \to (u_1, \ldots, u_n)$.
Wir erhalten also die Schursche Normalform.

2.40 Algorithmus (QR-Algorithmus)
Wir setzen wie oben: $Q_k := U_{k-1}^T U_k$.
Dann ist:

$$Q_{k+1}R_{k+1} = U_k^T U_{k+1} R_{k+1} = U_k^T AU_k = U_k^T \underbrace{AU_{k-1}}_{U_k R_k} Q_k = R_k Q_k$$

Damit erhalten wir den folgenden Algorithmus:

1. $A_0 = A = Q_0 R_0$ (QR-Zerlegung im letzten Schritt)
2. $A_1 = R_0 Q_0 = Q_1 R_1$ (QR-Zerlegung)
3. $A_2 = R_1 Q_1 = Q_2 R_2$ (QR-Zerlegung), usw.

2.41 Historische Notiz (QR-Algorithmus)
Der Algorithmus geht zurück auf Rutishauser, 1958, der allerdings die LR-Zerlegung verwendet hatte. Der QR-Algorithmus, den wir hier betrachten, geht zurück auf Francis, 1961 und Kublanosvkaya, 1961.

2.42 Motivation (Ausblick)

1. Die QR-Zerlegung einer beliebigen Matrix ist mit $\mathcal{O}(n^3)$ Operationen ist zu aufwendig.

   Wir transformieren daher in Zukunft A zunächst in Hessenberg-Form, also $Q^T AQ = H$ (welche auf Dreiecksform ist und eine Diagonale unterhalb der Hauptdiagonalen hat), was etwa so aufwendig wie eine QR-Zerlegung ist.

   Falls A symmetrisch ist, dann erhalten wir für H eine Tridiagonalmatrix.

   Für eine die QR-Zerlegung einer Hessenberg-Matrix benötigen wir nur $\mathcal{O}(n^2)$ Operationen bzw. $\mathcal{O}(n)$ Operationen, falls A symmetrisch ist. Dann sind alle A_k wieder Hessenberg-Matrix, was wir im nächsten Kapitel sehen werden.

2. Die Konvergenzgeschwindigkeit ist sehr langsam, etwa $\left|\frac{\lambda_2}{\lambda_1}\right|$.
   Wir betrachten die Idee, daß wir die Matrix A shiften, d.h. wir betrachten $A_k - \mu_k I$, wobei wir den Parameter μ_k so wählen, daß die Konvergenz beschleunigt wird. Dieses Verfahren werden wir im sechsten Kapitel betrachten.

3. Wir haben noch keine komplexen Eigenwerte erfasst. Wir können im Fall von komplexen, nicht reellen Eigenwerte A_k nicht gegen reelle obere Dreiecksmatrix konvergieren lassen. Dann konvergiert A gegen eine Matrix in Schurscher Normalform mit einem 2×2-Block auf der Hauptdiagonalen, der dessen Eigenwert

   $$\begin{pmatrix} a_{r,r}^{(k)} & a_{r,r+1}^{(k)} \\ a_{r+1,r}^{(k)} & a_{r+1,r+1}^{(k)} \end{pmatrix}$$

   Dies konvergiert gegen den Eigenwert $\alpha \pm \imath\beta$, was wir im siebten Kapitel sehen werden.

2.5 Transformation auf Hessenberg-Form

2.43 Satz (Transformation auf Hessenberg-Form)
Sei $A \in \mathbb{R}^{n\times n}$. Diese Matrix kann durch $(n-2)$ Householder-Transformationen auf Hessenberg-Form transformiert werden:

$$Q^T A Q = H = \begin{pmatrix} \star & \star & \star \\ \ddots & \star & \star \\ 0 & \ddots & \star \end{pmatrix}$$

wobei $Q := Q_1 \cdot \ldots \cdot Q_{n-2}$ und $Q_i := I - 2u_i u_i^T$ Householder-Transformationen sind.

Falls A symmetrisch ist, ist H tridiagonal.

Beweis

1. Wir wählen $\tilde{Q}_1 = I - 2u_1 u_1^T$ (mit $u_1^T u_1 = 1$ als eine $(n-1)\times(n-1)$ Householder-Matrix) so, daß

   $$\tilde{Q}_1 \begin{pmatrix} a_{21} \\ \vdots \\ a_{n1} \end{pmatrix} = \begin{pmatrix} \star \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

   Dann erhalten wir:

   $$Q_1 = \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix} \text{ und } A^{(1)} = Q_1 A_1 Q_1^T = \left(\begin{array}{c|c} \star & \\ \star & \\ 0 & \star \\ \ldots & \\ 0 & \end{array}\right) \cdot \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix} = \left(\begin{array}{c|c} \star & \\ \star & \\ 0 & \star \\ \ldots & \\ 0 & \end{array}\right)$$

2. Wir wählen $\tilde{Q}_2 := I - 2\tilde{u}_2 \tilde{u}_2^T$ eine $(n-2)\times(n-2)$ Householder-Matrix. Wir erhalten:

   $$\tilde{Q}_2 \begin{pmatrix} a_{32}^{(1)} \\ \ldots \\ a_{n2}^{(1)} \end{pmatrix} = \begin{pmatrix} \star \\ 0 \\ \ldots \\ 0 \end{pmatrix} Q_2 = \begin{pmatrix} I_2 & 0 \\ 0 & \tilde{Q}_2 \end{pmatrix} \Rightarrow A^{(2)} = Q_2 A^{(1)} Q_2^T = \left(\begin{array}{cc|c} \star & \star & \\ \star & \star & \\ 0 & \star & \\ \ldots & 0 & \star \\ \ldots & \ldots & \\ 0 & 0 & \end{array}\right)$$

3. Wir fahren so induktiv fort und erhalten:

   $$Q^T A Q = H$$

   mit $Q^T := Q_{n-2} \cdot \ldots \cdot Q_2 \cdot Q_1$.

4. Für symmetrische Matrizen bleibt unter Householder-Transformationen die Symmetrie erhalten und daher muß die Householdermatrix tridiagonal sein. □

2.44 Bemerkung (Betrachtung des Rechenaufwands)
Es sind etwa $\frac{5}{3}n^3$ Operationen zu machen für ein allgemeines A und für symmetrische Matrizen sind es $\frac{2}{3}n^3$ Operationen.

2.45 Satz (Vererbung der Hessenberg-Form)
Sei $H = QR$ eine QR-Zerlegung, wir setzen $\bar{H} := RQ$. Ist H eine Hessenberg-Matrix, so ist auch $\bar{H}$ eine Hessenberg-Matrix.
Ist H tridiagonal und symmetrisch, so ist auch $\bar{H}$ auch wieder tridiagonal und symmetrisch.

Beweis
Durch eine Householdertransformation erhalten wir (für die QR-Zerlegung):

$$Q_1 H = \left(\begin{array}{c|c} \star & \\ 0 & \\ \dots & \star \\ 0 & \end{array}\right) \text{ und } Q_1 := I - 2u_1 u_1^T \text{ mit } u_1 = \begin{pmatrix} \star \\ \star \\ 0 \\ \dots \\ 0 \end{pmatrix}$$

Dann ist:

$$RQ_1 = \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & * \end{pmatrix} \begin{pmatrix} \star & \star & 0 \\ \star & \star & \\ 0 & & I_{n-2} \end{pmatrix} = \begin{pmatrix} * & * & * \\ \star & * & * \\ 0 & 0 & * \end{pmatrix}$$

d.h. wir erhalten eine obere Dreiecksmatrix mit einem Eintrag auf der Nebendiagonalen. Per Induktion erhalten wir schließlich die Behauptung. □

2.46 Algorithmus (Modifikation des QR-Algorithmus)

1. Wir führen eine Vortransformation durch, d.h. wir bringen $Q^T AQ = H_0$ auf Hessenberg-Form. Hierfür brauchen wir $\mathcal{O}(n^3)$ Schritte.
2. Anwenden des klassischen QR-Algorithmus auf H_0, also $H_k = Q_k R_k$. Dies benötigt allgemein $\mathcal{O}(n^2)$ Operationen, im symmetrischen Fall nur $\mathcal{O}(n)$ Operationen.
3. $H_{k+1} := R_k Q_k$, was ebenfalls allgemein $\mathcal{O}(n^2)$ Operationen benötigt und im symmetrischen Fall $\mathcal{O}(n)$ Operationen.

2.6 QR-Algorithmus mit Shift

2.47 Motivation
Wir wollen nun stets voraussetzen, daß A nur reelle Eigenwerte haben, welche paarweise verschieden sind, also $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$. Außerdem arbeiten wir auf der Hessenberg-Matrix $H = Q^T AQ$.

Wir möchten nun die Konvergenzgeschwindigkeit mit einer Shift-Idee verbessern.

2.48 Idee (Shift)
Wir erwarten nach dem vierten Kapitel, daß wir die Konvergenzrate

$$|h_{i+1,i}^{(k)}| = \mathcal{O}\left(\left|\frac{\lambda_{i+1}}{\lambda_i}\right|^k\right)$$

haben. Dies kann unter Umständen sehr langsam geschehen.

Wir betrachten nun die geshiftete Matrix $\tilde{H} = H - \mu I$, welche die Eigenwerte $\lambda_i - \mu$ hat. Wenden wir nun hieraus den QR-Algorithmus an, erhalten wir:

$$|\tilde{h}_{i+1,i}^{(k)}| = \mathcal{O}\left(\left|\frac{\lambda_{i+1} - \mu}{\lambda_i - \mu}\right|^k\right)$$

Dies konvergiert dann sehr rasch, wenn $\mu \approx \lambda_{i+1}$ ist.

2.49 Bemerkung (Konvention)
Wir können im Folgenden o.B.d.A. annehmen, daß H eine nichtreduzierte Hessenbergmatrix ist, d.h. für alle i gilt: $h_{i+1,i} \neq 0$. Wäre ein Element Null, so hätte die Matrix die Gestalt:

$$H = \begin{pmatrix} H_1 & \star \\ 0 & H_2 \end{pmatrix}$$

und dann sind die Eigenwerte von H die Eigenwerte von H_1 und H_2.

2.50 Hilfssatz
Sei H nichtreduzierte Hessenberg-Matrix und μ ein Eigenwert von H.
Sei $H - \mu I = QR$ die QR-Zerlegung. Wir betrachten $\bar{H} := RQ + \mu I$.
Dann ist $\bar{h}_{n,n} = \mu$ und $\bar{h}_{n,n-1} = 0$, d.h. die Matrix zerfällt nach einem QR-Schritt.

Beweis
Wir betrachten $H - \mu I$ mit $h_{i+1,i} \neq 0 \quad \forall i$. Dies impliziert, daß die ersten $n-1$ Spalten dieser Matrix linear unabhängig sind, d.h. wir können schreiben:

$$Q^T(H - \mu I) = \begin{pmatrix} R_{n-1} & \star \\ 0 & r_{n,n} \end{pmatrix} =: R$$

Dies ist die QR-Zerlegung, welche immer existiert und R_{n-1} muß invertierbar sein. Also ist $H - \mu I$ singulär.
Damit ist $r_{n,n} = 0$, also auch die letzte Zeile von RQ. Damit gilt:

$$\bar{H} = RQ + \mu I = \begin{pmatrix} \star & \star & \star & \star \\ \ddots & \star & \star & \star \\ & \ddots & \star & \star \\ & & 0 & \mu \end{pmatrix}$$

□

2.51 Algorithmus (QR-Algorithmus mit Shift (für reelle Eigenwerte))
Sei ohne Einschränkung H_0 eine nichtreduzierte Hessenberg-Matrix. Wir betrachten $H_k - h_{n,n}^{(k)} I = Q_k R_k$ und $H_{k+1} := R_k Q_k + h_{n,n}^{(k)} I$ für $k = 1, 2, \ldots$ solange, bis

$$|h_{n,n-1}^{(k)}| \leq \texttt{eps} \left(|h_{n,n}^{(k)} + |h_{n-1,n-1}^{(k)}| \right)$$

wobei `eps` die Maschinenegenauigkeit ist. Dann akzeptieren wir $h_{n,n}^{(k)}$ als Eigenwert.

Dann beginnen wir von Neuem mit der Submatrix $(h_{ij}^{(k)})_{i,j=1}^{n-1}$, solange bis wir zu einer 1×1-Matrix kommen.

2.52 Bemerkung (Konvergenzgeschwindigkeit)
Sei

$$H - h_{n,n} I = \begin{pmatrix} \star & \star & \star & \star & \star \\ \ddots & \star & \star & \star & \star \\ & \ddots & \star & \star & \star \\ & & \ddots & \star & b \\ & & & \varepsilon & 0 \end{pmatrix} \Rightarrow Q_{n-2} \cdot Q_{n-3} \cdot \ldots \cdot Q_1 \cdot H = \begin{pmatrix} \star & \star & \star & \star & \star \\ \ddots & \star & \star & \star & \star \\ & \ddots & \star & \star & \star \\ & & \ddots & a & b \\ & & & \varepsilon & 0 \end{pmatrix}$$

wobei Q die Form wie oben erläutert hat, daß wir eine Einheitsmatrix haben und einmal einen 2×2-Block auf der Hauptdiagonalen. Außerdem bleiben ε und b (nach Konstruktion) unverändert.

Wir sehen, daß ε unverändert bleibt. Im symmetrischen Fall ist außerdem $\mathcal{O}(\varepsilon)$.

Wir müssen also noch die QR-Zerlegung von $\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix}$ berechnen. Es ist:

$$\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix} = \underbrace{\frac{1}{\sqrt{a^2+\varepsilon^2}} \begin{pmatrix} a & -\varepsilon \\ \varepsilon & a \end{pmatrix}}_{=:\tilde{Q}} \underbrace{\begin{pmatrix} \sqrt{a^2+\varepsilon^2} & \frac{ba}{\sqrt{a^2+\varepsilon^2}} \\ 0 & -\frac{b\varepsilon}{\sqrt{a^2+\varepsilon^2}} \end{pmatrix}}_{=:\tilde{R}}$$

Dann ist

$$\tilde{R} \cdot \tilde{Q} = \begin{pmatrix} \star & \star \\ -\frac{b\varepsilon^2}{a^2+\varepsilon^2} & \star \end{pmatrix}$$

Wir untersuchen nun:

$$H - h_{n,n} I = \underbrace{Q_1 \cdot \ldots \cdot Q_{n-1}}_{=:Q} R$$

Und es ist:

$$\bar{H} = RQ + h_{n,n} I = \begin{pmatrix} \star & \star & \star & \star & \star \\ \ddots & \star & \star & \star & \star \\ & \ddots & \star & \star & \star \\ & & \ddots & \star & \star \\ & & & -\frac{b\varepsilon^2}{a^2+\varepsilon^2} & \star \end{pmatrix}$$

Wir sehen, daß das linke Element immer kleiner wird und sobald dies unterhalb der Maschinenegenauigkeit ist, akzeptieren wir dies als Eigenwert.

Falls $\varepsilon^2 \ll a^2$. Dann erhalten wir *quadratische Konvergenz*, d.h. aus $h_{n,n-1} = \mathcal{O}(\varepsilon^2)$ folgt, daß $\bar{h}_{n,n-1} = \mathcal{O}(\varepsilon^2)$.

Für symmetrische Matrizen erhält man sogar *kubische Konvergenz* (wegen $b = \varepsilon$), d.h. aus $h_{n,n-1} = \mathcal{O}(\varepsilon)$ erhalten wir, daß $\bar{h}_{n,n-1} = \mathcal{O}(\varepsilon^3)$.

2.53 Bemerkung (Stabilitätsbetrachtung des QR-Algorithmus)
Wir erhalten schließlich, daß $\hat{Q}^T A \hat{Q} = \hat{R}$ die Schursche Normalform ist.

Der QR-Algorithmus ist *stabil* im Sinne der Rückwärtsanalysis, d.h. wenn $\hat{R} = \hat{Q}^T \hat{A} \hat{Q}$ die Schursche Normalform einer gestörten Matrix ist, dann ist:

$$\|A - \hat{A}\|_2 \leq C \cdot \texttt{eps} \cdot \|A\|_2, \qquad \hat{Q}^T \hat{Q} = I + F \text{ mit } \|F\|_2 \leq c \cdot \texttt{eps}$$

Beweis
Wilkonson: The Algebraic Eigenvalue Problem, etwa 1970 □

2.7 Berechnung komplexer Eigenwerte

2.54 Motivation
Wir untersuchen nun komplexe, nicht reelle Eigenwerte von reellen Matrizen $A \in \mathbb{R}^{n\times n}$, die in Paaren von konjugiert komplexen Eigenwerten auftreten. Hierzu werden wir iterativ die reelle Schursche Normalform berechnen.

2.55 Satz (reelle Schursche Normalform)
Sei $A \in \mathbb{R}^{n\times n}$. Dann gibt es eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ mit:

$$Q^T AQ = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1m} \\ & R_{22} & & \vdots \\ & & \ddots & \vdots \\ & & & R_{mm} \end{pmatrix}$$

Dabei ist jedes R_{ii} entweder eine reelle Zahl oder eine reelle 2×2-Matrix mit konjugiert komplexen Eigenwerten.

Beweis
Wir führen eine Induktion über die Anzahl der Paare komplex konjugierter Eigenwerte, k. Falls $k = 0$ ist, hat A nur reelle Eigenwerte, kann man in der Schurschen Normalform $Q = U$ reell wählen, wie im früheren Beweis.

Für den Induktionsschritt argumentieren wir: A hat ein die folgenden konjugiert komplexe Eigenwerte, d.h. Wir haben einen Eigenwert $\lambda = \alpha + \imath\beta$ und $\bar{\lambda} = \alpha - \imath\beta$ mit $\beta \neq 0$.

Dann gibt es die linear unabhänhigen Eigenvektoren $v = x + \imath y$ und $\bar{v} = x - \imath y$ (denn aus $Av = \lambda v$ folgt, daß $\bar{A}\bar{v} = \bar{\lambda}\bar{v}$).

Es ist $Av = \lambda v = (\alpha + \imath\beta)(x + \imath y) = (\alpha x - \beta y) + \imath(\alpha y + \beta x)$. Andererseits ist auch $Av = A(x + \imath y)$. Durch Koeffizientenvergleich ergibt sich:

$$Ax = (\alpha x - \beta y), \qquad Ay = \alpha y + \beta x, \quad \text{also: } A(x, y) = (x, y) \cdot \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}$$

Da $v, \bar{v}$ linear unabhängig sind, sind es auch x, y.müssen auch x, y, denn es gilt:

$$(v, \bar{v}) = (x, y) \cdot \begin{pmatrix} 1 & 1 \\ \imath & -\imath \end{pmatrix}$$

x und y spannen einen zweidimensionalen Unterraum des $\mathbb{R}^n$ auf, den A in sich abbildet. Sei nun (u_1, u_2) eine ONB dieses Unterraums, welche wir zu einer ONB $(u_1, \ldots, u_n)$ des $\mathbb{R}^n$ ergänzen. Wir setzen $U := (u_1, \ldots, u_n) \in \mathbb{R}^{n\times n}$, was eine orthongonale Matrix ist. Es gilt:

$$AU = U \cdot \left(\begin{array}{c|c} R_{11} & \star \\ \hline 0 & \tilde{A} \end{array}\right)$$

wobei R_{11} eine 2×2-Matrix mit einem Paar konjugiert komplexer Eigenwerte ist.

Nach der Induktionsvoraussetzung gibt es $\tilde{Q}$ so, daß $\tilde{Q}^T \tilde{A} \tilde{Q}$ die Blockdreicksform hat, womit die Behauptung sofort folgt. Wir setzen schließlich:

$$Q := \begin{pmatrix} I_2 & 0 \\ 0 & \tilde{Q} \end{pmatrix} U$$

□

2.56 Idee (QR-Algorithmus für komplexe Eigenwerte)
Wir setzen betrachten ein komplexes μ_k die folgende Iteration:

$$\begin{aligned} H_k - \mu_k I &= Q_k R_k, & H_{k+1} &:= R_k Q_k + \mu_k I \\ H_{k+1} - \bar{\mu}_k I &= Q_{k+1} R_{k+1}, & H_{k+2} &:= R_{k+1} Q_{k+1} + \bar{\mu}_k I \end{aligned}$$

Hierbei ist Q_k natürlich nicht orthogonal, sondern unitär.

2.57 Hilfssatz
Wenn H_k reell ist, dann kann man die QR-Zerlegung so wählen, daß H_{k+2} wieder reell ist.

Beweis
Wir wissen, daß gilt:

$$H_{k+1} = R_k Q_k + \mu_k I = Q_k^*(H_k - \mu_k I)Q_k + \mu_k I = Q_k^* H_k Q_k$$

und analog erhalten wir:

$$H_{k+2} = Q_{k+1}^* H_{k+1} Q_{k+1} = (Q_k Q_{k+1})^* H_k (Q_k Q_{k+1})$$

Es genügt zu zeigen, daß $Q_k Q_{k+1}$ reell ist. Wir rechnen nun mit Hilfe der obigen Rechnungen:

$$\begin{aligned} Q_k Q_{k+1} R_{k+1} R_k &= Q_k (H_{k+1} - \bar{\mu}_k I) R_k = Q_k (R_k Q_k + \mu_k I - \bar{\mu}_k I) R_k \\ &= (Q_k R_k)^2 + (\mu_k - \bar{\mu}_k) Q_k R_k = (H_k - \mu_k I)^2 + (\mu_k - \bar{\mu}_k)(H_k - \mu_k I) \\ &= (H_k - \mu_k I)(H_k - \mu_k I + \mu_k I - \bar{\mu}_k I) \\ &= H_k^2 - 2\,\mathrm{Re}(\mu_k) \cdot H_k + |\mu_k|^2 I =: M_k \end{aligned}$$

und wir sehen, daß M_k reell ist. Weil $R_{k+1}R_k$ eine Dreiecksmatrix ist und Q_kQ_{k+1} eine unitäre Matrix ist, haben wir die QR-Zerlegung von M_k berechnet. Wir können die QR-Zerlegung nach der Bemerkung so wählen, daß die Diagonalelemente von R_k, R_{k+1} reell sind, erhalten wir aus der Eindeutigkeit, daß auch $Q_k \cdot Q_{k+1}$ reell ist. □

2.58 Bemerkung (Eindeutigkeit der QR-Zerlegung)
Die QR-Zerlegung ist eindeutig bis auf Multiplikation mit einer Diagonalmatrix an Q.

2.59 Bemerkung (Rechenaufwand)
Wir möchten H_{k+2} ausgehend von H_k nur mit reellen Operationen berechnen:
Wir berechnen die Matrix M_k und deren QR-Zerlegung $M_k = QR$ (für $Q = Q_kQ_{k+1}$). Dann berechnen wir $H_{k+2} = Q^T H_k Q$. Diese Berechnung von M_k erfordert $\mathcal{O}(n^3)$ Operationen.

Wir zeigen nun, daß wir H_{k+2} aus H_k in $\mathcal{O}(n^2)$ reellen Operationen berechnen können.

2.60 Hilfssatz
Sei $A \in \mathbb{R}^{n\times n}$ und $H = Q^TAQ$ eine Hessenbergmatrix mit $h_{i+1,i} \neq 0$ für $i = 1, \ldots, n-1$.

Dann können Q und H aus der ersten Spalte von Q bestimmt werden.

Beweis
Sei $Q = (q_1, \ldots, q_n)$. Wir haben $AQ = QH$, d.h. $Aq_i = \sum_{j=1}^{i+1} q_j h_{ji}$. Andererseits ist $Q^TAQ = H$, d.h. $q_j^T Aq_i = h_{ji}$ Wir nehmen nun q_1 als gegeben an. Dann wissen wir, daß $h_{11} = q_1^T Aq_1$. Dann ist q_2 ein Vielfaches von $Aq_1 - h_{11}q_1$. Damit können wir q_2 bis auf das Vorzeichen eindeutig bestimmen (wegen $\|q_2\|_2 = 1$).

Damit erhalten wir auch h_{12}, h_{21}, h_{22}. Per Induktion ergibt sich die Behauptung. □

2.61 Algorithmus (Francis' QR-Schritt)

1. Wir berechnen die erste Spalte von $M_k = M_ke_1 = H_k(H_ke_1) - 2\,\mathrm{Re}(\mu_k)(H_ke_1) + |\mu_k|^2e_1$, was mit $\mathcal{O}(n^2)$ Operationen geht.
2. Wir berechnen die Householder-Matrix Q_0 mit $Q_0(M_ke_1) = \alpha e_1$, was eine Spiegelung darstellt und es ist:
$$Q_0 = \begin{pmatrix} \star & \star & \star & \\ \star & \star & \star & 0 \\ \star & \star & \star & \\ & 0 & & I \end{pmatrix}$$
3. Wir transformieren $Q_0^T H_k Q_0$ mit Householder-Matrizen $Q := Q_0\hat{Q}_1 \cdot \ldots \cdot \hat{Q}_{n-2}$ auf Hessenberg-Form $\tilde{H}$ in $\mathcal{O}(n^2)$ Operationen, d.h. wir berechnen $Q^T H_k Q = \tilde{H}$

Dann ist $\tilde{H} = H_{k+2}$.

Beweis
Es ist:

$$Q_0^T H_k Q_0 = \begin{pmatrix} \star & \star & \star & \star & & \\ \star & \star & \star & \star & & \\ * & \star & \star & \star & & \\ * & * & \star & \star & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \end{pmatrix}$$

Wir möchten die $*$-Einträge in der ersten Spalte wegbekommen, was wir mit Householdermatrizen der Form

$$\hat{Q}_1 = \begin{pmatrix} 1 & & 0 & & 0 \\ & \star & \star & \star & \\ 0 & \star & \star & \star & 0 \\ & \star & \star & \star & \\ 0 & & 0 & & I \end{pmatrix}, \qquad \hat{Q}_2 = \begin{pmatrix} 1 & 0 & & 0 & & 0 \\ 0 & 1 & & 0 & & 0 \\ & & \star & \star & \star & \\ 0 & 0 & \star & \star & \star & 0 \\ & & \star & \star & \star & \\ 0 & 0 & & 0 & & I \end{pmatrix}$$

usw. machen. Dann ist $\hat{Q}_i e_1 = e_1$ für $i = 1, \ldots, n-2$, also gilt $Qe_1 = Q_0 e_1$.
Wegen $Q_0(M_k e_1) = \alpha e_1$ und $Q_0^{-1} = Q_0^T = Q_0$ ist $Q_0 e_1$ ein Vielfaches von $M_k e_1$.

Andererseits wußten wir, daß $M_k e_1 = (Q_k Q_{k+1}) \underbrace{(R_k R_{k+1}) e_1}_{=\tilde{\alpha} e_1}$. Dann ist $Q_k Q_{k+1} e_1$ auch ein Vielfaches von $M_k e_1$. Zudem ist $H_{k+2} = (Q_k Q_{k+1})^T H_k (Q_k Q_{k+1})$ und damit $Q_k Q_{k+1} e_1 = \pm Q e_1$.

Nach dem Hilfssatz ergibt sich bei geeigneter Wahl der Vorzeichen in den Spalten von Q_k und Q_{k+1}, daß

$$Q = Q_k Q_{k+1}, \qquad \tilde{H} = Q^T H_k Q = H_{k+2}$$ □

2.62 Bemerkung (Abbruch der Iteration)
Wir brechen die Iteration ab, falls für $\ell = n$ (reelle Eigenwerte) oder $\ell = n-1$ (komplexe Eigenwerte) gilt:

$$|h_{\ell,\ell-1}^{(k)}| \leq \texttt{eps} \left(|h_{\ell-1,\ell-1}^{(k)}| + |h_{\ell,\ell}^{(k)}| \right)$$

1. Falls $\ell = n$ ist, akzeptieren wir $h_{n,n}^{(k)}$ als den Eigenwert und beginnen neu mit $\left(h_{ij}^{(k)}\right)_{i,j=1}^{n-1}$
2. Falls $\ell = n-1$ ist, akzeptieren wir die Eigenwerte des rechten unteren 2×2-Blocks von H_k als Eigenwerte von A und beginnen neu mit $\left(h_{ij}^{(k)}\right)_{i,j=1}^{n-2}$.

2.8 Berechnung von Singulärwerten

2.63 Satz (über Singulärwerte)
Sei $A \in \mathbb{R}^{m\times n}$. Dann gibt es orthogonale Matrizen $U \in \mathbb{R}^{m\times m}$ und $V \in \mathbb{R}^{n\times n}$ so, daß

$$A = U\Sigma V^T$$

mit $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m\times n}$ und $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$, wobei $p = \min\{m, n\}$.

Die σ_j heißen *Singulärwerte* von A, welche eindeutig bestimmt sind.

Beweis
Seien $x \in \mathbb{R}^n, y \in \mathbb{R}^m$ mit $\|x\|_2 = \|y\|_2 = 1$. Außerdem soll gelten, daß $Ax = \sigma y$ mit

$$\sigma = \|A\| = \max_{\|v\|=1} \|Av\|$$

Also kann x so gewählt werden, daß $\|Ax\| = \|A\|$ ist.

Sei V_1 eine orthogonale $n \times n$-Matrix mit x in der ersten Spalte (d.h. wir ergänzen x zu einer ONB des $\mathbb{R}^n$) und U_1 eine orthogonale $m \times m$-Matrix mit y in der ersten Spalte. Dann gilt:

$$A_1 := U_1^T A V_1 = \left(\begin{array}{c|c} \sigma & \\ 0 & \\ \ldots & \star \\ 0 & \end{array}\right) = \left(\begin{array}{c|c} \sigma & w^T \\ \hline 0 & \hat{A}_1 \end{array}\right)$$

wobei w ein geeigneter Vektor und $\hat{A}_1$ eine geeignte Matrix ist. Wir betrachten nun

$$\left\| A_1 \begin{pmatrix} \sigma \\ w \end{pmatrix} \right\| = \left\| \begin{pmatrix} \sigma^2 + w^T w \\ \star \end{pmatrix} \right\| \geq \sqrt{\sigma^2 + w^T w}$$

Andererseits ist $\|A_1\| = \|A\| = \sigma$. Damit ist bereits $w = 0$. Also ist $A_1 = \begin{pmatrix} \sigma & 0 \\ 0 & \hat{A}_1 \end{pmatrix}$.

Per Induktion (auf $\hat{A}_1$) folgt die Behauptung des Satzes. □

2.64 Bemerkung
Angenommen, wir haben bereits eine solche Zerlegung, dann gilt, falls $m \leq n$ ist:

$$AA^T = U\Sigma \underbrace{V^T V}_{=I} \Sigma^T U^T = U \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_m^2 \end{pmatrix} U^T$$

Also sind die σ_j die Wurzeln der Eigenwerte von AA^T (bzw. von $A^T A$, falls $n \leq m$).

2.65 Hilfssatz
Sei die gleiche Notation wie im obigen Satz gegeben und $U = (u_1, \ldots, u_m)$ sowie $V = (v_1, \ldots, v_n)$. Seien weiter $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_p = 0$. Dann gilt:

1. $\operatorname{rg} A = r$
2. $\operatorname{Ker} A = \langle v_{r+1}, \ldots, v_n \rangle$ (und $v_{r+1}, \ldots, v_n$ ist eine ONB von $\operatorname{Ker} A$).
3. $\operatorname{Bild} A = \langle u_1, \ldots, u_r \rangle$ (und $u_1, \ldots, u_r$ ist eine ONB von $\operatorname{Bild} A$)
4. $\|A\|_2 = \sigma_1$
5. Für die Frobenius-Norm gilt:

$$\|A\|_F^2 := \sum_{i,j} a_{ij}^2 = \sum_{k=1}^{r} \sigma_k^2$$

2.66 Bemerkung
Es gilt:

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i \underbrace{u_i v_i^T}_{\text{rg}=1}$$

Falls also der Rang von A klein ist, so kann man viele Informationen darüber bereits in den Vektoren u_i und v_i speichern. Die beste Approximation an A vom Rang $k \leq r$ ist:

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

Diese Information benötigt man z.B. in der Datenkompression.

2.67 Bemerkung (weitere Anwendung)
Bei Suchmaschinen hat man oft die Methode des „latent remantic indexing“, was man in einer sog. Term Dokumenten Matrix gespeichert wird, wo für jeden Suchbegriff und jedes Dokument aufgeschrieben wird, wie oft der Begriff dort vorkommt. Damit ersetzt man also die Term Dokumentan Matrix durch eine Niedrigrang-Approximation.

2.68 Bemerkung (Beobachtung bei der Berechnung von Singulärwerten)
Man könnte den QR-Algorithmus direkt auf $A^T A$ (für $n \leq m$) anwenden bzw. auf AA^T (für $m \leq n$). Die Produktbildung ist rechenaufwendig und es entstehen Rundungsfehler. Daher betrachten wir nun einen besseren Algorithmus:

Sind P und Q orthogonal, dann haben A und PAQ dieselben Singulärwerte, denn:

$$A = U\Sigma V^T \iff PAQ = \underbrace{PU}\,\Sigma\,\underbrace{V^T Q}$$

Der folgende Hilfssatz zeigt, daß wir jede Matrix A mit solchen Transformationen P und Q auf Bidiagonalform bringen können und damit wird sich das Problem darauf reduzieren, die Singulärwerte einer Bidiagonalmatrix zu bestimmen.

2.69 Hilfssatz
Sei $A \in \mathbb{R}^{m\times n}$ (mit o.B.d.A $m \geq n$). Dann gibt es orthogonale Matrizen P, Q mit

$$PAQ = \begin{pmatrix} B \\ 0 \end{pmatrix}, \qquad B = \begin{pmatrix} \ddots & \ddots & 0 \\ 0 & \ddots & \ddots \\ 0 & 0 & \ddots \end{pmatrix}$$

also B eine $n \times n$ Bidiagonalmatrix ist.

Beweis
Wir benutzen P_i und Q_i als Householder-Transformationen. Durch Multiplikation von Links mit P_1 erhalten wir:

$$A = \begin{pmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{pmatrix} \to \left(\begin{array}{c|cc} \star & & \\ 0 & & \\ \dots & & \star \\ 0 & & \end{array}\right)$$

Multiplikation von rechts mit Q_1 ergibt:

$$\to \left(\begin{array}{c|ccc} \star & \star & 0 & \dots & 0 \\ 0 & & & & \\ \vdots & & & \hat{A} & \\ 0 & & & & \end{array}\right)$$

Per Induktion auf eine kleinere Matrix $\hat{A}$ erhalten wir die Behauptung. □

2.70 Bemerkung (Berechnung von Singulärwerten einer Tridiagonalmatrix)
Wir haben eine Matrix $B^TB =: H$, die tridiagonal ist und betrachten einen Schritt des QR-Algorithmus für $\mu := h_{n,n}$:

$$M := H - \mu I = QR$$

Dann ist $\bar{H} = RQ + \mu I = Q^T(H - \mu I)Q + \mu I = Q^THQ$.

Wir möchten nun die Eigenwerte von H berechnen, ohne die Matrix H oder die Matrix M konkret zu berechnen.

Nach Kapitel 7 sind $\bar{H}$ und Q eindeutig durch die erste Spalte von Q bestimmt. Dann ist $M = QR$ und $Q^T = Q_{n-1} \cdot \ldots \cdot Q_2 \cdot Q_1$ als Produkt von Householder-Tranformationen mit

$$Q_1 = \begin{pmatrix} \star & \star & \\ \star & \star & 0 \\ 0 & & I_{n-2} \end{pmatrix}, \qquad Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \star & \star & 0 \\ & \star & \star & \\ 0 & 0 & & I_{n-3} \end{pmatrix}, \qquad \text{usw.}$$

Die erste Spalte von Q ist $Qe_1 = Q_1 e_1$. Q_1 ist so bestimmt, daß $Q_1(Me_1) = \alpha e_1$, d.h. $Me_1 = \alpha Q_1 e_1$. Das bedeutet, daß $Q_1 e_1$ ein Vielfaches ist von

$$Me_1 = \begin{pmatrix} h_{1,1} - \mu \\ h_{21} \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

Aber $H = B^T B$, also $h_{1,1} = b_{1,1}^2$, $h_{1,2} = b_{2,1} \cdot b_{1,1}$ und $\mu = b_{n-1,n}^2 + b_{n,n}^2$. Dann ist:

$$Q_1 e_1 = \pm \frac{Me_1}{\|Me_1\|_2} = \begin{pmatrix} c \\ s \\ 0 \\ \ldots \\ 0 \end{pmatrix} \quad \text{mit } c^2 + s^2 = 1, \quad \text{also: } Q_1 = \left(\begin{array}{cc|c} c & s & \\ s & -c & 0 \\ \hline & 0 & I_{n-2} \end{array}\right)$$

Wir transformieren also $Q_1^T H Q_1$ auf Tridiagonalform und erhalten

$$\bar{H} = \tilde{Q}^T (Q_1^T H Q_1) \tilde{Q}, \qquad \tilde{Q} = \begin{pmatrix} 1 & 0 \\ 0 & \hat{Q} \end{pmatrix}$$

und $\tilde{Q}$ ist orthogonal. Dann ist $Q_1 \tilde{Q} e_1 = Q_1 e_1 = Q e_1$. Daher muß $Q_1 \tilde{Q} = Q$ sein, weil die erste Spalte übereinstimmt.

Weil $H = B^T B$ ist, ist $\bar{H} = Q^T B^T B Q = Q^T B^T P^T P B Q$ tridiagonal, falls PBQ bidiagonal ist.

Wir transformieren

$$BQ_1 = \begin{pmatrix} \ddots & \ddots & 0 & 0 \\ * & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots \\ 0 & 0 & 0 & \ddots \end{pmatrix}$$

auf Bidiagonalform, indem wir von Links mit einer Householdermatrix P_1 multiplizieren und dann von rechts mit einer Matrix Q_2 multiplizieren, usw. Wir verjagen dabei das „böse“ Element $*$, bis es rausfällt. Das Verfahren heißt in der Literatur auch *Zickzackjagd*.

2.71 Algorithmus (Golub/Kahan, 1965)
Sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$. Dann führen wir den folgenden Algorithmus durch:

1. Wir transformieren A mit Householder-Matrizen auf Bidiagonalform:

$$PAQ = \begin{pmatrix} B \\ 0 \end{pmatrix}, \qquad B = \begin{pmatrix} \ddots & \ddots & 0 \\ 0 & \ddots & \ddots \\ 0 & 0 & \ddots \end{pmatrix}, \qquad \beta = \|B\|_F := \sqrt{\sum_{i,j=1}^{n} b_{ij}^2}$$

2. Wir berechnen

$$Q_1 = \begin{pmatrix} \star & \star & 0 \\ \star & \star & \\ 0 & & I \end{pmatrix}, \qquad BQ_1 = \begin{pmatrix} \ddots & \ddots & 0 & 0 \\ \star & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots \\ 0 & 0 & 0 & \ddots \end{pmatrix}$$

Wir transformieren dann BQ_1 durch „Zickzackjagd“ auf Bidiagonalform $\bar{B}$.

Wir wiederholen (mit $\bar{B}$ statt B) solange, bis $|b_{n-1,n}| \leq \beta \cdot \texttt{eps}$ ist. Wir reduzieren dann die Dimension um eins und machen mit $(b_{ij})_{i,j=1}^{n-1}$ weiter, bis wir schließlich Diagonalform erreicht haben.

2.72 Bemerkung
Wegen Äquivalenz zum QR-Algorithmus für $H = B^T B$ erhalten wir dann kubische Konvergenz, weil diese Matrix symmetrisch und tridiagonal ist.

3 Verfahren der konjugierten Gradienten

3.1 Motivation
Das Verfahren der konjugierten Gradienten läßt sich anwenden, um Funktionen zu minimieren und große lineare Gleichungssysteme (die von einer symmetrischen, positiv definiten Matrix kommen) zu lösen.

3.2 Motivation (Minimierung)
Sei $f : \mathbb{R}^n \supseteq D \to \mathbb{R}$. Wir wollen das Minimum von f bestimmen, d.h. $f'(x) = 0$. Dies können wir über Newton-artige Verfahren berechnen (vgl. Numerik I).

Wir benötigen dann aber noch Kenntnisse der Hessematrix, was recht rechenaufwändig ist. Wir suchen im Folgenden ein Verfahren, welches die zweite Ableitung nicht benötigt. Das Verfahren der konjugierten Gradienten wird dies erfüllt.

3.3 Motivation (Lösung eines LGS)
Sei f quadratisch, d.h. $f(x) = \frac{1}{2}x^T Ax - x^T b$ mit $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit und $b \in \mathbb{R}^n$. Dann ist $f'(x) = 0 \iff Ax - b = 0 \iff Ax = b$ als Lösung eines LGS.

Das Verfahren der konjugierten Gradienten ist iterativ und braucht keine Matrixzerlegung, nicht einmal die Matrix A, sondern nur die Abbildung $x \mapsto Ax$.

In vielen Anwendungen (Lösung von Diskretisierungen von FE-Verfahren) treten oft sehr große Matrizen auf, die sehr dünn besetzt sind. Bei diesen ist es nicht angebracht, diese LGS mit einer Zerlegung zu lösen, da der Rechenaufwand zu groß ist und Nicht-Null-Elemente aufgefüllt werden.

3.1 Eindimensionale Minimierung

3.4 Algorithmus (Verfahren des goldenen Schnitts)
Sei $f : \mathbb{R} \to \mathbb{R}$. Wir suchen das Minimum in einem Intervall $[a, b]$. Wir wählen nun $v, w \in (a, b)$ so, daß gilt:

$$\frac{v-a}{w-a} = \frac{w-a}{b-a}, \qquad \frac{b-w}{b-v} = \frac{b-v}{b-a}$$

Dann ergibt sich durch Auflösen der Gleichungen:

$$v = a + \frac{3-\sqrt{5}}{2}(b-a), \qquad w = a + \frac{\sqrt{5}-1}{2}(b-a)$$

Dieses Verhältnis ist gerade der goldene Schnitt. Wir fahren nun fort:

1. Falls $f(v) < f(w)$, dann ist das neue Suchintervall $[a, w]$. Wir können berechnen, daß dort dann v anstelle von w zu nehmen ist und wir brauchen nur eine weitere Funktionsauswertung.
2. Falls $f(v) > f(w)$, dann ist das neue Suchintervall $[v, b]$ und w nimmt die Rolle von v ein.

Wir wiederholen diese Intervallreduktion, bis $b - a \leq$ `tol`, wobei mit a und b die Intervallgrenzen im k-ten Schritt sind.

3.5 Algorithmus (Quadratische Interpolation)
Wir legen eine Parabel durch 3 Punkte $(x_0, f(x_0)), (x_1, f(x_1))$ und $(x_2, f(x_2))$ mit $x_0 < x_1 < x_2$ und bestimmen deren Minimum x^*.

Wir nehmen als neues Intervall dann entweder $[x_0, x_1]$ oder $[x_1, x_2]$ und zwar jenes, welches x^* enthält.

Beim neuen Intervall fahren wir so iterativ fort und wiederholen die Iteration, bis die Intervallänge kleiner als eine vorgegebene Tolerenz `tol` ist.

3.2 Verfahren des steilsten Abstiegs (Gradientenverfahren)

3.6 Algorithmus (Gradientenverfahren)
Wir haben $f : \mathbb{R}^n \to \mathbb{R}$ und $f(x^*) = \min$ und ersetzen das n-dimensionale Minimierungsproblem durch eine Folge von eindimesionalen Minimierungsproblemen.

Wir haben einen Ausgangspunkt $x_0 \in \mathbb{R}^n$ und suchen nun eine Richtung, wie wir in x^* kommen. Die Suchrichtung $0 \neq d \in \mathbb{R}^n$ wählen wir so, daß dies lokal den steilsten Abstieg repräsentiert und minimieren dann $f(x_0 + \alpha d)$ als α^* (was über die eindimensionalen Verfahren geht) und setzen $x_1 := x_0 + \alpha^* d$.
Dies führen wir iterativ fort und erhalten, daß $x_k \to x^*$ konvergiert.

3.7 Beispiel
Sei $f(x) = \frac{1}{2}x^T Ax - x^T b$ mit A symmetrisch, positiv definit. Das Minimum von $f(x + \alpha d)$ wird bei

$$\alpha = -\frac{d^T(Ax - b)}{d^T Ad}$$

angenommen, was man leicht zeigen kann.

Wir beachten dabei, daß $\nabla f(x) = Ax - b$ ist.

3.8 Bemerkung (Wahl der Suchrichtung)
Mit der Taylorentwicklung erhalten wir:

$$f(x + \alpha d) = f(x) + \alpha f'(x) \cdot d + \mathcal{O}(\alpha^2)$$

Lokal ist also der steilste Abstieg in Richtung des negativen Gradienten von f, also:

$$d = -\nabla f(x)$$

3.9 Bemerkung
Lokal ist dieses Verfahren natürlich optimal, aber global brauchen wir viele Schritte, um zum Ziel zu kommen. Dazu betrachten wir ein anderes Verfahren.

3.10 Bemerkung (Konjugierte Gradienten)
Sei $g := Ax - b$. Dann betrachten wir den Vektorraum $V_k := \langle g, Ag, \ldots, A^{k-1}g \rangle$ und wir minimieren im k-ten Schritt für den Startvektor x_0:

$$f(x_k) = \min_{d \in V_k} f(x_0 + d)$$

3.11 Motivation (Ausblick)
Wir werden sehen, daß sich die x_k rekursiv aus eindimensionalen Minimierungen berechnen lassen.

3.3 Ritz-Galerkin-Verfahren

3.12 Motivation (Problemstellung)
Sei $f : V \to \mathbb{R}$ und V ein reeller (evtl. unendlich-dimensionaler) Vektorraum und sei:

$$f(v) = \frac{1}{2}a(v, v) - b(v)$$

wobei $a : V \times V \to \mathbb{R}$ eine symmetrische, positiv definite Bilinearform und $b : V \to \mathbb{R}$ eine Linearform ist.

Wir möchten das Minimum von f bestimmen.

3.13 Bemerkung
Falls $V = \mathbb{R}^n$ ist, dann ist $a(v,v) = \frac{1}{2}v^T Av$ und $b(v) = v^T\tilde{b}$ mit A symmetrisch, pos. def. und $\tilde{b} \in \mathbb{R}^n$. Hier gilt, wie wir bereits wissen: $f(v) = \min \iff Av = b$.

3.14 Satz (Charakterisierung der Minimierung)
Für $u \in V$ gilt:

$$f(u) = \min_{v \in V} f(v) \iff a(u,v) = b(v) \quad \forall v \in V$$

Beweis
Wir setzen $v := u + \lambda w$ für $\lambda \in \mathbb{R}$ und $w \in V$. Dann ist

$$f(u+\lambda w) - f(u) = \frac{1}{2}\Big(a(u+\lambda w, u+\lambda w) - a(u,u)\Big) - \Big(b(u+\lambda w) - b(u)\Big)$$

Mit der (Bi)Linearität:

$$\begin{aligned} &= \frac{1}{2}\Big(a(u,u) + 2\lambda a(u,w) + \lambda^2 a(w,w) - a(u,u)\Big) \\ &\quad - (b(u) + \lambda \cdot b(w) - b(u)) \\ &= \lambda\big(a(u,w) - b(w)\big) + \frac{1}{2}\lambda^2 a(w,w) \end{aligned}$$

Also ist $f(u) \leq f(u+\lambda w) \quad \forall \lambda \in \mathbb{R}\ \forall w \in V$ (Dies bedeutet, daß u das Minimum ist) äquivalent zu $a(u,w) = b(w) \quad \forall w \in V$. □

3.15 Idee (Approximation des Minimums)
Wir wählen einen k-dimensionalen Unterraum $V_k \subseteq V$ und bestimmen das Minimum dieses k-dimensionalen Unterraums für $u_k \in V_k$ so, daß

$$f(u_k) = \min_{v_k \in V_k} f(v_k)$$

Dies ist bekannt als der *Ritz-Ansatz*.

Nach dem oben bewiesenen Satz ist dies aber äquivalent dazu, einen Vektor $u_k \in V_k$ so zu bestimmen, daß gilt:

$$a(u_k, v_k) = b(v_k) \quad \forall v_k \in V_k$$

was als der *Galerkin-Ansatz* bekannt ist.

3.16 Bemerkung (Anwendungen)
Diese Ansätze sind in den folgenden Beispielen sehr wichtig:

1. Zur Lösung von $Ax = b$ für A eine symmetrische, positiv definiten Matrix und A endlichdimensional und schwach besetzt. Als Approximationsraum nehmen wir hierbei oft $V_k := \langle g, Ag, \ldots, A^{k-1}g \rangle$. Über diesen Raum werden wir im nächsten Kapitel sprechen, wenn wir das Verfahren der konjugierten Gradienten besprechen.

   Besonders gut wird dieser Ansatz, wenn g ein Eigenvektor von A ist.

2. Finite Elemente, was ein wichtiges Verfahren zur Lösung von PDE ist. Auf dieses wird in Numerik für partielle Differentialgleichungen besprochen.

3.17 Satz (Existenz und Eindeutigkeit der Minimierung in Unterraum)
Es existiert genau eine Lösung $u_k \in V_k$ vom Minimierungsproblem

$$f(u_k) = \min_{v_k \in V_k} f(v_k)$$

Beweis
Sei $(\varphi_1, \ldots, \varphi_k)$ eine Basis von V_k. Wir suchen nun

$$u_k = \sum_{i=1}^{k} \mu_i \varphi_i \in V_k$$

so daß $a(u_k, v_k) = b(v_k)$ für alle v_k, die darstellbar sind mit $v_k = \sum_{j=1}^{k} \nu_j \varphi_j$, d.h.

$$\sum_{i=1}^{k} \sum_{j=1}^{k} \mu_i \nu_j a(\varphi_i, \varphi_j) = \sum_{j=1}^{k} \nu_j b(\varphi_j) \quad \forall \nu := (\nu_j) \in \mathbb{R}^k$$

Wenn wir setzen $A := a(\varphi_i, \varphi_j)$, $\mu := (\mu_i)$ und $\ell := b(\varphi_j)$, dann müssen wir in dieser Schreibweise also folgendes Problem lösen:

$$\nu^T A \mu = \nu^T \ell \quad \forall \nu \in \mathbb{R}^k$$

Da dies für alle ν gelten muß, muß dies insbesondere für $\nu = (e_i)$ gelten und wir erhalten damit das neue LGS $A\mu = \ell$.

Wir rechnen nun nach, daß A positiv definit ist:

$$\nu^T A \nu = \sum_{i,j=1}^{k} \nu_i \nu_j a(\varphi_i, \varphi_j) = a\left(\sum_{i=1}^{k} \nu_i \varphi_i, \sum_{j=1}^{k} \nu_j \varphi_j \right) > 0$$

falls $\sum_{i=1}^{k} \nu_i \varphi_i \neq 0$, also $\nu = (\nu_i)_{i=1}^{k} \neq 0$, weil φ eine Basis ist.

Also ist A positiv definit (und klarerweise sogar symmetrisch) und damit insbesondere invertierbar. Damit folgt die Behauptung. □

3.18 Satz (Lemma von Cea)
Falls $u \in V$ eine Lösung des Minimierungsproblems auf V ist und $u_k \in V_k$ eine Lösung des Unterraums V_k ist, so gilt in der Energienorm $\|v\|_a := \sqrt{a(v,v)}$, daß

$$\|u_k - u\|_a = \min_{v_k \in V_k} \|v_k - u\|_a$$

d.h. u_k ist in diesem Sinne die bestmögliche Approximation an u.

Die Ritz-Galerkin-Approximation hat also unter allen Elementen des Approximationssraums V_k den kleinsten Abstand (bezüglich der Energienorm) zur exakten Lösung.

Beweis
V_k ist ein Unterraum von V, also gilt für alle $v_k \in V$, daß $a(u, v_k) = b(v_k)$ und $a(u_k, v_k) = b(v_k)$. Wir subtrahieren beide Gleichungen voneinander und erhalten mit der Bilinearität:

$$a(u_k - u, v_k) = 0 \quad \forall v_k \in V_k$$

Da wir v_k durch $u_k - v_k \in V_k$ ersetzen dürfen und eine Null einfügen, gilt damit auch:

$$a(u_k - u, u_k - u + u - v_k) = 0 \iff a(u_k - u, u_k - v) = a(u_k - u, v_k - u) \quad \forall v_k \in V_k$$

Mit der Definition der Energienorm, der obigen Gleichung und anschließend Cauchy-Schwarz erhalten wir also:

$$\|u_k - u\|_a^2 = a(u_k - u, u_k - u) \leq \|u_k - u\|_a \cdot \|v_k - u\|_a \quad \forall v_k \in V_k$$

Also gilt:

$$\|u_k - u\|_a \leq \|v_k - u\|_a \quad \forall v_k \in V_k$$

□

3.4 Das Verfahren der konjugierten Gradienten

3.19 Motivation
Wir wollen in diesem Kapitel stets voraussetzen, daß wir ein LGS $Ax = b$ lösen möchten, wobei $A \in \mathbb{R}^{n \times n}$ symmetrisch, positiv definit ist und $b \in \mathbb{R}^n$. Typischerweise ist n sehr groß und A schwach besetzt, wie es etwa in der Anwendung bei der Lösung von partiellen Differentialgleichungen vorkommt.

Wir wissen bereits, daß dies äquivalent zur Minimierung von $f(x) = \frac{1}{2}x^T Ax - x^T b$ ist.

Das Verfahren der konjugierten Gradienten ist in der Literatur oft als cg-Verfahren bekannt (conjugate gradient method).

3.20 Historische Notiz
Dieses Verfahren haben HESTENES und STIEFEL 1952 vorgestellt.

3.21 Definition (Krylov-Raum)
Aus dem Gradientenverfahren kennen wir bereits den Raum $V_k := \langle g_0, Ag_0, \ldots, A^{k-1}g_0\rangle$ mit $g_0 := Ax_0 - b = \nabla f(x_0)$ für einen Startvektor x_0.

Den Vektorraum V_k nennen wir den *k-ten Krylovraum.*

3.22 Idee (Ritz-Galerkin-Ansatz)
Wir verwenden nun jetzt das Ritz-Galerkin-Ansatz auf den Approximationsraum V_k, d.h. wir suchen ein $x_k \in x_0 + V_k$ mit

$$f(x_k) = \min_{v_k \in V_k} f(x_0 + v_k)$$

3.23 Bemerkung
Da dieser Raum im Allgemeinen k-dimensional ist, ist es schwer, solche Minima zu berechnen. Wir werden aber im Folgenden zeigen können, daß es hinreichend ist, k eindimensionale Probleme rekursiv zu lösen, um an das Minimum in V_k zu kommen.

Für die Herleitung nehmen wir o.B.d.A. an, daß $x_0 = 0$ ist (Betrachte sonst $y = x - x_0$; dann wäre $Ay = b - Ax_0$ für den Startwert $y_0 = 0$ zu lösen).

3.24 Bemerkung (von A induziertes Skalarprodukt)
Es ist $\langle u, v\rangle_A := u^T Av = \langle Au, v\rangle$, wobei $\langle \cdot, \cdot \rangle$ das Euklidische Skalarprodukt ist.

3.25 Idee (Zerlegung des Krylov-Raums)
Wir zerlegen den Krylovraum auf den folgende Art und Weise:

$$V_{k+1} = V_k \oplus V_k^\perp$$

mit $V_k^\perp = \{w \in V_{k+1} | w^T Av = 0 \quad \forall v \in V_k\}$, der in der Literatur auch oft als der zu V_k konjugierte (A-orthogonale) Unterraum von V_{k+1} ist und natürlich per Konstruktion orthogonal bezüglich das von A induzierten Skalarprodukts ist.

3.26 Konstruktion (Herleitung des Algorithmus)
Es ist klar, daß $\dim V_k^\perp \leq 1$.

Damit können wir eindeutig schreiben: $x_{k+1} = u + w$ mit $u \in V_k$ und $w \in V_k^\perp$. Also ist x_{k+1} bestimmt durch die Bedingungen, daß

$$x_{k+1} \in V_{k+1}, \qquad f(x_{k+1}) = \min_{v_{k+1} \in V_{k+1}} f(v_{k+1})$$

bzw. dies ist äquivalent dazu, daß wir ein $v_{k+1} \in V_{k+1}$ suchen so, daß

$$\langle Ax_{k+1}, v_{k+1}\rangle = \langle b_k, v_{k+1}\rangle \quad \forall v_{k+1} \in V_{k+1}$$

Wegen $V_k \leq V_{k+1}$ ist insbesondere auch:

$$\langle Ax_{k+1}, v_k \rangle = \langle b, v_k \rangle \quad \forall v_k \in V_k$$

Aber die linke Seite ist wegen der Orthogonaliät:

$$\langle Ax_{k+1}, v_k \rangle = \langle Au, v_k \rangle + \underbrace{\langle Aw, v_k \rangle}_{=0}$$

Andererseits ist $x_k \in V_k$ eindeutig bestimmt durch:

$$\langle Ax_k, v_k \rangle = \langle b, v_k \rangle \quad \forall v_k \in V_k$$

Damit ist $u = x_k$. Wir erhalten also, daß $x_{k+1} = x_k + w$ mit $w \in V_k^\perp$ bzw. für d_k (z.B. eine Basis von $V_k^\perp$) mit $V_k^\perp = \mathbb{R}d_k$. Damit gilt für ein $\alpha \in \mathbb{R}$:

$$x_{k+1} = x_k + \alpha d_k$$

x_{k+1} ist also die Lösung eines höchstens eindimensionalen Minimierungsproblems:

$$f(x_{k+1}) = \min_{\alpha \in \mathbb{R}} f(x_k + \alpha d_k)$$

Wir müssen also das Minimum einer quadratischen Funktion finden, wofür wir bereits aus dem zweiten Kapitel ein optimales α kennen, und zwar:

$$\alpha_k = -\frac{\langle d_k, g_k \rangle}{\langle Ad_k, d_k \rangle}$$

3.27 Konstruktion (Bestimmung der Suchrichtung)
Wir möchten die Suchrichtung d_k als einen Basisvektor von $V_k^\perp$ nun bestimmen. Hierzu betrachten wir verschiedene Fälle:

1. $V_{k+1} = V_k$. Dann ist $V_k^\perp = 0$ und damit ist $d_k = 0$. Also gilt für $x_k \in V_k$, daß auch $Ax_k \in V_{k+1} = V_k$. Aber es gilt:

   $$\langle Ax_k, v_k \rangle = \langle b, v_k \rangle \quad \forall v_k \in V_k, \qquad \langle Ax_k - b, v_k \rangle = 0 \quad \forall v_k \in V_k$$

   Weil aber $Ax_k - b \in V_k$ ist die einzige Lösung der Nullvektor in der ersten Komponente des Skalarprodukts, d.h. es ist $Ax_k = b$. Also ist x_k die Lösung des LGS.

2. Sei $\dim V_k^\perp = 1$. Dann ist $g_k = Ax_k - b \neq 0$, also ist $g_k \notin V_k$, aber $g_k \in V_{k+1}$.

   Wir zerlegen nun $g_k = c_k + d_k$ mit $c_k \in V_k$ und $d_k \in V_k^\perp$, insbesondere ist $d_k \neq 0$. c_k ist also die A-orthogonale Projektion von g_k auf V_k bezüglich des von A induzierten Skalarprodukts.

   Sei nun $\{d_0, d_1, \ldots, d_{k-1}\}$ eine ONB (bezüglich A) von V_k, d.h. es gilt $\langle Ad_i, d_j\rangle = 0$ für $i \neq j$. Aufgrund des Gram-Schmidt-Verfahrens gilt also für c_k:

   $$c_k = \sum_{i=0}^{k-1} \frac{\langle Ad_i, g_k\rangle}{\langle Ad_i, d_i\rangle} d_i$$

   Damit gilt:

   $$d_k = g_k - c_k = g_k - \sum_{i=0}^{k-1} \frac{\langle Ad_i, g_k\rangle}{\langle Ad_i, d_i\rangle} d_i$$

   Weil $Ad_j \in V_k$, solange $j \leq k-2$ ist, ist wegen der Definition von x_k:

   $$\langle g_k, Ad_j\rangle = \langle Ax_k - b, Ad_j\rangle = 0$$

   Damit vereinfacht sich unsere Summe sehr stark und wir erhalten letztendlich:

   $$d_k = g_k - \frac{\langle Ad_{k-1}, g_k\rangle}{\langle Ad_{k-1}, d_{k-1}\rangle} d_{k-1}$$

3.28 Bemerkung (Rechenaufwand der Iteration)
Wir möchten $g_{k+1} = Ax_{k+1} - b$ berechnen.

Allerdings ist der Aufwand der Matrix-Multiplikation zu hoch. Aufgrund der Definition von x_{k+1} gilt:

$$g_{k+1} = A(x_k + \alpha_k d_k) - b = (Ax_k - b) + \alpha_k Ad_k = g_k + \alpha_k Ad_k$$

Das Produkt Ad_k bräuchten wir sowieso beim nächsten Schritt für die Berechnungn der neuen Suchrichtung d_{k+1}, d.h. diese Berechnung ist ziemlich optimal im Rechenaufwand.

3.29 Algorithmus (cg-Verfahren)
Sei $x_0 \in \mathbb{R}^n$ ein belieger Startvektor. Dann ist $d_0 = g_0 = Ax_0 - b$.
Für $k = 0, 1, 2, \ldots$ ist dann (bis $\|Ax_k - b\| \leq \texttt{tol} \cdot \|b\|$) die folgende Rekursion zu machen:

$$x_{k+1} = x_k - \alpha_k d_k$$
$$g_{k+1} = g_k - \alpha_k Ad_k$$
$$d_{k+1} = g_{k+1} - \beta_k d_k$$

mit den Koeffizienten:

$$\alpha_k = \frac{\langle d_k, g_k\rangle}{\langle Ad_k, d_k\rangle}$$
$$\beta_k = \frac{\langle Ad_k, g_{k+1}\rangle}{\langle Ad_k, d_k\rangle}$$

Nach Konstruktion ist dann

$$f(x_k) = \min_{v_k \in V_k} f(x_0 + v_k)$$

3.30 Bemerkung (Rechenaufwand)
Wir benötigen pro Schritt nur eine einzige Multiplikation mit der Matrix und drei Skalarprodukte.

3.31 Bemerkung
Wir werden in einer Übungsaufgabe zeigen können, daß wir schreiben können:

$$\alpha_k = \frac{\langle g_k, g_k\rangle}{\langle Ad_k, d_k\rangle}, \qquad \beta_k = -\frac{\langle g_{k+1}, g_{k+1}\rangle}{\langle g_k, g_k\rangle}$$

Damit bräuchten wir pro Schritt nur zwei verschiedene Skalarprodukte.

3.5 Fehleruntersuchung des cg-Verfahrens

3.32 Satz (Maximale Schrittweite)
Nach höchstens n Schritten des cg-Verfahrens erhalten wir die exakte Lösung des LGS $Ax^* = k$, d.h. $\exists k \leq n$ so, daß $x_k = x^*$.

Beweis
Es ist $V_1 \leq V_2 \leq \ldots \leq \mathbb{R}^n$, also gibt es ein $k \leq n$ mit $V_{k+1} = v_k$. Also ist x_k exakt. □

3.33 Bemerkung
Für die praktische Anwendung ist dieser Satz uninteressant, da wir dieses Verfahren meistens für sehr große n benutzen. Außerdem genügt uns bereits eine sehr gute Näherung an die Lösung, also z.B. $\|x_k - x\| \leq \texttt{tol} \cdot \|x_0 - x\|$ für $k \ll n$. Im Übrigen können Rundungsfehler das Ergebnis jedem Schritt schlechter machen.

3.34 Satz (Fehlerabschätzung beim cg-Verfahren)
Es gilt für alle q_k Polynom mit $\deg q_k \leq k$ und $q_k(0) = 1$:

$$\|x_k - x^*\|_A \leq \left(\max_{\lambda \text{ EW von } A} |q_k(\lambda)| \right) \|x_0 - x^*\|_A$$

wobei x^* die exakte Lösung ist.

3.35 Bemerkung (Beobachtung)
Sei A eine Matrix beliebiger Dimension, mit genau k verschiedenen Eigenwerte.
Dann liefert das cg-Verfahren die exakte Lösung nach spät. k Schritten.

Wir nehmen dazu das Polynom, welches die Eigenwerte als Nullstellen hat.
Zudem geht das cg-Verfahren schnell, wenn die Eigenwerte geclustert auftreten.

Beweis
Wir nehmen o.B.d.A. an, daß $x_0 = 0$ ist (sonst betrachten wir $y = x - x_0$ und für $Ay = b - Ax_0$ hätten wir $y_0 = 0$).

1. Wir wissen nach dem Satz von Cea, daß

   $$\|x_k - x\|_A = \min_{v_k \in V_k} \|v_k - x\|_A$$

   Für ein $V_k \ni v_k = p_{k-1}(A)b$ mit $\deg p_{k-1} \leq k-1$, gilt nun wegen $Ax^* = b$:

   $$x^* - v_k = \underbrace{(I - p_{k-1}(A) \cdot A)}_{=:q_k(A)} x^*$$

   Wir sehen, daß q_k ein Polynom vom Grad $\deg q_k \leq k$ mit $q_k(0) = 1$ und rechnen nun:

   $$\|x_k - x^*\|_A = \min_{v_k \in V_k} \|v_k - x^*\|_A = \min_{\deg q_k \leq k, q_k(0)=1} \|q_k(A)x^*\|_A$$

2. Wir wissen, daß A symmetrisch und positiv definit ist, also können wir A durch eine Orthogonalmatrix diagonalisieren mit $A = Q^T \Lambda Q$ mit Q orthogonal und $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Wegen positiver Definitheit sind die $\lambda_i > 0$. Wir rechnen nun:

   $$\begin{aligned}
   \|q_k(A)x^*\|_A^2 &= \langle Aq_k(A)x^*, q_k(A)x^* \rangle = \langle Q^T \Lambda Q Q^T q_k(\Lambda) Q x^*, Q^T q_k(\Lambda) Q x^* \rangle \\
   &= \langle \Lambda q_k(\Lambda) Q x^*, q_k(\Lambda) Q x^* \rangle = \langle \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} q_k(\Lambda) Q x^*, q_k(\Lambda) Q x^* \rangle \\
   &= \left\langle q_k(\Lambda) \Lambda^{\frac{1}{2}} Q x^*, q_k(\Lambda) \Lambda^{\frac{1}{2}} Q x^* \right\rangle = \left\| q_k(\Lambda) \Lambda^{\frac{1}{2}} Q x^* \right\|_2^2 \\
   &\leq \underbrace{\|q_k(\Lambda)\|_2^2}_{=(\max_{i=1,\ldots,n} |q_k(\lambda_i)|)^2} \cdot \|\Lambda^{\frac{1}{2}} Q x^*\|_2^2
   \end{aligned}$$

   Wir müssen nun noch einen geschickten Term für $\|\Lambda^{\frac{1}{2}} Q x^*\|_2^2$ finden:

   $$\|\Lambda^{\frac{1}{2}} Q x^*\|_2^2 = \langle \Lambda^{\frac{1}{2}} Q x^*, \Lambda^{\frac{1}{2}} Q x^* \rangle = \langle \Lambda Q x^*, Q x^* \rangle = \langle Q^T \Lambda Q x^*, x^* \rangle = \langle Ax^*, x^* \rangle = \|x^*\|_A^2$$

Damit erhalten wir schließlich:

$$\|x_k - x^*\|_A \leq \|q_k(A)x^*\|_A \leq \max_{i=1,\dots,n} |q_k(\lambda_i)| \|x^*\|_A$$

□

3.36 Wiederholung

1. Für die Konditionszahl bzgl. der euklidischen Norm für eine symmetrisch positive Matrix gilt: $\kappa := \text{cond}_2(A) := \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$.
2. Für alle Polynome vom Grad k ist das Tschebyscheff-Polynom das minimalste, denn angenommen, es gelte $|p_k(t)| < \frac{1}{|T_k(t_0)|}$.
   Dann gilt für mindestens k Punkte aus $[-t, t]$, daß $p_k(t) = \frac{T_k(t)}{T_k(t_0)}$, aber es ist auch $p_k(t_0) = \frac{T_k(t_0)}{T_k(t_0)} = 1$. Weil aber $\deg p_k \leq k$, muß gelten, daß $p_k(t) = \frac{T_k(t)}{T_k(t_0)}$ ist.

3.37 Satz (Konditionszahlsabhängige Fehlerabschätzung)
Es gilt für die Konditionszahl $\kappa := \text{cond}_2(A)$:

$$\|x_k - x\|_A \leq 2 \cdot \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \cdot \|x_0 - x\|_A$$

Beweis
Seien alle Eigenwerte von A in $[\alpha, \beta]$ mit $0 < \alpha < \beta$ optimal, dann ist $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\beta}{\alpha}$. Nach dem letzten Satz gilt für alle Polynome q_k mit $\deg q_k \leq k$ und $q_k(0) = 1$:

$$\|x_k - x\|_A \leq \max_{\lambda \text{ EW von } A} |q_k(\lambda)| \cdot \|x_0 - x\|_A$$

Wir transformieren nun $[\alpha, \beta]$ auf $[-1, 1]$. Dies geschieht durch die Abbildung

$$\lambda \mapsto t = \frac{2}{\beta - \alpha} \cdot \lambda + \frac{\beta + \alpha}{\beta - \alpha}, \quad \text{inbesondere: } 0 \mapsto -\frac{\beta + \alpha}{\beta - \alpha} = -\frac{\kappa + 1}{\kappa - 1} =: t_0 < -1$$

Wir erhalten damit dann die folgende Abschätzung:

$$\|x_k - x\|_A \leq \max_{t \in [-1,1]} |p_k(t)| \cdot \|x_0 - x\|_A$$

für alle Polynome p_k mit $\deg p_k \leq k$ und $p_k(t_0) = 1$. Aus Numerik I wissen wir, daß die rechte Seite der Abschätzung minimal wird für das Tschebyscheff-Polynom $\frac{T_k(t)}{T_k(t_0)}$ und damit gilt:

$$\|x_k - x\|_A \leq \frac{1}{|T_k(t_0)|} \cdot \|x_0 - x\|_A$$

Wir benutzen nun, daß für ein Tschebyscheff-Polynom gilt:

$$T_k(t) = \frac{1}{2}\left((t+\sqrt{t^2-1})^k + (t+\sqrt{t^2-1})^{-k}\right) \Rightarrow |T_k(t_0)| = T_k(|t_0|) \geq \frac{1}{2}\left(|t_0| + \sqrt{t_0^2-1}\right)^k$$

Wir rechnen hiermit:

$$\begin{aligned} |t_0| + \sqrt{t_0^2-1} &= \frac{\kappa+1}{\kappa-1} + \sqrt{\frac{\kappa^2+2\kappa+1-(\kappa^2-2\kappa+1)}{(\kappa-1)^2}} = \frac{1}{\kappa-1}(\kappa+1+2\sqrt{\kappa}) \\ &= \frac{(\kappa+1)^2}{(\kappa+1)(\kappa-1)} = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \end{aligned}$$

Wir erhalten damit schließlich:

$$|T_k(t_0)| \geq \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k, \quad \text{also: } \|x_k - x\|_a \leq 2 \cdot \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_0 - x\|_A$$

□

3.38 Bemerkung

1. Für die Methode des steilsten Abstiegs gilt die entsprechende Aussage:

   $$\|x_k - x\|_A \leq \left(\frac{\kappa-1}{\kappa+1}\right)^k \|x_0 - x\|_A$$

   Für eine große Konditionszahl ist dieser Ausdruck viel näher bei Eins als beim ersten Ausdruck, d.h. wir haben eine sehr langsame Konvergenz.

2. Eine rasche Fehlerreduktions im cg-Verfahren tritt auf, falls

   a) die Kondition klein ist oder

   b) die Eigenwerte von A in einigen wenigen Clustern zusammenliegen.

3.6 Vorkonditioniertes cg-Verfahren

3.39 Idee

Wir möchten nun $Ax = b$ bestimmen für A symmetrisch und positiv definit. In der Regel ist A sehr groß und schwach besetzt.

Falls B symmetrisch und positiv definit ist, dann bestimmen wir $B \approx A^{-1}$ so, daß $x \mapsto Bx$ einfach zu berechnen ist. Dann haben wir ein äquivalentes LGS $BAx = Bb$. Wir fordern nun noch $\text{cond}_2(BA) \ll \text{cond}_2(A)$, damit das cg-Verfahren schneller konvergiert.

3.40 Bemerkung (Wahl des Vorkonditionierers)

Wir haben das Problem, B mit den Forderungen zu erfüllen. Daher führen wir eine unvollständige Cholesky-Zerleung durch:

Für $k = 1, \ldots, n$ berechnen wir:

$$\tilde{\ell}_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} \ell_{kj}^2}$$

und für $i = k+1, \ldots, n$ berechnen wir:

$$\tilde{\ell}_{ik} = \begin{cases} \frac{1}{\ell_{kk}} a_{ik} - \sum_{j=1}^{i-1} \ell_{ij}\ell_{kj}, & a_{ik} \neq 0 \\ 0, & a_{ik} = 0 \end{cases}$$

Wir summieren damit nur für die j mit $a_{ij} \neq 0$. Damit bleibt die Besetzungsstruktur von A erhalten. Wir haben $A = \tilde{L}\tilde{L}^T + R$, wobei R eine Restmatrix ist. $\tilde{L}\tilde{L}^T$ ist symmetrisch und positiv definit und der Kandidat für B. Wir füllen nicht mit Nichtnullstellen auf und dies bewirkt im Allgemeinen, daß die Eigenwerte von BA in Clustern liegen, womit das cg-Verfahren schnell konvergiert.

Insbesondere ist die Abbildung $x \mapsto Bx$ relativ einfach zu berechnen, da nur schwach besetzte Dreieckssysteme zu lösen sind.

3.41 Bemerkung
Die Schwierigkeit ist, daß BA nicht symmetrisch ist, selbst wenn A, B symmetrisch und positiv definit sind. Wir versuchen einen Ausweg zu finden:

Falls wir $B = CC^T$ schreiben können (z.B. durch die vollständige Cholesky-Zerlegung), dann ist:

$$CC^T Ax = CC^T b \iff C^T ACC^{-1}x = C^T b \iff \tilde{A} \cdot \tilde{x} = \tilde{b}$$

mit $\tilde{A} := C^T AC$ wieder symmetrisch, positiv definit, worauf wir das cg-Verfahren anwenden können.

3.42 Konstruktion (Formale Anwendung auf das cg-Verfahrens)
Wir nehmen den Startwert $\tilde{x}_0 = C^{-1}x_0$ in den Algorithmus wie in Kapitel 4 anwenden auf alle Größen mit Tilde. Wir zeigen, daß wir dies durchführen können, ohne C zu kennen. Dazu rechnen wir:

$$\tilde{x}_{k+1} = \tilde{x}_k - \alpha_k \tilde{d}_k \iff C^{-1}x_{k+1} = C^{-1}x_k - \alpha_k C^{-1}d_k$$

wobei $d_k := C\tilde{d}_k$ gesetzt werden muß. Multiplikation mit C liefert dann:

$$x_{k+1} = x_k - \alpha_k d_k$$

Wir drücken nun den $\tilde{g}_k$ aus durch:

$$\tilde{g}_k = \tilde{A}\tilde{x}_k - \tilde{b} = C^T ACC^{-1}x_k - C^T b = C^T(Ax_k - b) = C^T g_k$$

Damit gilt mit dem Algorithmus des cg-Verfahrens:

$$\tilde{g}_{k+1} = \tilde{g}_k - \alpha_k \tilde{A}\tilde{d}_k \iff C^T g_{k+1} = C^T g_k - \alpha_k C^T A C C^{-1} d_k \iff g_{k+1} = g_k - \alpha_k A d_k$$

Die Bestimmung der g_k bleibt also wie gehabt; für die Suchrichtungen gilt:

$$\tilde{d}_{k+1} = \tilde{g}_{k+1} - \beta_k \tilde{d}_k \iff C^{-1} d_{k+1} = C^T g_{k+1} - \beta_k C^{-1} d_k \iff d_{k+1} = B g_{k+1} - \beta_k d_k$$

Wir zeigen nun noch, daß in den Koeffizienten kein C vorkommt:

$$\alpha_k = \frac{\langle \tilde{g}_k, \tilde{g}_k \rangle}{\langle \tilde{A}\tilde{d}_k, \tilde{d}_k \rangle} = \frac{\langle C^T g_k, C^T g_k \rangle}{\langle C^T A C C^{-1} d_k, C^{-1} d_k \rangle} = \frac{\langle g_k, C C^T g_k \rangle}{\langle A d_k, d_k \rangle} = \frac{\langle g_k, B g_k \rangle}{\langle A d_k, d_k \rangle}$$

$$\beta_k = -\frac{\langle \tilde{g}_{k+1}, \tilde{g}_{k+1} \rangle}{\langle \tilde{g}_k, \tilde{g}_k \rangle} = -\frac{\langle g_{k+1}, B g_{k+1} \rangle}{\langle g_k, B g_k \rangle}$$

3.43 Algorithmus (vorkonditioniertes cg-Verfahren)
Wir möchten $Ax = b$ lösen mit A symmetrisch, positiv definit und verwenden B symmetrisch positiv definit mit $B \approx A^{-1}$.

Wir wählen $x_0 \in \mathbb{R}^n$ beliebig und $d_0 = g_0 = Ax_0 - b$, $\varrho_0 = \langle Bg_0, g_0 \rangle$ und dann iterieren wir für $k = 0, 1, 2, \ldots$ (bis $\sqrt{\varrho_k} \leq \texttt{tol} \cdot \|b\|_2$):

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k d_k \\ g_{k+1} &= g_k - \alpha_k A d_k \\ d_{k+1} &= B g_{k+1} + \beta_k d_k \end{aligned}$$

mit den Koeffizienten:

$$\alpha_k = \frac{\varrho_k}{\langle A d_k, d_k \rangle}, \qquad \beta_k = \frac{\varrho_{k+1}}{\varrho_k}, \qquad \varrho_{k+1} = \langle B g_{k+1}, g_{k+1} \rangle$$

3.44 Bemerkung (Implementierung)
Wir speichern für das Verfahren lediglich 5 Vektoren: (x, g, Bg, d, Ad).

Weiter beachten wir, daß wir nicht die Matrizen A, B selbst brauchen, sondern benötigen nur die Abbildungen $u \mapsto Au$ und $v \mapsto Bv$.

3.45 Satz
Es gelte $0 < \gamma \langle v, B^{-1} v \rangle \leq \|v\|_A \leq \Gamma \langle v, B^{-1} v \rangle$ für alle $v \in \mathbb{R}^n$.

Dann gilt:

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\tilde{\kappa}} - 1}{\sqrt{\tilde{\kappa}} + 1} \right)^k \|x_0 - x^*\|_A$$

mit

$$\tilde{\kappa} = \text{cond}_2(BA) \leq \frac{\Gamma}{\gamma}$$

Beweis

1. Wir rechnen zunächst wegen $\tilde{A} = C^T AC$:

$$\begin{aligned}\|x_k - x^*\|_A^2 &= \langle x_k - x^*, A(x_k - x^*)\rangle = \langle C(\tilde{x}_k - \tilde{x}^*), AC(\tilde{x}_k - \tilde{x}^*)\rangle \\ &= \langle \tilde{x}_k - \tilde{x}^*, \tilde{A}(\tilde{x}_k - \tilde{x})^*\rangle = \|\tilde{x}_k - \tilde{x}^*\|_{\tilde{A}}^2\end{aligned}$$

   Mit dem Satz über die konditionsabhängige Fehlerabschätzung auf das Tilde-System erhalten wir sofort die Abschätzung für unseren Satz mit $\tilde{\kappa} = \text{cond}_2(\tilde{A})$.

2. Mit der Definition der Kondition gilt:

$$\text{cond}_2(\tilde{A}) = \sqrt{\frac{\lambda_{\max}(\tilde{A}^T\tilde{A})}{\lambda_{\min}(\tilde{A}^T\tilde{A})}}, \qquad \text{cond}_2(BA) = \sqrt{\frac{\lambda_{\max}((BA)^T(BA))}{\lambda_{\min}((BA)^T(BA))}}$$

   Um die Kondition von BA zu berechnen, berechnen wir zunächst:

$$\begin{aligned}(BA)^T BA = ABBA &= ACC^T CC^T A = (C^T)^{-1}C^T ACC^T CC^T ACC^{-1} \\ &= (C^T)^{-1}\tilde{A}C^T \cdot C\tilde{A}C^{-1} = (C\tilde{A}C^{-1})^T(C\tilde{A}C^{-1})\end{aligned}$$

   Aber dies hat genau die gleichen Eigenwerte wie $\tilde{A}$ aufgrund des Basiswechsels. Damit stimmt $\text{cond}_2(BA) = \text{cond}_2(\tilde{A})$.

3. Es gilt klarerweise:

$$\lambda_{\min}(\tilde{A})\langle v, B^{-1}v\rangle \leq \langle v, Av\rangle \leq \lambda_{\max}(\tilde{A})\langle v, B^{-1}v\rangle \quad \forall v \in \mathbb{R}^n$$

   und damit gilt $\gamma \leq \lambda_{\min}(\tilde{A})$ und $\Gamma \geq \lambda_{\max}(\tilde{A})$. □

3.7 cg-Verfahren zur Minimierung nichtquadratischer Funktionen

3.46 Motivation
Wir wollen im Folgenden eine Funktion $f : \mathbb{R}^n \to \mathbb{R}$ betrachten und ein (lokales) Minimum von f suchen, wobei f im Allgemeinen nicht quadratisch ist.

3.47 Wiederholung
Aus dem Analysis-Grundstudium wissen wir, daß $x^* \in \mathbb{R}^n$ ein Minimum lokales Minimum von f ist (mit f zwei mal stetig diff'bar), dann ist $g(x^*) = \nabla f(x^*) = 0$ und die Hessematrix, $H(x^*) = \nabla^2 f(x^*)$ ist symmetrisch und positiv semidefinit.

Ist umgekehrt die $\nabla f(x^*) = 0$ und die Hessematrix positiv definit, dann ist x^* ein lokales Minimum.

3.48 Idee
Wir wählen einen Ausgangspunkt x in der Nähe von x^*. Dann ist nach der Taylorentwicklung:

$$f(x) = f(x^*) + \underbrace{\nabla f(x^*)(x - x^*)}_{=0} + \frac{1}{2}(x - x^*)^T \nabla^2 f(x^*)(x - x^*) + \mathcal{O}(\|x - x^*\|^3)$$

Dies verhält sich also wie lokal wie eine quadratische Funktion. Wir modifizieren daher die Verfahren, die quadratische Minimieriungsprobleme gut lösen können, was sich als das cg-Verfahren herausstellt.

Ist x entfernt von x^*, dann laufen wir zumindest in die Abstiegsrichtung des cg-Verfahrens.

3.49 Algorithmus (Modifikation des cg-Verfahrens)
Wir haben $x_0 \in \mathbb{R}^n$ beliebig und wählen $d_0 = g_0 = \nabla f(x_0)$ und iterieren dann für $k = 0, 1, 2, \ldots$, bis z.B. das Abbruchkriterium $\|g_k\| \leq \texttt{tol} \cdot \|g_0\|$ erfüllt ist:
Wir wählen das Minimum von f in Suchrichtung $-d_k$, d.h. wir finden ein α für $x_{k+1} := x_k - \alpha_k d_k$ mit $f(x_{k+1}) = \min_\alpha f(x_k - \alpha d_k)$, was ein eindimensionales Minimierungsproblem ist, das wir z.B. auch nur näherungsweise berechnen können, etwa durch das Kriterium:

$$|\langle \nabla f(x_k + \alpha \cdot d_k), d_k \rangle| \leq \sigma |\langle \nabla f(x_k), d_k \rangle|$$

für ein $\sigma \in \mathbb{R}$.

Wir setzen dann $g_{k+1} := \nabla f(x_{k+1})$.

Dann betrachten wir die neue Suchrichtung $d_{k+1} = g_{k+1} + \beta_k d_k$.

3.50 Bemerkung
Für den Koeffizienten der neuen Suchrichtung gibt es in der Literatur zwei verschiedene Ansätze für β_k:

1. $\beta_k = \frac{\langle g_{k+1}, g_{k+1} \rangle}{\langle g_k, g_k \rangle}$ (Fletcher-Reeves)
2. $\beta_k = \frac{\langle g_{k+1} - g_k, g_{k+1} \rangle}{\langle g_k, g_k \rangle}$ (Polak-Ribiere)

Im quadratischen Fall sind beide Ansätze gleich, da $g_{k+1} \perp g_k$ ist.

Im nicht quadratischen Fall ist die Bedingung von Polak-Ribiere besser, denn:
Falls der Algorithmus „steckenbleibt“, d.h. $x_{k+1} \approx x_k$ und $g_{k+1} \approx g_k$, dann ist bei Polak-Riebiere $\beta_k \approx 0$, d.h. es gibt eine neue Suchrichtung und das Verfahren beginnt von neuem mit $d_{k+1} = g_{k+1}$. Dagegen wäre bei Fletcher-Reeves $\beta_k \approx 1$ und der Abstieg ist nicht garantiert.

3.51 Algorithmus (Vorkonditioniertes cg-Verfahren)
Wir können das vorkonditionierte cg-Verfahren als eine Beschleunigung des vereinfachten Newton-Verfahrens auffassen. Es ist

$$\nabla f(x) = 0, \qquad x_{k+1} = x_k + \alpha_k \cdot \Delta x_k$$

mit $\Delta x_k = -\nabla^2 f(x_0)^{-1} \underbrace{\nabla f(x_k)}_{=g(x_k)}$.
Hierbei benutzen wir die Hesse-Matrix $H(x_0) \approx L_0 L_0^T$ als evtl. unvollständige Cholesky-Zerlegung.
Dann ist $H(x_0)^{-1} \approx B := L_0^{-T} L_0^{-1} \approx CC^T$.
Dabei ist B wieder ein Vorkonditionierer.
Dann gilt $g(x) = 0 \iff \tilde{g}(\tilde{x}) = C^T \cdot g(C \underbrace{C^{-1}x}_{=\tilde{x}}) = 0$. Weiter ist $\tilde{H} = C^T H C$.

Wir können hier nun analog wie im Kapitel 6 weitermachen und erhalten als Suchrichtung:

$$d_{k+1} = Bg_{k+1} + \beta_k \cdot d_k$$

mit

$$\beta_k = \frac{\langle g_{k+1} - g_k, Bg_{k+1} \rangle}{\langle g_k, Bg_k \rangle}$$

3.52 Bemerkung
Wir könnten auch zeigen, daß das cg-Verfahren für beliebig strikt konvexe Funktionen konvergiert, d.h. die Hesse-Matrix ist in jedem Punkt positiv definit.
Für nicht konvexe Funktionen ist allerdings die Konvergenz nicht garantiert.

Wir werden im kommenden Kapitel sehen, wie wir ein Verfahren konstruieren, falls A nicht symmetrisch und positiv definit ist.

4 Iterative Verfahren für große lineare Gleichungssysteme

4.1 Motivation
Wir möchten im Folgenden das LGS $Ax = b$ mit $A \in \mathbb{R}^{n\times n}$ invertierbar betrachten. Üblicherweise ist n sehr groß, A aber schwach besetzt. Hierzu möchten wir ein Verfahren konstruieren, welches mit $\mathcal{O}(n)$ Operationen auskommt. Aus Numerik I kennen wir direkte Verfahren hierfür, etwa die LR-Zerlegung. Dabei haben wir allerdings das Problem eines sog. „fill-ins“, d.h. Nullelemente werden aufgefüllt und dadurch wird der Aufwand mit $\mathcal{O}(n^3)$ groß.

Ein Verfahren, welches einen besseren Rechenaufwand hat, ist ein iteratives indirektes Verfahren. Aus dem vorherigen Abschnitt wissen wir, daß es für ein symmetrisches und positiv definites A ein solches Verfahren gibt auf dem Krylovraum.

Obwohl unser A nun nicht mehr symmetrisch und positiv definit ist, suchen wir trotzdem ein iteratives Verfahren auf einem Krylovraum mit der Approximation

$$x_k \in \mathcal{K}_k(A, b) := \langle b, Ab, \ldots, A^{k-1}b\rangle$$

denn um diesen Raum zu konstruieren benötigen wir nur Multiplikationen mit A, was sich leicht berechnen läßt, wenn A dünn besetzt ist.

4.2 Bemerkung (Problem bei der Basiswahl)
Falls $V_k = (v_1, \ldots, v_k) \in \mathbb{R}^{n\times k}$ die Basis von $\mathcal{K}_k$ ist, dann läßt sich ein Element $x_k \in \mathcal{K}_k$ schreiben:

$$x_k = \sum_{i=1}^{k} y_{k,i} v_i = V_k y_k$$

Hierbei gibt es jedoch Probleme:

1. Wie wählen wir die Basis? Es ist keine gute Idee einfach $A^k b$ zu nehmen, denn beim Potenzverfahren dominieren große Eigenwerte die Matrix und wir erhalten damit keine handhabbare Basis, da die Vektoren fast linear abhängig werden.
2. Wir wissen nicht, wie wir die Koeffizienten wählen sollen.

Wir werden im Folgenden zwei Verfahren sehen, die auf dem Gram-Schmidt-Verfahren aufbauen, das Arnoldi- und das Lanczos-Verfahren, die nicht die Krylov-Iterationen berechnen. Außerdem werden wir sehen, wie wir den Koeffizientenvektor y_k geschickt wählen.

4.1 Arnoldi-Verfahren

4.3 Idee
Wir konstruieren uns eine ONB von $\mathcal{K}_k(A,b)$ über eine Modifikation des Gram-Schmidt-Verfahrens.

Wir betrachten deswegen eine Modifikation, da das ursprüngliche Gram-Schmidt-Verfahren numerisch nicht stabil ist und durch Rundungsfehler Vektoren konstruiert werden, die nicht orthogonal sind. Wir berechnen nun rekursiv:
Sei v_1 eine ONB von $\mathcal{K}_1$. Dann setzen wir $v_1 := \frac{1}{\beta} \cdot b$ mit $\beta = \|b\|_2$.
Für eine ONB $v_1, \ldots, v_k$ von $\mathcal{K}_k$ (bezüglich $\|.\|_2$). Dann wählen wir für $\mathcal{K}_{k+1}$:

$$\tilde{v}_{k+1} := Av_k - \sum_{j=1}^{k} h_{jk} v_j \in \mathcal{K}_{k+1}$$

mit den Koeffizienten h_{jk} so, daß $\tilde{v}_{k+1}$ orthogonal auf den anderen steht. Wir setzen

$$v_{k+1} := \frac{\tilde{v}_{k+1}}{\|\tilde{v}_{k+1}\|_2}$$

Wir möchten nun h_{jk} bestimmen. Für $i = 1, \ldots, k$ muß gelten:

$$0 = \langle v_i, \tilde{v}_{k+1} \rangle = \langle v_i, Av_k \rangle - \sum_{j=1}^{k} h_{jk} \underbrace{\langle v_i, v_j \rangle}_{=\delta_{ij}} = \langle v_i, Av_k \rangle - h_{ik}$$

Also ist $h_{ik} = \langle v_i, Av_k \rangle$.

4.4 Algorithmus (Arnoldi, 1951)
Wir bestimmen: $\beta := \|b\|$ und $v_1 := \frac{1}{\beta} b$.

Für $k \geq 1$ berechnen wir $\tilde{v}_{k+1}^{(1)} = Av_k$ und nacheinander für $j = 1, \ldots, k$:

$$h_{jk} := \left\langle v_j, \tilde{v}_{k+1}^{(j)} \right\rangle \qquad \tilde{v}_{k+1}^{(j+1)} := \tilde{v}_{k+1}^{(j)} - h_{jk} v_j$$

und anschließend:

$$h_{k+1,k} := \left\| \tilde{v}_{k+1}^{(k+1)} \right\| \qquad v_{k+1} := \frac{1}{h_{k+1,k}} \cdot \tilde{v}_{k+1}^{(k+1)}$$

4.5 Bemerkung (Abbruchverhalten)
Das Arnoldi-Verfahren bricht ab, falls $h_{k+1,k} = 0$. Dann ist nämlich

$$AV_k = V_k H_k \iff \mathcal{K}_{k+1} = \mathcal{K}_k$$

4.6 Bemerkung (Vereinfachungen in Matrixschreibweise)
Wir setzen $H_k = (h_{ij})_{i,j=1}^{k}$ und sehen $h_{ij} = 0$ für $i > j+1$, d.h. dies ist eine Hessenbergmatrix. Dann ist $\tilde{H}_k = (h_{ij})_{i,j=1}^{(k+1,k)}$ eine Hessenbergmatrix mit einer zusätzlichen Zeile. Mit dieser Notation und dem Algorithmus schreiben wir nun für jedes k (mit $V_k := (v_1, \ldots, v_k)$):

$$Av_k = h_{k+1,k}v_{k+1} + \sum_{j=1}^{k} h_{jk}v_j, \quad \text{also: } AV_k = V_{k+1}\tilde{H}_k = V_kH_k + h_{k+1,k}v_{k+1}e_k^T$$

Die Orthogonalität $\langle v_i, v_j\rangle = v_i^T v_j = \delta_{ij}$ wird dann zu: $V_k^T V_k = I$. Damit erhalten wir:

$$V_k^T AV_k = V_k^T V_{k+1}\tilde{H}_k = \begin{pmatrix} I_k & 0 \end{pmatrix} \tilde{H}_k = H_k, \quad \text{also: } H_k = V_k^T AV_k$$

4.7 Bemerkung (Spezialfall einer symmetrischen Matrix)
Falls A symmetrisch ist, vereinfachen sich einige Rekursionen erheblich. Dann ist H_k symmetrisch, also tridiagonal. Damit erhalten wir folgende Drei-Term-Rekursion:

$$h_{k+1,k}v_{k+1} = \tilde{v}_{k+1} = Av_k - h_{kk}v_k - h_{k-1,k}v_{k-1}$$

d.h. wir haben $\mathcal{O}(k)$ Operationen zur Berechnung von $v_1, \ldots, v_k$.
Im allgemeinen (nicht symmetrischen) Fall sind es $\mathcal{O}(k^2)$ Operationen.

4.8 Hilfssatz
Ist $h_{k+1,k} = 0$ und $h_{j+1,j} \neq 0$ für $j \leq k-1$. Dann gilt:

1. Jeder Eigenwert von H_k ist ein Eigenwert von A. Dies gilt sogar, falls A nicht invertierbar ist.
2. $\exists y_k \in \mathbb{R}^k$ so, daß $Ax = b$ gilt mit $x = V_k y_k$.

Beweis

1. Sei λ ein Eigenwert von H_k zum EV $y \neq 0$. Dann ist $H_k y = \lambda y \iff V_k H_k y = \lambda V_k y$, aber $V_k H_k = AV_k$ wegen $h_{k+1,k} = 0$. Also ist $A(V_k y) = \lambda(V_k y)$. Dann ist λ ein Eigenwert von A mit dem Eigenvektor $V_k y$.
2. Da A invertierbar ist, ist 0 kein Eigenwert von A, also ist auch 0 kein Eigenwert von H_k, d.h. H_k ist ebenfalls invertierbar. Wir möchten, daß $b = AV_k y_k = V_k H_k y_k$. Aber wir haben $b = \beta v_1 = V_k \beta e_1$.

   Da V_k invertierbar ist, impliziert also die Lösung von $H_k y_k = \beta e_1$, daß $AV_k y_k = \beta v_1$ gilt. $H_k y_k = \beta e_1$ ist aber lösbar, denn H_k ist invertierbar. □

4.2 FOM und GMRES: Garlekin und Minimierung des Residuums

Wir möchten nun den Koeffizientenvektor y_k bestimmen. Dazu gibt es zwei Ansätze:

4.9 Idee (Galerkin-Ansatz)
Wir verlangen, daß das Residuum $Ax_k - b$ orthogonal auf dem Krylovraum stehen soll, d.h.:

$$\langle Ax_k - b, v\rangle = 0 \quad \forall v \in \mathcal{K}_k$$

Falls A symmetrisch, positiv definit ist, erhielten wir dafür das cg-Verfahren. Wir ignorieren die Definitheit von A und rechnun nun analog weiter.

Es genügt nun, daß dies orthogonal auf allen Basisvektoren ist, d.h. $V_k^T(Ax_k - b) = 0$. Dann folgt aus $x_k = V_k y_k$, daß $Ax_k = AV_k y_k$ ist und daß $b = \beta v_1 = \beta V_k e_1$ ist. Daraus ergibt sich dann, wenn wir $V_k^T V_k = I_k$ und $V_k^T v_{k+1} = 0$ benutzen:

$$V_k^T(Ax_k - b) = V_k^T Ax_k - V_k^T b = V_k^T V_{k+1}\tilde{H}_k y_k - \beta V_k^T V_k e_1 = H_k y_k - \beta e_1$$

Wir fordern, daß dies Null ist, d.h. wir erhalten die Gleichung $H_k y_k = \beta e_1$ und dies ist eindeutig lösbar, falls H_k invertierbar ist.

4.10 Bemerkung
Beim cg-Verfahren war A symmetrisch, positiv definit, also auch H_k.

4.11 Algorithmus (FOM)
Wir lösen iterativ $x_k = V_k y_k$ mit $H_k y_k = \beta e_1$.

4.12 Historische Notiz (FOM)
FOM steht für „full orthogonalisation method“ und wurde von Saad, 1981 vorgestellt.

4.13 Idee (Minimierung des Residuums)
Wir minimieren nun das Residuum, d.h. wir suchen $x_k \in \mathcal{K}_k(A, b)$ mit $\|Ax_k - b\|_2 = \min$. Weil V_{k+1} orthogonal ist und Normen invariant unter orthogonalen Matrizen ist, gilt:

$$\|Ax_k - b\|_2 = \|AV_k y_k - \beta v_1\|_2 = \|V_{k+1}(\tilde{H}_k y_k - \beta e_1)\|_2 = \|\tilde{H}_k y_k - \beta e_1\|_2$$

Wir erhalten hier also ein lineares Ausgleichsproblem: Finde $y_k \in \mathbb{R}^k$ mit $\|\tilde{H}_k y_k - \beta e_1\|_2 = \min$ für $y_k \in \mathbb{R}^k$.

Ist $h_{i+1,i} \neq 0$ für $i = 1, \ldots, k$, dann ist das Ausgleichsproblem eindeutig lösbar.

4.14 Algorithmus (GMRES)
Wir berechnen $x_k := V_k y_k$ mit $\|\tilde{H}_k y_k - \beta e_1\|_2 = \min$.

4.15 Historische Notiz
GMRES steht für „generalized minimum residual method“ und wurde von Saad und Schultz 1986 entwickelt.

4.16 Proposition (Eigenschaften von FOM und GMRES)

1. Ist k der erste Index mit $h_{k+1,k} = 0$ (dann gilt $AV_k = V_k H_k$), dann liefern sowohl FOM als auch GMRES im k-ten Schritt die exakte Lösung von $Ax = b$.
2. Für das GMRES-Residuum gilt:
$$\|Ax_k - b\|_2 = \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \|p_k(A)b\|_2$$
3. Sei $A = T\Lambda T^{-1}$ mit $\Lambda = \operatorname{diag}(\lambda_j)$. Dann gilt:
$$\|Ax_k - b\| = \operatorname{cond}(T) \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \max_{j=1,\dots,n} |p_k(\lambda_j)| \|b\|$$
4. Genau dann wenn H_k nicht invertierbar ist (d.h. $x_k^{\texttt{FOM}}$ existiert), stagniert $x_{k+1}^{\texttt{GMRES}} = x_k^{\texttt{GMRES}}$

Beweis

1. Für FOM folgt dies aus dem Beweis des Hilfssatzes des vorigen Kapitels. Für GMRES rechnen wir:
$$\|Ax_k^{\texttt{GMRES}} - b\|_2 \leq \|Ax_k^{\texttt{GMRES}} - b\|_2 = 0$$
2. Es ist $x_k = q_{k-1}(A)b$ mit $\deg q_{k-1} = k-1$ und wir rechnen:
$$b - Ax_k = (I - Aqk - 1(A))b = p_k(A)b \quad \text{mit } p_k(\lambda) = 1 - \lambda q_{k-1}(\lambda)$$
Aus der Minimierungs-Forderung des Residuums folgt dann die Aussage.
3. Mit dem zweiten Teil gilt:
$$\begin{aligned}
\|Ax_k - b\| &= \min_{p_k(0)=1} \|p_k(A)b\| \\
&= \min_{p_k(0)=1} \|Tp_k(\Lambda)T^{-1}b\| \\
&\leq \min_{p_k(0)=1} \|T\| \|p_k(\Lambda)\| \|T^{-1}\| \|b\| \\
&\leq \operatorname{cond}(T) \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \max_{j=1,\dots,n} |p_k(\lambda_j)| \|b\|
\end{aligned}$$
4. ohne Beweis. □

4.17 Bemerkung (Implementierung von GMRES)
Wir machen eine QR-Zerlegung von $\tilde{H}_k$, d.h. es gibt ein Q_k mit:

$$\tilde{H}_k = Q_k \begin{pmatrix} R_k \\ 0 \end{pmatrix}$$

und damit gilt:

$$\|\tilde{H}_k y_k - \beta e_1\|_2 = \left\| \begin{pmatrix} R_k y_k \\ 0 \end{pmatrix} - \beta Q_k^T e_1 \right\|$$

4.18 Bemerkung (Rechenaufwand beim GMRES-Verfahren)
Wir möchten nun den Rechenaufwand für den nichtsymmetrischen Fall, $A \neq A^T$ betrachten.

1. Wir müssen v_{k+1} mit dem Arnoldi-Verfahren berechnen und haben k Multiplikationen von A mit einem Vektor. Hierfür brachen wir $\mathcal{O}(k^2 n)$ Operationen.
2. Die Berechnung von y_k benötigt $\mathcal{O}(k^2)$ Operationen.
3. Die Berechnung von $x_k = V_k y_k$ benötigt kn Operationen.

Ein großes Problem stellt die Speicherung dar: Wir benötigen $(k+1)n$ Speicherungen, d.h. das GMRES-Verfahren ist nur durchzuführen, wenn k nicht allzu groß ist, sofern n groß ist.

Diesen Umstand können wir z.B. dadurch lösen, nur eine feste Anzahl von Iterationen durchzuführen und dann das Verfahren neu zu starten. Der Nachteil dieser Methode ist eine schlechtere Konvergenz.

4.19 Bemerkung (Abbruchkriterium)
Als typische Abbruchbedingung verlangen wir, daß $\|Ax_k - b\| \leq \texttt{tol}$ ist.

Dies ist eine sehr aufwendige Rechnung. Wir wissen aber, daß $\|Ax_k - b\| = \|\tilde{H}_k y_k - \beta e_1\|$ ist, was sich leichter berechnen läßt.

Wir brechen also unser Verfahren ab, wenn gilt:

$$\|\tilde{H}_k yk - \beta e_1\| \leq \texttt{tol}$$

4.3 Lanczos-Verfahren

4.20 Motivation
Die Schwierigkeit beim Arnoldi-Verfahren ist, daß wir für v_{k+1} die Vektoren $v_1, \ldots, v_k$, was zu einem großen Rechen- und vor allem Speicheraufwand führt.

Unser Ziel ist es, eine Basis von $\mathcal{K}_k(A, b)$ mit einer kurzen Rekursion zu erhalten (nicht nur für symmetrische A).

4.21 Idee (Laczos)
1. Wir verzichten auf die ONB

2. Wir berechnen zwei Basen $v_1, \ldots, v_k$ von $K_k(A, b)$ und $w_1, \ldots, w_k$ von $\mathcal{K}_k(A^T, b)$.
3. Wir verlangen Bi-Orthogonalität, d.h.

$$\langle w_i, v_j \rangle = 0 \quad \forall i \neq j$$

4.22 Konstruktion (Herleitung des Algorithmus)
Wir machen folgenden Ansatz:

$$v_{k+1} = Av_k - \sum_{j=1}^{k} h_{jk} v_j, \qquad w_{k+1} = A^T w_k - \sum_{j=1}^{k} \ell_{jk} w_j$$

Wegen der Bi-Orthogonalität fordern wir nun für $i = 1, \ldots, k$:

$$0 = \langle w_i, v_{k+1} \rangle = \langle w_i, Av_k \rangle - \langle w_i, v_i \rangle h_{i,k}, \qquad 0 = \langle w_{k+1}, v_i \rangle = \langle A^T w_k, v_i \rangle - \langle w_i, v_i \rangle \ell_{ik}$$

Wir berechnen nun mit Hilfe der Basisdarstellung:

$$\langle w_i, Av_k \rangle = \langle A^T w_i, v_k \rangle = \left\langle w_{i+1} + \sum_{j=1}^{i} \ell_{ji} w_j, v_k \right\rangle = 0 \quad \text{für } i + 1 < k$$

Falls außerdem $\langle w_i, v_i \rangle \neq 0$, dann folgt daraus, daß $h_{ik} = 0$ ist für $i + 1 < k$.

Analog erhalten wir $\ell_{ik} = 0$ für $i + 1 < k$. Wir berechnen nun:

$$h_{kk} = \frac{\langle w_k, Av_k \rangle}{\langle w_k, v_k \rangle} = \frac{\langle A^T w_k, v_k \rangle}{\langle w_k, v_k \rangle} = \ell_{kk}$$

Weiter berechnen wir mit Hilfe der Rekursion:

$$h_{k-1,k} = \frac{\langle w_{k-1}, Av_k \rangle}{\langle w_{k-1}, v_{k-1} \rangle} = \frac{\langle A^T w_{k-1}, v_k \rangle}{\langle w_{k-1}, v_{k-1} \rangle} = \frac{\langle w_k + \sum \ldots \rangle}{\langle w_{k-1}, v_{k-1} \rangle} = \frac{\langle w_k, v_k \rangle}{\langle w_{k-1}, v_{k-1} \rangle} = \ell_{k-1,k}$$

4.23 Algorithmus (Lanczos, 1952)

1. Wir wählen $b, c \neq 0$ und wir setzen $v_1 := b$, $w_1 := c$ und $v_0 := w_0 := 0$.
2. Für $k = 1, 2, 3 \ldots$ (solange $\langle w_k, v_k \rangle \neq 0$) setzen wir:

$$\alpha_k := \frac{\langle w_k, Av_k \rangle}{\langle w_k, v_k \rangle}, \qquad \beta_k := \frac{\langle w_k, v_k \rangle}{\langle w_{k-1}, v_{k-1} \rangle}$$

wobei β_k nur für $k \geq 2$ existiert. Schließlich setzen wir:

$$v_{k+1} := Av_k - \alpha_k v_k - \beta_k v_{k-1}, \qquad w_{k+1} := A^T w_k - \alpha_k w_k - \beta_k w_{k-1}$$

4.24 Bemerkung
Für $A = A^T$ stimmen Arnoldi- und Lanczos-Verfahren überein.

4.25 Notation (Vereinfachung in Matrix-Schreibweise)
Wir bezeichnen mit $V_k := (v_1, \dots, v_k)$ und $W_k := (w_1, \dots, w_k)$ die Basis-Matrizen und

$$T_k := \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ 1 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \dots & 0 & 1 & \alpha_k \end{pmatrix}$$

und wir setzen:

$$\tilde{T}_k := \left(\begin{array}{cccc|c} & & & & \\ & & T_k & & \\ & & & & \\ \hline 0 & \dots & 0 & 1 & \end{array} \right) \in \mathbb{R}^{(k+1)\times k}$$

4.26 Bemerkung (Lanczos-Verfahren in Matrix-Schreibweise)
Wir erhalten in der obigen Schreibweise:

$$AV_k = V_{k+1}\tilde{T}_k, \qquad A^T W_k = W_{k+1}\tilde{T}_k$$

Insbesondere ist $\mathcal{K}(A, b) = \text{Span}\{v_1, \dots, v_k\}$ und $\mathcal{K}(A^T, c) = \text{Span}\{w_1, \dots, w_k\}$ und wir erhalten mit der Biorthogonalität:

$$W_k^T V_k = \text{diag}(w_1^T v_1, \dots, w_k^T v_k) =: D_k$$

Es gilt dann $W_k^T A V_k = D_k T_K$ und damit $T_k = D_k^{-1} W_k^T A V_k$, denn es gilt:

$$W_k^T A V_k = W_k^T V_{k+1}\tilde{T}_k = \begin{pmatrix} D_k & 0 \end{pmatrix} \tilde{T}_k = \begin{pmatrix} D_k & 0 \end{pmatrix} \begin{pmatrix} T_k \\ \hline \star \end{pmatrix} = D_k T_k$$

4.27 Bemerkung

1. In der Praxis werden oft v_i und w_i normiert.
2. Wir wählen als Abbruchbedingung $\langle w_{k+1}, v_{k+1} \rangle = 0$.
   Dies können wir in zwei verschiedene Abbrucharten unterteilen:
   a) Einen *regulären Abbruch* haben wir, falls $v_{k+1} = 0$ ist, dann ist nämlich $AV_k = V_k T_k$ und $\mathcal{K}_{k+1}(A, b) = \mathcal{K}_k(A, b)$. Wegen der Symmetrie von v und w ergibt sich analog bei $w_{k+1} = 0$, daß $A^T W_k = W_k T_k$ und $\mathcal{K}_{k+1}(A^T, c) = \mathcal{K}_k(A^T, c)$
   b) Einen *ernsthaften Abbruch* haben wir, falls $v_{k+1} \neq 0$ und $w_{k+1} \neq 0$, aber es könnte sein, daß $\langle v_{k+1}, w_{k+1} \rangle \approx 0$, da wir beim Rechnen Rundungsfehler haben bzw. es zu einer numerischen Instabilität beim Invertieren aufgrund schlechter Kondition kommt.

4.28 Bemerkung (look-ahead Lanczos)
Wir möchten, daß ein ernsthafter Abbruch möglichst nicht vorkommt.
Dazu modifizieren wir das Lanczos-Verfahren, indem wir die Biorthogonalität zur Blockorthogonalität abschwächen, d.h. wir haben:

$$W_k^T V_k = \begin{pmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & B_\ell \end{pmatrix}$$

mit B_i irgendein quadratischer Matrix-Block und $\|B_i^{-1}\|$ nicht zu groß.

Wir erhalten damit einen stabilen Algorithmus mit einer kurzen Rekursion.

4.4 BiCG und QMR

4.29 Motivation
Wir haben noch immer das LGS $Ax = b$ und suchen eine Approximation $x_k = V_k y_k$ mit $V_k = \begin{pmatrix} v_1 & \dots & v_k \end{pmatrix}$ eine Lanczos-Basis. Wir haben dafür verschiedene Ansätze:

4.30 Bemerkung (Galerkin-Ansatz)
Wir fordern, daß $\langle w, Ax_k - b\rangle = 0 \quad \forall w \in \mathcal{K}_k(A^T, c)$ gilt. Wir testen dies für $w = w_i$. Dann ist $W_k^T(Ax_k - b) = 0$. Ähnlich wie beim Arnoldi-Verfahren erhalten wir dann für $b = \beta v_1 = V_k \beta e_1$:

$$W_k^T A x_k - W_k^T V_k \beta e_1 = 0$$

Wegen $x_k = V_k y_k$ gilt damit: $W_k^T A V_k y_k - W_k^T V_k \beta e_1 = 0$. Aber weil $W_k^T A V_k = D_k T_k$ und $W_k^T V_k = D_k$ ist, erhalten wir damit:

$$D_k T_k y_k = D_k \beta e_1$$

Falls D_k invertierbar ist, ergibt sich das folgende LGS mit der tridiagonalen Matrix T_k :

$$T_k y_k = \beta e_1$$

4.31 Algorithmus (bikonjugiertes Gradientenverfahren (BiCG))
Wir lösen $x_k = V_k y_k$ und mit $T_k y_k = \beta e_1$.

4.32 Bemerkung (Rechenaufwand)
Wir lösen das LGS über eine LR-Zerlegung $T_k = L_k \cdot R_k$, d.h.

$$x_k = V_k y_k = \underbrace{V_k R_k^{-1}}_{=:P_k} \underbrace{L_k^{-1} \beta e_1}_{=:q_k} \quad \text{mit } q_k = \begin{pmatrix} \varrho_1 \\ \dots \\ \varrho_k \end{pmatrix} = \begin{pmatrix} q_{k-1} \\ \varrho_k \end{pmatrix}$$

Dann ist $L_k q_k = \beta e_1$, also $\varrho_k = -\ell_{k,k-1} \varrho_{k-1}$, womit wir dies in einer kurzen Rekursion berechnen können.

Mit der Definition von P_k gilt $P_k R_k = V_k = \begin{pmatrix} v_1 & \dots & v_k \end{pmatrix}$.

Dann ist $P_k := \begin{pmatrix} p_1 & \dots & p_k \end{pmatrix}$ und aus $V_k = P_k R_k$ erhalten wir:

$$v_1 = p_1 r_{11}, \qquad v_2 = p_1 r_{12} + p_2 r_{22}, \qquad \dots, \qquad v_k = p_{k-1} r_{k-1,k} + p_k r_{kk}$$

Dies sind kurze Rekursionen für p_k und damit gilt für x_k:

$$x_k = P_k q_k = P_{k-1} q_{k-1} + p_k \varrho_k = x_{k-1} + p_k \varrho_k$$

Damit haben wir also auch für x_k kurze Rekurssionen und wir können dann iterativ aus (x_k, p_k, ϱ_k) die Größen $(x_{k+1}, p_{k+1}, \varrho_{k+1})$.

4.33 Historische Notiz
Das BiCG-Verfahren wurde 1952 von Lanczos und 1976 von Fletcher vorgeschlagen.

4.34 Konstruktion (Minimierungsansatz)
Wir betrachten nun einen weiteren Ansatz zur Lösung des Problems und verlangen:

$$\beta := \|b\|, \qquad \|v_j\| = \|w_j\| = 1 \quad \forall j$$

und dann ergibt sich:

$$\|Ax_k - b\| = \|AV_k y_k - \beta v_1\| = \|V_{k+1} \tilde{T}_k y_k - V_{k+1} \beta e_1\| \leq \|V_{k+1}\| \cdot \|\tilde{T}_k y_k - \beta e_1\|$$

Falls $\|v_j\| = 1 \; \forall j$, dann ist $\|V_{k+1}\| \leq \sqrt{k+1}$.

Die ist Idee ist es nun, *nur* die Norm des Koeffizientenvektors $\|\tilde{T}_k y_k - \beta e_1\|$ zu minimieren.

4.35 Algorithmus (Quasi-Minimierung des Residuums (QMR))
Wir berechnen $x_k = V_k y_k$ mit $\|\tilde{T}_k y_k - \beta e_1\| = \min$.

4.36 Bemerkung (Rechenaufwand)
Die Implementierung QMR-Verfahrens läuft ähnlich wie beim GMRES-Verfahren ab und benutzt Ideen aus dem BiCG-Verfahren.

Es ist $\tilde{T}_k = Q_k \begin{pmatrix} R_k \\ 0 \end{pmatrix}$ und

$$Q_k^T(\beta e_1) = \begin{pmatrix} q_k \\ \varrho_k \end{pmatrix} \quad \text{mit } q_k = \begin{pmatrix} \varrho_1 \\ \vdots \\ \varrho_k \end{pmatrix} = \begin{pmatrix} q_{k-1} \\ \varrho_k \end{pmatrix}$$

Weil die Lösung des linearen Ausgleichsproblem äquivalent zu $R_k y_k = q_k$ ist und $x_k = V_k yk = V_k R_k^{-1} q_k$ gilt mit $V_k R_k^{-1} = P_k = \begin{pmatrix} p_1 & \dots & p_k \end{pmatrix}$, erhalten wir nun:

$$x_k = \underbrace{P_{k-1} q_{k-1}}_{=x_{k-1}} + p_k \varrho_k$$

Damit erhalten wir bei bei BiCG kurze Rekursionen, um aus (x_k, P_k, ϱ_k) dann die Größen $(x_{k+1}, P_{k+1}, \varrho_{k+1})$ zu berechnen.

4.37 Historische Notiz
Das QMR-Verfahren wurde 1991 von Freud und Nachtigal vorgeschlagen.

4.38 Bemerkung
Das QMR-Verfahren verläuft in der Konvergenz deutlich glatter als das BiCG-Verfahren, ist jedoch nicht unbedingt monoton fallend (wie etwa beim GMRES-Verfahren).

4.39 Zusammenfassung
Wir hatten verschiedene Verfahren:

	Galerkin	Minimierung des Residuums
Arnoldi	FOM	GMRES
Lanczos	BiCG	QMR

Wir vergleichen nochmals das Arnoldi- und Laczos-Verfahren:

Arnoldi	Lanczos
ONB	Biorthogonal-Basen
stabil	unter Umständen Stablisierung mit look-ahead nötig
lange Rekursion	kurze Rekursion

Wir können auch Galerkin und Minimierung vergleichen:

Galerkin	Minimierung des Residuums
LGS	Lineares Ausgleichsproblem
	Residuumsnormen hängen glatter von der Iterationszahl ab
	(bei GMRES: monoton fallend)

In der Praxis ist die Vorkonditionierung sehr wichtig. Dies fällt jedoch unter das Motto „more art than science“.

Wir lösen also statt $Ax = b$ dann $M_1 A M_2 u = M_1 b$ mit $x = M_2 u$, z.B. könnte man eine unvollständige LR-Zerlegung machen.

5 Lineare Optimierung

5.1 Beispiele (aus der Wirtschaft)

5.1 Beispiel (Produktionsorganisation)
Eine Fabrik stellt zwei Produkte p_1 und p_2 aus drei Rohstoffen q_1, q_2, q_3 her. Die Erzeugung einer Einheit von p_1 benötigt 1 Einheit von q_1, 2 Einheiten von q_2 und 4 Einheiten von q_3. Für die Erzeugung von p_2 werden 6 Einheiten q_1 benötigt, 2 Einheiten q_2 und 1 Einheit q_3.

Uns stehen 30 Einheiten von q_1, 15 Einheiten von q_2 und 24 Einheiten von q_3 zur Verfügung. Wir haben einen Verkaufsgewinn von p_1 zwei Euro pro Einheit und p_2 ein Euro pro Einheit.

Wir möchten die optimale Produktionszahlen von p_1 und p_2 finden, d.h. wir suchen nun $x = (x_1, x_2)$ so, daß $2x_1 + 1x_2 = \max$. Dabei sind die Nebenbedingungen $x_1 \geq 0$ und $x_2 \geq 0$ zu beachten. Durch die begrenzten Rohstoffe erhalten wir weitere Nebenbedingungen:

$$1x_1 + 6x_2 \leq 30, \qquad 2x_1 + 2x_2 \leq 15, \qquad 4x_1 + 1x_3 \leq 24$$

Wir können dies grafisch lösen, indem wir die Nebenbedingungen in ein Schaubild zeichnen und den zulässigen Bereich als Polygon erkennen. Wir suchen dann Äquipotentiallinien, indem wir betrachten, wo $2x_1 = \text{const}$, z.B. $2 \cdot x_1 = 0$. Dies verschieben wir dann solange parallel nach oben (dabei steigt die Konstante natürlich an), bis wir gerade noch einen Punkt im zulässigen Bereich finden.

Die Lösung tritt also an einer Ecke auf, d.h. wir können alle Ecken ausprobieren und erhalten dann durch Vergleich von endlich vielen Werten die Lösung. Hier ist die Lösung eindeutig, und zwar $(x_1, x_2) = \left(\frac{11}{2}, 2\right)$. Als Gewinn erhalten wir: $2 \cdot \frac{11}{2} + 1 \cdot 2 = 13$.

5.2 Bemerkung

1. Für $2x_1 + 2x_2 = \max$ wäre dies z.B nicht eindeutig, sondern würde entlang einer Kante überall den gleichen maximalen Gewinn bringen.
2. Die Lösung $x_1 = \frac{11}{2}$ setzt die Teilbarkeit des Produktes p_1 voraus, andernfalls könnten wir verlangen, daß $(x_1, x_2) \in \mathbb{Z} \times \mathbb{Z}$ liegt, was zur ganzzahligen Optimierung führt, die viel komplizierter ist und hier nicht behandelt wird.

5.3 Beispiel (Konsumentenproblem)
Wir haben vier Nahrungsmittel und charakterisieren diese über die folgende Tabelle:

	A	B	C	D
Nährwert	2	1	0	1
Vitamine	3	4	3	5
Preis	2	2	1	8

Wir möchten nun zu einem minimalen Preis 12 Nährwert-Einheiten und 7 Vitamin-Einheiten kaufen müssen. Wir suchen $x = (x_1, x_2, x_3, x_4)$ mit $x_i \geq 0$ mit den Nebenbedingungen:

$$2x_1 + 1x_2 + 0x_3 + 1x_4 \geq 12, \qquad 3x_1 + 4x_2 + 3x_3 + 5x_4 \geq 7$$

und dem Minimierungsproblem:

$$2x_1 + 2x_2 + 1x_3 + 8x_4 = \min$$

5.4 Beispiel (Erweiterung auf das Konkurrentenproblem)
Ein Konkurrent bringt zwei neue Lebensmittel auf den Markt und hat die folgenden Produkte im Sortiment:

	I	II
Nährwert	1	0
Vitamine	0	1
Preis	y_1	y_2

Wir möchten nun die Preise y_1 und y_2 bestimmen so, daß der Gesamtverkaufspreis maximal, d.h. wir haben das Problem $12y_1 + 7y_2 = \max$, da die Konsumenten nur das Nötige kaufen. Wir müssen also dieselbe Menge an Nährwert und Vitaminen finden, dies aber billiger tun als die anderen Produkte, d.h. wir haben unter $y_1, y_2 \geq 0$ die Nebenbedingungen

$$2y_1 + 3y_2 \leq 2, \qquad 1y_1 + 4y_2 \leq 2, \qquad 0y_1 + 3y_2 \leq 1, \qquad 1y_1 + 5y_2 \leq 8$$

5.5 Bemerkung
Die beiden Probleme sind zueinander *duale Probleme* im folgenden Sinne:

- Das eine Konsumentenproblem läßt sich schreiben als: $c^T x = \min$ mit $x \in \mathbb{R}^4_+$ und einer Matrix A so, daß $Ax \geq b$.
- Beim Konsumentenproblem haben wir $b^T y = \max$ mit $y \in \mathbb{R}^2_+$ und $A^T y \leq c$.

wobei wir in beiden Fällen die Ungleichungen als komponentenweise Ungleichung auffassen.

5.6 Beispiel (Transportproblem)
Wir haben m Produzenten und n Märkte. Die Transportkosten für eine Einheit von i nach j ist durch C_{ij} erfasst. Das Angebot ist $a_1, \ldots, a_m$ und die Nachfrage ist $b_1, \ldots, b_n$.

Das Problem ist, was der kostengünstigste Transportplan ist, um die Produkte an die Märkte zu versenden. Unsere Unbekannten ist die Liefermenge von i nach j, genannt $x_{ij} \geq 0$. Dabei berechnen sich die Gesamtkosten wie folgt:

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} x_{ij}$$

Wir setzen voraus, daß das Gesamtangebot die Gesamtnachfrage ist, also $\sum a_i = \sum b_j$. Damit ergibt sich:

- Der Gesamtversand des Produzenten i ist $\sum_{j=1}^{n} x_{ij} = a_i$.
- Die Gesamtaufnahme des Marktes j ist $\sum_{i=1}^{m} x_{ij} = b_j$.

Wir haben also $m \cdot n$ Unbekannte $x_{ij} \geq 0$ unter $m + n$ Nebenbedingungen, wovon aber nur $m + n - 1$ unabhängig sind.

5.7 Beispiel (Duales Problem zum Transportproblem)
Ein Paketservice bietet alternativ einen kostenfreien Transport auf allen Strecken, verlangt aber eine Gebühr u_i bei der Aufgabe einer Einheit in i und v_j beim Abholen einer Einheit in j.

Die Einnahmen dieses Service sind:

$$z' = \sum_{i=1}^{m} u_i a_i + \sum_{j=1}^{n} v_j b_j$$

z' soll optimiert werden, allerdings muß als Nebenbedingung gelten, daß dies noch immer billiger ist als die traditionelle Post, d.h.

$$u_i + v_j \leq C_{ij} \quad \forall i, j$$

Für die Optima gilt immer, daß $z' \leq z$ ist, denn:

$$\begin{aligned} z' &= \sum_{i=1}^{m} u_i a_i + \sum_{j=1}^{n} v_j b_j \\ &= \sum_{i} u_i \sum_{j} x_{ij} + \sum_{j} v_j \sum_{i} x_{ij} \\ &= \sum_{i,j} (u_i + v_j) x_{ij} \\ &\leq \sum_{i,j} C_{ij} x_{ij} = z \end{aligned}$$

Wir berechnen nun, wann Gleichheit gilt:

$$z' = z \iff (u_i + v_j - C_{ij}) x_{ij} = 0 \; \forall i, j \iff u_i + v_j = C_{ij} \text{ oder } x_{ij} = 0 \quad \forall i, j$$

Wenn dies gilt, ist bereits z' maximal und z minimal ist

5.8 Bemerkung
Es gilt auch die Umkehrung der letzten Aussage, d.h. wenn z' maximal und z minimal ist, dann gilt bereits $z = z'$, was wir im Kapitel über die Dualität sehen werden.

5.2 Lineare Programme (Optimierungsaufgaben)

5.9 Definition (Mathematische Problemstellung)
Wir haben $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$ gegeben. Dies schreiben wir in ein Tableau:

A	b
c^T	

Dies führt uns zu verschiedenen Problemstellungen:

1. Wir suchen $x \in \mathbb{R}^n$ mit:
$$Ax \le b, \qquad c^T x = \min$$
2. Wir suchen $x \in \mathbb{R}^n$ mit:
$$Ax \le b, \qquad x \ge 0, \qquad c^T x = \min$$
3. Wir suchen $x \in \mathbb{R}^n$ mit:
$$Ax = b, \qquad x \ge 0, \qquad c^T x = \min$$

wobei wir in jedem Fall die Ungleichungen komponentenweise auffassen. Die Funktion $c^T x$ wird auch Ziel- bzw. Kostefunktion genannt.

5.10 Proposition (Umformulierung der Standardformen)
Alle drei Probleme lassen sich äquivalent ineinander umformulieren.

Beweis
(1) $\Rightarrow$ (2)**:** Wir setzen $x := x_+ - x_-$ mit $x_+, x_- \ge 0$.
Dann ist

$$\begin{aligned}(1) &\iff A(x_+ - x_-) \le b, \quad x_+, x_- \ge 0, \quad c^T(x_+ - x_-) = \min \\ &\iff (A_+, -A)\begin{pmatrix} x_+ \\ x_- \end{pmatrix} \le b, \quad \begin{pmatrix} x_+ \\ x_- \end{pmatrix} \ge 0, \quad (c^T, -c^T)\begin{pmatrix} x_+ \\ x_- \end{pmatrix} = \min\end{aligned}$$

(2) $\Rightarrow$ (3)**:** Wir führen sogenannte Schlupfvariablen (slack variables) ein:
Sei $z \in \mathbb{R}^m$ mit $z \ge 0$. Damit erhalten wir:

$$\begin{aligned}(2) &\iff Ax + z = b, \quad x \ge 0, \quad z \ge 0, \quad c^T x = \min \\ &\iff (A, I)\begin{pmatrix} x \\ z \end{pmatrix} = b, \quad \begin{pmatrix} x \\ z \end{pmatrix} \ge 0, \quad (c^T, 0)\begin{pmatrix} x \\ z \end{pmatrix} = \min\end{aligned}$$

(3) $\Rightarrow$ (1)**:** Es ist:

$$\begin{aligned}(3) &\iff Ax \le b, \quad Ax \ge b, \quad x \ge 0 \\ &\iff Ax \le b, \quad -Ax \le -b, \quad -x \le 0 \\ &\iff \begin{pmatrix} A \\ -A \\ -I \end{pmatrix} x \le \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix}\end{aligned}$$

5.11 Bemerkung
Wir beachten, daß die Übergänge jeweils zu Dimensionserhöhungen führen.

5.12 Bemerkung
Wir werden in Zukunft immer nur den dritten Typ betrachten. Der zulässige Bereich der Lösung ist $\{x \in \mathbb{R}^n, Ax = b, x \geq 0\}$. Dabei ist n die Dimension der Lösung und m die Anzahl der Gleichungsnebenbedingungen.
Es ist außerdem sinnvoll anzunehmen, daß $n \geq m$ ist.

5.13 Beispiel

1. $n = 2, m = 1$. Dann ist der zulässige Bereich der Teil einer Geraden, der im ersten Qudranten eines Koordinatensystems verläuft. Dieser kann beschränkt, unbeschränkt oder gar leer sein.
2. $n = 3, m = 1$. Dann ist der zulässige Bereich eine Ebene, die die Seitenfläche eines Polyeders (auch *Simplex* genannt) ist. Der zulässige Bereich kann allerdings auch unbeschränkt oder leer sein.
3. $n = 3, m = 2$. Dies ist der Schnitt zweier Ebenen, der im Allgemeinen eine Gerade ist.

Wir beachten, daß die Ecken (bzw. Endpunkte) in unseren Beispielen höchstens m von Null verschiedene Komponenten haben. Dies motiviert die allgemeine Definition, was wir in höheren Dimensionen als Ecke verstehen.

5.14 Definition (Ecke)
$x \in \mathbb{R}^n$ heißt eine *Ecke*, wenn die folgenden Eigenschaften erfüllt sind:

1. x ist zulässig, d.h. $Ax = b$ und $x \geq 0$.
2. x hat höchstens m verschiedene von Null verschiedene Komponenten.

5.3 Simplex-Verfahren

5.15 Idee
Wir starten auf einer Ecke und suchen uns eine Nachbar-Ecke mit einer kleineren Zielfunktion. Dann beginnen wir von vorn, solange bis wir in einem Minimum stecken.

5.16 Historische Notiz
Der Algorithmus geht auf Dantzig, 1946, zurück.

5.17 Konstruktion (Herleitung der Vorgehensweise)
Sei x eine Ecke, also $Ax = b$ und $x \geq 0$ und mindestens $n - m$ Nullkomponenten, d.h. nach einer Koordinatentransformation können wir schreiben $x = \begin{pmatrix} x_B \\ 0 \end{pmatrix}$, wobei $0 \leq x_B \in \mathbb{R}^m$ ist. Durch Aufspaltung von A erhalten wir:

$$b = Ax = \begin{pmatrix} B & N \end{pmatrix} \cdot \begin{pmatrix} x_B \\ 0 \end{pmatrix} = Bx_B \quad \text{mit } B \in \mathbb{R}^{m \times m} \text{ invertierbar}$$

Die Kostenfunktion in der Ecke ist $c^T x = \begin{pmatrix} c_B^T & c_N^T \end{pmatrix} \cdot \begin{pmatrix} x_B \\ 0 \end{pmatrix} = c_B^T x_B$ mit $c_B \in \mathbb{R}^m$.
Wir suchen nun eine andere Ecke mit geringeren Kosten. Wir betrachten anstelle des oben defineirten x nun die Kosten an einer Stelle $x' = \begin{pmatrix} x'_B \\ x'_N \end{pmatrix}$. Wegen $Ax' = b \iff Bx'_B + Nx'_N = b \iff x'_B = B^{-1}b - B^{-1}Nx'_N$ und $B^{-1}b = x_B$ gilt dann:

$$x' = \begin{pmatrix} x'_B \\ x'_N \end{pmatrix} = \begin{pmatrix} x_B - B^{-1}Nx'_N \\ x'_N \end{pmatrix}$$

Die Differenz der Kostenfunktion sind dann:

$$c^T x' - c^T x = c_B^T(x'_B - x_B) + c_N^T x'_N = -c_B^T B^{-1} N x'_N + c_N^T x'_N = \underbrace{(c_N^T - c_B^T B^{-1} N)}_{=:r^T \in \mathbb{R}^{n-m}} x'_N$$

$$= \sum_{k=1}^{n-m} r_k \cdot \underbrace{x'_{N,k}}_{\geq 0}$$

Falls $r \geq 0$ ist, dann nehmen die Kosten bei keiner Wahl von $x'_N \geq 0$ ab, d.h. die Ecke x war bereits optimal.

Falls umgekehrt die k-te Komponenten $r_k < 0$ ist, dann nehmen die Kosten ab, wenn die k-te Komponente von x'_N zunimmt. Wir wählen x'_N in Richtung der Komponenten mit der stärksten Abnahme. Wenn also $r_i = \min_j\{r_j < 0\}$ und $x'_{N,i} = \xi \geq 0$. Dann ist die Steigung $x'_N = \xi \cdot e_i$ zu wählen, d.h. $c^T x' - c^T x = r_i \cdot \xi < 0$.

Wir wählen dabei ξ möglichst groß, denn je größer ξ ist, desto stärker ist die Kostenreduktion. Dabei sind aber noch die Nebenbedingungen $Ax' = b, x' \geq 0$ zu erfüllen. Es gilt:

$$0 \leq x'_B = x_B - B^{-1}Nx'_N = x_B - \underbrace{B^{-1}Ne_i}_{=:v \in \mathbb{R}^m} \cdot \xi$$

Wir betrachten nun hierzu zwei Fälle:

$v \leq 0$**:** Dann ist $x'_B \geq 0$ für jede Wahl von $\xi > 0$ erfüllt. Wir lassen dann $\xi \to \infty$, damit ist die Kostenfunktion unbeschränkt, also gibt es keine Lösung und wir sind fertig.

$v_k > 0$**:** Dann wählen wir

$$\xi := \min_{k:\, v_k > 0} \left\{ \frac{x_{B,k}}{v_k} \right\} \quad \Rightarrow \quad x'_B = x_B - v\xi \geq 0$$

und (mindestens) eine Komponente von x'_B ist Null. Dann gilt:

$$x' = \left(x'_1, \ldots, x'_{j-1}, 0, x'_{j+1}, \ldots, x'_m, 0, \ldots, 0, \xi, 0, \ldots, 0\right)^T$$

Wir haben also eine neue Ecke gefunden mit geringeren Kosten, tauschen dann die Nullzeile an der Stelle j mit ξ aus und beginnen von vorn.

5.18 Algorithmus (Simplex-Schritt)
Sei $x = (x_B, 0)^T$ mit $x_B \in \mathbb{R}^m$, $A = \begin{pmatrix} B & N \end{pmatrix}$ und $c^T = \begin{pmatrix} c_B^T & c_N^T \end{pmatrix}$. Dann rechnen wir:

1. Wir setzen $r^T := c_N^T - c_B^T B^{-1} N$
2. Falls $r \geq 0$, dann ist x bereits optimal. Sonst bestimmen wir $r_i := \min_k\{r_k < 0\}$.
3. Wir setzen $v := B^{-1} \cdot N \cdot e_i$, wobei $N \cdot e_1$ die i-te Spalte von N ist.
4. Falls $v \leq 0$, dann ist die Kostenfunktion unbeschränkt, also gibt es keine Lösung. Sonst bestimmen wir j so, daß $\frac{x_j}{v_j} = \min_{k:\, v_k > 0} \left\{ \frac{x_k}{v_k} \right\} =: \xi \geq 0$ ist.
5. Wir ersetzen nun x_k durch $x_k - \xi v_k$ für $k = 1, \ldots, m$ (außer $k = j$, dort setzen wir $x_j := \xi$). Dann vertauschen in der Matrix A und im Vektor c^T die j-te Spalte mit der $(m+i)$-ten Spalte und beginnen wieder beim ersten Schritt.

5.19 Bemerkung
Es könnten ausgeartete Ecken auftreten, d.h. Ecken mit mehr als $n - m$ Nullkomponenten. Dann kann $\xi = 0$ sein (und zwar, falls $x_j = 0$ und $v_j > 0$). Dann wäre die neue Ecke die alte und wir würden feststecken. Dies kommt jedoch wegen Rundungsfehlern nicht vor.

5.20 Bemerkung (Numerische lineare Algebra des Simplex-Schrittes)

1. Wir berechnen $c_B^T B^{-1} N$ durch das Lösen des LGS $B^T u = c_B$ ist für $u^T := c_B^T B^{-1}$ und berechnen danach $r := c_N - N^T u$, ebenso erhalten wir v durch Lösen von $Bv = N \cdot e_i$.

2. Im ersten Schritt berechnen wir eine LR-Zerlegung von B mit $PB = LR$. Dann ersetzen wir die j-te Spalte von B durch die i-te Spalte von N, d.h. $B' = B \cdot \Gamma$ mit

   $$\Gamma e_\ell = e_\ell \; \forall 1 \leq \ell \leq j-1, \qquad \Gamma e_\ell = e_{\ell+1} \; \forall j+1 \leq \ell \leq m-1, \qquad \Gamma e_m = (v_1, \ldots, v_m)$$

   Dann ist $v_j > 0$ nach Konstruktion. Damit sind Γ und B' invertierbar.
   Es ist dann $B'v' = d \iff B\Gamma v' = d \iff Bw = d$ und $v' = \Gamma^{-1} w$, denn $v'_k = w_k - \frac{v_k}{v_j} w_j$ für $k \neq j$ und $v'_j = \frac{w_j}{v_j}$.
   Wir speichern die LR-Zerlegung von B und v_j für weitere Schritte.
   Falls es mehr als m Simplex-Schritte gibt, dann ist:
   $B\Gamma_1 \cdot \ldots \cdot \Gamma_m = \tilde{B}$ die neue LR-Zerlegung.

3. Wir führen einen Spaltentausch durch, evtl. über ein Zeigerfeld, d.h.:

   $$\begin{pmatrix} B & N \end{pmatrix} = \left(\begin{array}{ccc|ccc} A_{i_1} & \ldots & A_{i_m} & A_{i_{m+1}} & \ldots & A_{i_n} \end{array} \right)$$

   und i_k ist die ursprüngliche Position der gegenwärtigen Spalte k.

Für die Zerlegung in einem Schritt brauchen wir (falls $PB = LR$ bereits bekannt ist) $m^2 + (m-j)m \leq 2m^2$ Operationen.

5.21 Bemerkung
Bisher haben wir beschrieben, wie wir von einer Ecke zur nächsten gehen (zweite Phase).
Es ist also noch zu klären, wie wir eine Anfangsecke so bestimmen, daß $Ax = b, x \geq 0$ mit $n - m$ Nullkomponenten (erste Phase).

5.22 Idee (Anfangseckensuche)
Wir führen m neue Variablen ein: $x_{n+1}, \dots, x_{n+m}$ und betrachte den Vektor $\bar{x} = \begin{pmatrix} x_1 \\ \dots \\ x_{n+m} \end{pmatrix}$.
Weiter führen wir $\bar{c}^T := \begin{pmatrix} 0 & 1^T \end{pmatrix}$ mit $1^T = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^m$ und $\bar{A} := \begin{pmatrix} A & I_m \end{pmatrix}$ ein.
Wir nehmen o.B.d.A. an, daß $b \geq 0$ ist (falls dies nicht der Fall ist, multiplizieren wir die bösen Komponente mit (-1)).

5.23 Bemerkung (Hilfsproblem)
Wir möchten nun das Hilfsproblem $\bar{A}\bar{x} = b$ mit $\bar{x} \geq 0$ unter $\bar{c}^T\bar{x} = \min$ lösen.
Hier ist der Minimalwert 0 erreicht, falls $x_{n+1}, \dots, x_{n+m} = 0$ sind. Dann ist $Ax = b, x \geq 0$ und x hat mindestens $n-m$ Nullkomponenten, denn $\bar{x}$ hat $(n+m)-m = n$ Nullkomponenten.

Wir müssen noch zeigen, was die Anfangsecke für das Hilfsproblem ist. Dies ist leicht zu finden. Hierzu wählen wir $\bar{x} = \begin{pmatrix} 0 \\ b \end{pmatrix}$. Dann gilt:

$$\bar{A}\bar{x} = \begin{pmatrix} A & I \end{pmatrix} \begin{pmatrix} 0 & b \end{pmatrix} = b$$

und damit ist $\bar{x}$ zulässig.

5.24 Bemerkung
Hiermit haben wir nun den Algorithmus mitsamt Implementierung fertiggestellt. Wir müssen nun nur noch zeigen, daß die Minima wirklich an den Ecken auftreten, was wir für die Konstruktion der Anfangsecke benötigen. Diesen Beweis führen wir mit dem gerade konstruierten Algorithmus, jedoch ohne dabei einen Ringschluß mit dem Hilfsproblem zu erzeugen.

5.25 Satz (Minima an Ecken)
Die lineare Optimierungsaufgabe $Ax = b, x \geq 0$ und $c^T x = \min$ habe eine Lösung. Dann gibt es eine Ecke, die Lösung ist.

Beweis
Sei $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ im zulässigen Bereich, also $Ax = b$ und $x \geq 0$. Wir spalten A nun auf in $A = \begin{pmatrix} B & N \end{pmatrix}$ mit B invertierbar und $\operatorname{rg} A = m$.

Wir betrachten nun einen anderen Vektor $x' = \begin{pmatrix} x'_B \\ x'_N \end{pmatrix}$ und rechnen:

$$Ax = b = Ax' \iff A(x' - x) = 0 \iff B(x'_B - x_B) + N(x'_N - x_N) = 0$$

Damit gilt: $x'_B - x_B = -B^{-1}N(x'_N - x_N)$.
Die Kosten ändern sich dann folgendermaßen:

$$c^T x' - c^T x = \underbrace{(c_N^T - c_B^T B^{-1} N)}_{=r^T}(x'_N - x_N) = \sum_k r_k(x'_{N,k} - x_{N,k})$$

Wir nehmen an, daß $x_{N,i} > 0$ ist. Dann gibt es zwei Fälle (die aus dem Simplex-Algorithmus ergeben):

$r_i < 0$**:** Die Kosten werden kleiner, wenn $x'_{N,i}$ möglichst groß werden: Entweder es existiert keine Lösung, oder eine Komponente von x'_B wird Null.

$r_i \geq 0$**:** $x'_{N,i}$ ist dann möglichst klein zu wählen: Entweder ist $x'_{N,i} = 0$ oder eine Komponente von x'_B wird Null.

In beiden Fällen haben wir mehr Nullkomponenten von x' als davor. Wir fahren dann solange fort, bis wir genügend Nullen haben. □

5.26 Bemerkung
Falls eine Lösung existiert, dann findet das Simplex-Verfahren diese bereits nach endlich vielen Schritten, denn es existieren auch nur endlich viele Ecken.

Der statistische Erwartungswert liegt bei $\frac{3m}{2}$ Schritten, im ungünsten Fall müssen alle Ecken abgelaufen werden, d.h. 2^{m-1} Schritte sind nötig.

5.4 Dualität

5.27 Wiederholung
Wie im zweiten Kapitel gesehen, gibt es ein lineares Programm, welches wir zukünftigt „primales Programm“ nennen möchten. Hierbei hatten wir:

$$\begin{array}{c|c} A & b \\ \hline c^T & \end{array}$$

Dabei hatten wir verschiedene Probleme:

I: $Ax \leq b$ und $c^T x = \min$

II: $Ax \leq b$, $x \geq 0$ und $c^T x = \min$

III: $Ax = b$, $x \geq 0$ und $c^T x = \min$

5.28 Definition (Duales Programm)
Wir betrachten nun das duales Programm:

$$\begin{array}{c|c} A^T & c \\ \hline b^T & \end{array}$$

Dann haben wir die verschiedenen Probleme:

I* $A^T y = c$, $y \leq 0$ und $b^T y = \max$

II* $A^T y \leq c$, $y \leq 0$ und $b^T y = \max$

III* $A^T y \leq c$, $b^T y = \max$

5.29 Bemerkung
Wir sehen, daß z.B. I* zu III äquivalent ist und natürlich I*,II*,III* zueinander äquivalent sind. Wir werden in Zukunft mit dem Problem III* arbeiten.

5.30 Definition (Zulässiger Bereich)
Wir sagen, $x \in \mathbb{R}^n$ ist zulässig, falls $Ax = b$ und $x \geq 0$. $y \in \mathbb{R}^m$ ist zulässig, falls $A^T y \leq c$.

5.31 Satz (Schwache Dualität)
Für alle zulässigen x und y gilt:

$$c^T x \geq y^T b$$

Beweis
Da x zulässig ist, ist $Ax = b$ und $x \geq 0$ und da y zulässig ist, gilt $A^T y \leq c$. Damit gilt:

$$y^T b = y^T (Ax) = (y^T A)x \leq c^T x$$

□

5.32 Korollar (optimale Lösungen)
Falls $c^T x = y^T b$ für zulässige x und y, dann sind bereits x und y optimal.

Beweis
Für alle zulässigen $\eta \in \mathbb{R}^m$ gilt: $y^T b = c^T x \geq \eta^T b$. Dann gilt genau dann $y^T b = \max$, wenn y optimal für das duale Problem ist.

Analog erhalten wir für alle zulässigen $\zeta \in \mathbb{R}^n$ dann $c^T x = y^T b \leq c^T \zeta$, was äquivalent dazu ist, daß x optimal ist. □

5.33 Korollar
Falls das Infimum im primalen Problem $-\infty$ ist, dann der zulässige Bereich des duales Problems leer.

Umgekehrt gilt, falls das Supremum des dualen Problems $+\infty$ ist, dann ist der zulässigen Bereich des primalen Problems leer.

Beweis
Nach der schwachen Dualität gilt: $-\infty = \inf\{c^T x : x \text{ zulässig}\} \geq y^T b \quad \forall y$ zulässig. Doch so ein y kann es nicht geben.

Analog erhalten wir $+\infty \leq c^T x \quad \forall x$ zulässig. Also gibt es kein zulässiges x. □

5.34 Satz (Dualitätssatz)
Seien $x \in \mathbb{R}^n$ und $y \in \mathbb{R}^m$ zulässig. Dann gilt:

$$x, y \text{ optimal} \iff c^T x = y^T b$$

Beweis
Die Richtung „$\Leftarrow$“ wurde bereits gezeigt. Wir müssen also noch unter der Voraussetzung, daß x optimal ist, zeigen, daß $\exists y$ zulässig mit $c^T x = y^T b$.

Wir verwenden das Simplex-Verfahren für $A = \begin{pmatrix} B & N \end{pmatrix}$, $c^T = \begin{pmatrix} c_B^T & c_N^T \end{pmatrix}$ und $b = Ax = Bx_B$:

$$x = \begin{pmatrix} x_B \\ 0 \end{pmatrix} \text{ optimal} \iff r^T := c_N^T - c_B^T B^{-1} N \geq 0$$

Dann ist $c^T x = c_B^T x_B = c_B^T B^{-1} b = y^T b$ für $y^T := c_B^T B^{-1} \in \mathbb{R}^m$. Wir müssen nun noch zeigen, daß dieses definierte y zulässig ist. Wir rechnen:

$$\begin{aligned} y^T A &= c_B^T B^{-1} A = c_B^T B^{-1} \begin{pmatrix} B & N \end{pmatrix} \\ &= \begin{pmatrix} c_B^T & c_B^T B^{-1} N \end{pmatrix} = \begin{pmatrix} c_B^T & c_N^T - r^T \end{pmatrix} \end{aligned}$$

da $r^T \geq 0$ ist, erhalten wir schließlich:

$$\leq \begin{pmatrix} c_B T & c_N^T \end{pmatrix} = c^T$$

Damit ist y zulässig. □

5.35 Korollar
Falls das primale Problem eine Lösung hat, dann hat das duale Problem auch eine Lösung. Insbesondere gilt, daß das Minimum des primalen Problems das Maximum des dualen Problems ist.

5.36 Bemerkung
Das Simplex-Verfahren berechnet immer die Lösung des dualen Problems automatisch mit (denn es muß ohnehin $B^T y = c_B$ gelöst werden).

5.37 Korollar (Optimalitätskriterium)
Seien x, y zulässig. Dann sind x und y genau dann optimal, wenn für alle $j = 1, \ldots, n$ gilt: $x_j = 0$ oder $(A^T y)_j = c_j$.

Beweis
⇐: Sei $(y^T A)_j = c_j^T$ für $x_j \neq 0$. Dann gilt:

$$(y^T A - c^T)x = 0 \iff 0 = y^T Ax - c^T x = y^T b - c^T x \iff y^T b = c^T x$$

Da x und y zulässig sind, erhalten wir mit dem Dualitätssatz, daß x und y optimal sind. Diese Rechnung läuft allerdings genau gleich, wenn ein $x_j = 0$ ist.

⇒: Seien x, y optimal. Dann ist:

$$y^T b = c^T x \quad \Rightarrow \quad \underbrace{(c^T - y^T A)}_{\geq 0} \underbrace{x}_{\geq 0} = 0$$

Damit gilt für jedes j, daß dann $(c^T - y^T A)_j = 0$ oder $x_j = 0$. □

5.38 Beispiel
In der freien Marktwirtschaft geht es um Gleichgewichtsbedingungen.

Zur Herstellung der Produkte $j = 1, \ldots, n$ wird die Menge a_{ij} der Rohstoffe $i = 1, \ldots, m$ benötigt. Der Wert des Produkts j ist c_j und die erzeugte Menge von j ist $x_j \geq 0$. Der Preis des Rohstoff i ist $y_i \geq 0$ und die Menge Rohstoffs i ist b_i.

Damit erhalten wir als Nebenbedingung $Ax \leq b$, d.h.

$$\sum_j a_{ij} x_j \leq b \quad \forall i$$

Die Einnahmen des Betriebs sind

$$\sum_j c_j x_j = c^T x$$

und die Ausgaben

$$\sum_j \left(\sum_i a_{ij} y_i \right) x_j = y^T Ax$$

Demnach ergibt sich für den Gewinn dann $c^T x - y^T Ax = (c^T - y^T A)x$. Dieser soll maximiert werden.

Aus der Perspektive der Rohstofflieferanten ergibt sich die Taktik, daß solange der Gewinn des Betriebs positiv ist, die Preise zu erhöhen (Denn da der Betrieb keinen Verlust macht, wird er weiterhin einkaufen).

Damit erhalten wir als Nebenbedingung $y^T A \geq c^T$ mit $y \geq 0$ für das duale Problem des Rohstofflieferanten.

Diese beiden Sichtweisen spiegeln sich in den Dogmen der Betriebe bzw. Lieferanten wider:

1. Falls $(c^T - y^T A)_j < 0$, dann setze $x_j := 0$. Hier sind die Kosten für die Produktion der j-ten Produkts größer als der Verkaufserlös, also wird die Produkt hiervon eingestellt.

2. Falls $(Ax - b)_i < 0$, dann setze $y_i := 0$. Hier ist das Angebot für den Rohstoff i größer als die Nachfrage, dann ist dieser Rohstoff umsonst zu haben.

Mit dem Dualitätssatz gilt also:
$c^T x = \max$ und $Ax \leq b$ mit $x \geq 0$ sowie $b^T y = \min$ und $A^T y \geq c$ mit $y \geq 0$.

Wir interpretieren dies, daß bei der freien Marktwirtschaft dann die Wertschöpfung $c^T x$ maximiert wird.

5.39 Bemerkung (Auswirkungen kleiner Störungen)
Wir haben $c^T x = \min$ und $Ax = b$ mit $x \geq 0$ gegeben.
Wir möchten wissen, wie sich die minimalen Kosten $c^T x$ ändern, wenn wir b ändern?
Wir sehen, daß wenn anstelle von b nun $b + \Delta b$ mit Δb „genügend klein“, dann wird x zu $x + \Delta x$.

Beim dualen Problem $A^T y \leq c$, $(b + \Delta b)^T y = \max$ ändert sich nicht der zulässige Bereich, also bleibt die Lösung y des neuen dualen Problems in derselben Ecke, falls Δb klein genug ist.

Wir betrachten nun die Änderungen der Kosten und benutzen den Dualitätssatz:

$$c^T(x + \Delta x) - c^T x = (b + \Delta b)^T y - b^T y = (\Delta b)^T y$$

Wir interpretieren die duale Variable y als Änderung der Kosten geteilt durch die Änderung des Angebots.

In Termen (wo y der Preis war) des obigen Beispiels ergibt sich damit:

$$\text{Preis} = \frac{\text{Änderung der Kosten}}{\text{Änderung des Angebots}}$$

5.5 Vorbereitung auf Karmarkar, Projektion, Umskalierung

5.40 Motivation
Wir werden nun in diesem und den nächsten Kapiteln den Algorithmus von Karmarkar betrachten, der 1984 in Artikel „A new polynomial-time algorithm for linear programming“ zu diesem Algorithmus veröffentlicht hat.

Beim Simplex-Verfahren hatten wir im schlimmsten Fall eine exponentielle Laufzeit, wobei wir bei diesem Algorithmus auf jeden Fall eine polynomielle Laufzeit haben.

In diesem Kapitel werden wir den Algorithmus vorbereiten.

Wir suchen x mit $Ax = b$, $x \geq 0$ und $c^T x = \min$.

Sei x^0 im Inneren des zulässigen Bereits gegeben (also $x_j > 0$ für $j = 1, \ldots, n$). Wir suchen nun ein zulässiges $x^1 := x^0 + \Delta x$ mit geringeren Kosten. Hierzu gibt es zwei Ideen, auf die wir im folgenden eingehen werden.

5.41 Idee (Projektion)
Wir möchten Δx in Richtung des steilsten Abstiegs der Kostenfunktion wählen, d.h. Δx soll ein negatives Vielfaches von c sein, denn es gilt:

$$c^T \Delta x = \|c^T\| \cdot \|\Delta x\| \cos \angle(c, \Delta x)$$

und dies ist dann am kleinsten, wenn der Kosinus -1 ist

Wir benötigen aber wieder die zulässigkeit von x^1, insbesondere $Ax^1 = b$, d.h. es muß gelten: $A\Delta x = 0$, also $\Delta x \in \mathrm{Ker}(A)$. Daher ist im Allgemeinen Δx nicht wie oben wählbar. Wir betracten nun hierzu eine orthogonale Projektion auf $\mathrm{Ker}(A)$, also:

$$\Delta x := -\xi v$$

wobei ξ ein (noch zu bestimmender) Skalisierungsfunktion ist und v die orthogonale Projektion von c auf $\mathrm{Ker}(A)$, d.h. $v \in \mathrm{Ker}(A)$ und $v - c \perp \mathrm{Ker}(A)$ (d.h. $v - c \in \mathrm{Ker}(A)^\perp = \mathrm{Bild}(A^T)$). Insbesondere gibt es $\lambda \in \mathbb{R}^m$ mit $v - c = -A^T\lambda$. In Matrixform bedeutet dies also:

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

O.B.d.A. können wir annehmen, daß die Zeilen der Matrix A linear unabhängig sind (sonst lassen wir linear abhängige Zeilen weg, die nur redundante Informationen an unser Problem lösen). Damit sind v und λ immer eindeutig bestimmt.

Wir beschäftigen uns nun mit der Wahl des Skalierungsfaktors ξ. Wir dürfen hier nicht zu weit gehen, da wir sonst aus dem zulässigen Bereich herausfallen.

Wir beachten zunächst, daß gilt $c = v + A^T\lambda$. Also gilt für die Änderung der Kosten, da $v \in \mathrm{Ker}(A)$ ist:

$$c^T \Delta x = (v + A^T\lambda)^T(-\xi v) = -\xi v^T v - \xi \lambda^T A v = -\xi |v|^2$$

Wir haben 3 Möglichkeiten, was v sein kann.

1. $v = 0$:
   Dann gibt es ein λ mit $c = A^T\lambda$, also ist $c^T x = \lambda^T A x = \lambda^T b$ für alle zulässigen x. Also sind alle zulässigen x optimal und wir sind fertig.
2. Für alle ξ ist $x^1 = x^0 - \xi v \geq 0$:
   Dann ist die Kostenfunktion innerhalb des zulässigen Bereichs unbeschränkt, also gibt es kein Minimum und wir sind fertig.

3. Es gibt ein $\xi^* > 0$ mit $x^0 - \xi^* v$ ist am Rand des zulässigen Bereichs, d.h. wir haben $x^0 - \xi^* v \geq 0$ und es gibt ein j mit $(x^0 - \xi^* v)_j = 0$.

   Wir wählen (analog zum Simplex-Verfahren) dann

   $$\xi^* := \min_{j:\, v_j > 0} \frac{x_j}{v_j}$$

   Wir stoppen allerdings kurz vor dem Rand, d.h. für ein $\alpha < 1$ setzen wir dann

   $$x^1 := x^0 - \alpha \xi^* v$$

5.42 Idee (Umskalierung)
Wir wählen eine Diagonalmatrix $D = \operatorname{diag}(d_1, \ldots, d_n)$ so, daß $Dx^1 = x^0$, d.h.

$$d_j := \frac{x_j^0}{x_0^1} > 0$$

Wir schreib nun unser Problem $Ax = b$, $x \geq 0$ und $c^T x = \min$ für $Dx =: \tilde{x}$ äquivalent in ein neues Problem um:

$$AD^{-1}\tilde{x}b, \qquad \tilde{x} \geq 0, \qquad c^T D^{-1} \tilde{x} = \min$$

Als Startpunkt für das umskalierte Problem wählen wir $\tilde{x}^1 = Dx^1 = x^0$.

Wir wenden nun die Projektionsschritt (aus der ersten Idee) auf das umskalierte Problem an:

$$\begin{pmatrix} I & D^{-1}A^T \\ AD^{-1} & 0 \end{pmatrix} \begin{pmatrix} \tilde{v} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} D^{-1}c \\ 0 \end{pmatrix}$$

bzw. wenn wir $\tilde{v} := Dv$ und $\tilde{\lambda} := \lambda$ setzen, dann erhalten wir äquivalent:

$$\begin{pmatrix} D^2 & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

Wir haben nun hier keine orthogonale Projektion mehr, sondern eine gewichtete Projektion. Genaugenommen ist dies aber die orthogonale Projektion des induzierte Skalarprodukts von D^2, d.h.

$$\langle v, w \rangle = v^T D^2 w = \tilde{v}^T \tilde{w}$$

Anschließend setzen wir dann (analog zur Idee der Projektion)

$$\tilde{x}^2 := \tilde{x}^1 - \alpha \xi^* \tilde{v}$$

bzw. (durch Reskalierung) ergibt sich:

$$x^2 = x^1 - \alpha \xi^* v$$

5.43 Idee (zweiter Schritt)
Wir wählen nun eine Diagonalmatrix D mit $Dx^2 = x^0$, machen die Projektion, usw.

5.44 Algorithmus (Umskalierungsalgorithmus)
Sei x^0 im Inneren des zulässigen Bereichs gegeben. Für $k = 0, 1, \ldots$ führen wir dann folgende Schritte durch:

1. Zur Matrix $M := \operatorname{diag}(d_1^2, \ldots, d_n^2)$ mit $d_j := \frac{x_j^0}{x_j^k} > 0$ lösen wir das LGS

$$\begin{pmatrix} M & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

2. Falls $v_j \leq 0$ ist für alle j, dann gibt es kein Minimum und wir wären fertig.

   Sonst wählen wir

$$\xi := \min_{j:v_j>0} \frac{x_j^k}{v_j}$$

   und setzen $\Delta x := -\alpha \xi v$ für ein geeignetes $\alpha < 1$ nahe bei Eins.

   Wir setzen anschließend:

$$x^{k+1} := x^k - \Delta x$$

5.45 Bemerkung
Wir beachten, daß dabei per Konstruktion die Kostenfunktion in jedem Schritt abnimmt.

5.46 Bemerkung (Lineare Algebra des Algorithmus)
Für den ersten Schritt haben wir drei Möglichkeiten, diesen auf Linearen-Algebra-Level durchzuführen:

1. Wir können eine Gauß-Elimation der Matrix $\begin{pmatrix} M & A^T \\ A & 0 \end{pmatrix}$ durchführen und erhalten als Lösung für λ das LGS:

$$AM^{-1}A^T\lambda = AM^{-1}c$$

   und für v erhalten wir:

$$Mv = c - A^T\lambda$$

   und damit ist das LGS entkoppelt.

   Für $AM^{-1}A^T$ können wir eine Cholesky-Zerlegung durchführen.

2. Wir können auch für $AM^{-1}A =: AD^{-1}D^{-1}A^T$ eine QR-Zerlegung durchführen in dem Sinne, daß

$$D^{-1}A^T = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

ist, wobei Q orthogonal und R eine invertierbare Dreiecksmatrix ist. Dann gilt:

$$AM^{-1}A^T = \begin{pmatrix} R^T & 0 \end{pmatrix} Q^T Q \begin{pmatrix} R \\ 0 \end{pmatrix} = R^T R$$

Dann ist $R^T R\lambda = AM^{-1}c$ zu lösen.

3. Wir können mit iterativen Verfahren, etwa dem cg-Verfahren, den ersten Schritt berechnen.

   Da üblicherweise A dünn besetzt ist (und M bzw. M^{-1} sowieso), bietet sich das Lösung durch das cg-Verfahren an. Hierdurch werden keine Nichtnullelemente aufgefüllt.

5.47 Bemerkung (Auffinden des Startwerts)
Wir führen nun eine zusätzliche Spalte in A ein, d.h.

$$\hat{A} := \begin{pmatrix} A & b - Ae \end{pmatrix} \quad \text{mit } e := \begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix} \in \mathbb{R}^n \quad \text{und } \hat{e} := \begin{pmatrix} e \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1}$$

Dann gilt:

$$\hat{A}\hat{e} = Ae + b - Ae = b$$

Wir möchten dies nun auf ein Problem der Art $\hat{A}\hat{x} = b$, $\hat{x} \geq 0$, $\hat{c}^T\hat{x} = \min$ zurückführen. Hierfür wählen wir:

$$\hat{c} := \begin{pmatrix} c \\ c_{n+1} \end{pmatrix} \quad \text{mit } c_{n+1} \text{ sehr groß}$$

Dann wäre die letzte Komponente von $\hat{x}$ Null, um die Größe von c_{n+1} auszugleichen, womit wir wieder die letzte Komponente vergessen können.

Als Startwert für das modifizierte Problem wählen wir $x^0 := \hat{e}$ als Startwert, der im Inneren des zulässigen Bereichs liegt.

5.48 Bemerkung (Abbruchbedingung)
Falls Δx „klein“ ist, dann sind wir in der Nähe einer Ecke und dann springen wir an die nächstgelegene Ecke vom aktuell gewählten x^k. Anschließend kontrollieren wir (mit dem Simplex-Algorithmus) die Ecke auf Optimalität.

5.6 Algorithmus von Karmarkar

5.49 Bemerkung (Spezielle Formulierung des Problems)
Wir betrachten nun eine spezielle Formulierung unsereres Problems, für den wir den Algorithmus von Karmarkar betrachten. Wir betrachten nun in drei Schritten den Umformulierungsprozess:

1. $Ax = 0$ mit $A \in \mathbb{R}^{n\times n}$ und $x \in \mathbb{R}^n$
2. $\sum_{i=1}^n x_i = 1$
3. $x \geq 0$
4. $c^T x = \min$ mit $c \in \mathbb{R}^n$, wobei das Minimum 0 ist.
5. $e := \frac{1}{n}\begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix}$ ist zulässig.

5.50 Bemerkung (Erreichen der speziellen Formulierung)
Wir zeigen nun, daß wir die allgemeine Bedingung immer auf die spezielle Formulierung zurückführen können:

1. Wir kombinieren das primale und duale Problem und gehen aus von $Ax = b$, $x \geq 0$ und $c^T x = \min$. Das duale Problem ist dann $A^T y \leq c$ bzw. $A^T(y_+ - y_-) + z = c$ und $y_+, y_- \geq 0$.

   Mit dem Dualitätssatz erhalten gilt: $c^T x = \min \iff b^T y = c^T x$. Damit ist zum Lösen des Optimierungsproblems das folgende LGS äquialent:

   $$\begin{pmatrix} A & 0 & 0 & 0 \\ 0 & A^T - A^T & I & \\ c^T & -b^T & b^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y_+ \\ y_- \\ z \end{pmatrix} = \begin{pmatrix} b \\ c \\ 0 \end{pmatrix}$$

   mit $x, y_+, y_-, z \geq 0$.

   Dieses Problem ist nun von der Form $\bar{A}\bar{x} = \bar{b}$ und $\bar{x} \geq 0$.

2. Wir setzen nun $x_i = \frac{z_i}{z_{n+1}}$ und wählen z_{n+1} so, daß $\sum_{i=1}^{n+1} z_i = 1$ ist. Dann erhalten wir:

   $$z_{n+1} \sum_{i=1}^{n} x_i + z_{n+1} = 1, \quad \text{also: } z_{n+1} = \frac{1}{1 + \sum_{i=1}^n x_i}$$

   Wir erhalten damit eine Bijektion zwischen

   $$\{(x_1, \dots, x_n) : x_i \geq 0\} \cong \left\{(z_1, \dots, z_{n+1}) : z_i \geq 0,\ \sum_{i=1}^{n+1} z_i = 1\right\}$$

Damit erhalten wir als LGS

$$Ax = b \iff A\begin{pmatrix} z_1 \\ \dots \\ z_n \end{pmatrix} = bz_{n+1} \iff \begin{pmatrix} A & -b \end{pmatrix}\begin{pmatrix} z_1 \\ \dots \\ z_n \\ z_{n+1} \end{pmatrix} = 0$$

Dieses neue Problem ist nun, wenn wir noch eine Nullzeile von A ergänzen, von der Form $Ax = 0$, $x \geq 0$ und $\sum_{i=1}^{n} x_i = 1$.

3. Wir führen nun ein neue Variable $y \in \mathbb{R}$ ein, welcher minimal sein soll und es soll gelten $Ax = (A\hat{e})y = 0$ mit $\hat{e} = \begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix} \in \mathbb{R}^n$ und weiter $\sum_{i=1}^{n} x_i + y = 1$ mit $x \geq 0$ und $y \geq 0$. Dann wird das Minimum bei $y = 0$ angenommen.

   Mit der Bedingung $Ax - (A\hat{e})y = 0$ ist $e = \frac{1}{n+1}\begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1}$ ein zulässiger Punkt.

Damit haben wir das allgemeine Optimierungsproblem auf unser spezielles Problem umgeformt.

5.51 Motivation
Beim Algorithmus von Karmarkar werden wir nun eine solche Transformation des zulässigen Bereichs betrachten, daß wir den Simplex in einen Simplex überführen. Dies geschieht durch eine nichtlineare Abbildung.

Beim Skalierungsalgorithmus wurde dabei das Dreieck gedreht und verzerrt. Dies wird hier nicht mehr passieren.

5.52 Notation
Wir bezeichnen mit Δ den zulässigen Bereich, d.h.

$$\Delta := \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1 \; x_i \geq 0 \right\}$$

5.53 Konstruktion (Projektive Transformation des Simplex)
Sei a ein innerer Punkt in Δ, d.h. $\sum a_i = 1$ und $a_i > 0$. Wir betrachten nun die Abbildung $T_a : \Delta \to \Delta$ mit

$$x \mapsto \frac{Dx}{\sum_{i=1}^{n} \frac{x_i}{a_i}} = T_a(x) \quad \text{mit } D = \operatorname{diag}\left(\frac{1}{a_i}\right)$$

Die Abbildung T_a hat die folgenden Eigenschaften:

1. $T_a(a) = e$ mit $e = \frac{1}{n}\begin{pmatrix} 1 \\ \dots \\ 1 \end{pmatrix}$.

2. T_a ist wohldefiniert und bijektiv mit der Umkehrabbildung

$$T_a^{-1}(\tilde{x}) = \frac{D^{-1}\tilde{x}}{\sum_{i=1}^n a_i \tilde{x}_i}$$

3. $T_a(\partial\Delta) = \partial\Delta$.

4. Wenn wir $\tilde{x} = T_a(x)$ schreiben, dann ist

   a) $Ax = 0 \iff AD^{-1}\tilde{x} = 0$,

   b) $c^T x = 0 \iff c^T D^{-1}\tilde{x} = 0$,

   c) $c^T x > 0 \iff c^T D^{-1}\tilde{x} > 0$.

5.54 Konstruktion (Iteration von Karmarkar)
Wir bezeichnen $x^{k+1} := \mathcal{K}(x^k)$.

$$\begin{array}{ccc} x^k & \xrightarrow{T_{x^k}} & \tilde{x}^k = e \\ & & \Big\downarrow * \\ x^{k+1} & \xleftarrow[T_{x^k}^{-1}]{} & \tilde{x}^{k+1} = e + \Delta\tilde{x} \end{array}$$

wobei im Schritt (*) $c^T D^{-1}\tilde{x} = \min$, $AD^{-1}\tilde{x} = 0$, $\sum \tilde{x}_i = 1$ und $\tilde{x} \geq 0$ haben und dann $\Delta\tilde{x}$ in die negative Richtung der orthogonalen Projektion von $\tilde{c} = D^{-1}c$ auf $\operatorname{Ker}\begin{pmatrix} AD^{-1} \\ 1 \end{pmatrix} =: B$ wählen und bezeichnen diese Projektion mit P. Wir lösen dann $P\tilde{c} = \tilde{v}$ mit

$$\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \tilde{v} \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{c} \\ 0 \end{pmatrix}$$

Wir setzen dann

$$\Delta\tilde{x} := -\alpha r \frac{P\tilde{c}}{\|P\tilde{c}\|} \quad \text{also: } \tilde{x}^{k+1} = e - \alpha r \frac{P\tilde{c}}{\|P\tilde{c}\|}$$

mit einem Parameter $0 < \alpha < 1$ und wobei $r := \frac{1}{\sqrt{n(n-1)}}$ der Radius der größten Kreisscheibe im Simplex Δ ist.

5.55 Algorithmus
Wir wählen $x^0 := e$ und für $k = 0, 1, 2 \ldots$ berechnen wir $x^{k+1} = \mathcal{K}(x^k)$ bis das Abbruchkriterium erfüllt ist.

5.7 Konvergenz des Algorithmus von Karmarkar

5.56 Motivation
Wir gehen nun von den Voraussetzung wie im letzten Kapitel aus und werden nun die Konvergenz des Algorithmus zeigen, dessen Aufwand polynomial wächst.

5.57 Satz (Konvergenz des Karmarkar-Algorithmus)
Sei $x^0 = e$ und sei $x^{k+1} = \mathcal{K}(x^k)$ die durch den Karmarkar-Algorithmus definierte Folge (mit $\alpha = \frac{1}{2}\sqrt{\frac{n-1}{n}}$).

Falls das Problem eine Lösung hat (d.h. falls $\exists x$ mit $c^T x = 0$), dann so ist nach

$$k \geq \frac{10}{3} n\Big(\log n + 0,7q\Big)$$

Iterationen gilt die Rekudktion der Zielfunktion:

$$\frac{c^T x^k}{c^T x^0} \leq 2^{-q}$$

Beweis
Wir benötigen nun einige Hilfsmittel, um dies zu beweisen. □

5.58 Definition (Potentialfunktion)
Sei Δ der Simplex des zulässigen Bereichs. Dann definieren wir die *Potentialfunktion*:

$$\varphi : \Delta^\circ \to \mathbb{R} : \ \varphi(x) := \sum_{i=1}^{n} \log \frac{c^T x}{x_i} = n \log c^T x - \sum_{i=1}^{n} \log x_i$$

5.59 Hilfssatz
Sei $x^0 = e$ und sei $x^{k+1} = \mathcal{K}(x^k)$ die durch den Karmarkar-Algorithmus definierte Folge (mit $\alpha = \frac{1}{2}\sqrt{\frac{n-1}{n}}$).

Falls das Problem eine Lösung hat, dann gilt für alle $k \in \mathbb{N}$, daß

$$\varphi(x^{k+1}) \leq \varphi(x^k) - \delta$$

mit $\delta > 0,3$.

Beweis
Wir benötigen hierfür noch einige Eigenschaften der Potentialfunktion, um dies beweisen zu können. □

5.60 Proposition (Eigenschaften der Potentialfunktion)

1. Falls x^* Lösung des Problems $c^T x^* = 0$ ist. Dann gilt für eine Folge $x \to x^*$ entlang einer Geraden in Δ°, daß
$$\varphi(x) \to -\infty$$
2. Falls $\bar{x} \in \partial\Delta$ ist mit $c^T \bar{x} > 0$ (also keine Lösung ist), dann gilt $\varphi(x) \to +\infty$ für $x \to \bar{x}$ längs einer Geraden in Δ°.
3. Die Potenentialfunktion ist invariant unter projektiven Transformationen:

Beweis

1. Für $x = x^* + tu$ mit $u_i > 0$ für jedes i mit $x_i^* = 0$. Für $t \searrow 0$ berechnen wir damit:
$$0 < c^T x = t \underbrace{c^T u}_{>0}$$
Wir rechnen nun:
$$\varphi(x^* + tu) = \sum_{i=1}^{n} \log\left(\frac{tc^T u}{x^* + tu_i}\right)$$
Das Argument des Logarithmus ist unabhängig von t, falls $x_i^* = 0$ ist. Sonst geht es gegen 0, falls $x_i^* > 0$ ist. Aber mindestens ein x_i^* muß positiv sein, da x^* normiert ist. Damit geht das Argument gegen Null und die Funktion damit gegen $-\infty$.
2. Wir schreiben analog zu oben $x = \bar{x} + tu$ und sehen $u_i > 0$, falls $\bar{x}_i = 0$. Für $t \searrow 0$ berechnen wir:
$$\varphi(\bar{x} + tu) = \sum_{i=1}^{n} \log\left(\frac{c^T \bar{x} + tc^T u}{\bar{x}_i + tu_i}\right)$$
Wegen $\bar{x} \in \partial\Delta$ gibt es ein i mit $\bar{x}_i = 0$. Der Zähler ist strikt positiv (wegen $c^T \bar{x} > 0$) und in dieser Komponente geht damit die Funktion $\varphi(\bar{x} + tu) \to \infty$.
3. Für $\tilde{x} = T_a(x)$, also
$$x = T_a^{-1}(\tilde{x}) = \frac{D^{-1}\tilde{x}}{\sum_{i=1}^n a_i \tilde{x}_i}$$
mit $D = \operatorname{diag}\left(\frac{1}{a_i}\right)$, definieren wir $\tilde{\varphi}$ durch:
$$\tilde{\varphi}(\tilde{x}) = \varphi(x)$$
Damit berechnen wir:
$$\tilde{\varphi}(\tilde{x}) = n \log\left(c^T \frac{D^{-1}\tilde{x}}{\sum_{i=1}^n a_i \tilde{x}_i}\right) - \sum_{i=1}^{n} \log\left(\frac{a_i \tilde{x}_i}{\sum_{j=1}^n a_j \tilde{x}_j}\right)$$
Für $\tilde{c}^T := c^T D^{-1}$ erhalten wir damit (mit den Logarithmusregeln):
$$= n \log \tilde{c}^T \tilde{x} - \sum_{i=1}^{n} \log \tilde{x}_i - \underbrace{\sum_{i=1}^{n} \log a_i}_{=\text{const}}$$

Damit gelten alle Eigenschaften. □

5.61 Bemerkung
Wir zeigen nun mit dem nächsten Lemma, daß der Projektionsschritt im Algorithmus von Karmarkar aufgefasst werden kann als Minimierung über eine Kreisscheibe.

5.62 Lemma
Es gilt:

$$\min_{\substack{\|x-e\|\leq\delta \\ Ax=0 \\ \sum_i x_i=1}} c^T x = c^T \left(e - \varrho \frac{Pc}{\|Pc\|} \right)$$

wobei P die orthogonale Projektion auf Ker $\begin{pmatrix} A \\ 1 \end{pmatrix}$ ist.

Beweis
Wir rechnen:

$$\begin{aligned} c^T(x-e) &= c^T P(x-e) \\ &= (Pc)^T(x-e) \end{aligned}$$

Cauchy-Schwarz:

$$\begin{aligned} &\geq -\|Pc\|\|x-e\| \\ &\geq -\varrho\|Pc\| \end{aligned}$$

Dabei gilt Gleichheit, falls $\frac{Pc}{\|Pc\|} = -\frac{x-e}{\|x-e\|}$ ist und $\|x-e\| = \varrho$ ist. Dies ist genau dann der Fall, wenn $x - e = -\varrho\frac{Pc}{\|Pc\|}$ ist. Damit gilt bereits die gewünschte Gleichheit. □

Beweis (des Hilfssatzes)

1. Wir müssen zeigen, daß $\varphi(x^{k+1}) \leq \varphi(x^k) - \delta$ ist mit $\delta > 0,3$ mit das gegebene α.

   Wir wissen bereits, daß gilt:

   $$\varphi(x^{k+1}) = \tilde{\varphi}(\tilde{x}^{k+1}) = \tilde{\varphi}(e - \alpha r \tilde{d})$$

   mit $\tilde{d} = \frac{P\tilde{c}}{\|P\tilde{c}\|}$ und $r = \frac{1}{\sqrt{n(n-1)}}$. Und es gilt:

   $$\varphi(x^k) = \tilde{\varphi}(\tilde{x}^k) = \tilde{\varphi}(e)$$

   Wir wissen, daß $\tilde{\varphi}$ dieselbe Form hat wie φ, aber mit $\tilde{c}$ anstelle von c.

   Wir lassen in der folgenden Rechnung, um die Notation zu erleichtern, alle Schlangen-Größen weg, rechnen aber in den geschlängelten (transformierten) Variablen.

2. Wir müssen also nun noch zeigen, daß gilt:

   $$\varphi(e - \alpha r d) \leq \varphi(e) - \delta$$

   mit $x(a\alpha) := e - \alpha r d$, $d := \frac{Pc}{\|Pc\|}$ und P ist die orthogonale Projektion auf Ker $\begin{pmatrix} A \\ 1 \end{pmatrix}$.

   Insbesondere ist wegen $d \in \text{Ker} \begin{pmatrix} A \\ 1 \end{pmatrix}$ dann $\sum_{i=1}^{n} d_i = 0$.

   Wir betrachten nun die Hilfsfunktion

   $$\psi(\alpha) := \varphi(x(\alpha)) - n\frac{c^T x(\alpha)}{c^T e}$$

   und mit dieser Hilfsfunktion können wir die folgende Formel verifizieren:

   $$\varphi(x(\alpha)) - \varphi(e) = \varphi'(e)(x(\alpha) - e) + \underbrace{\psi(\alpha) - \psi(0)}_{=\int_0^\alpha \psi'(\beta)\, d\beta}$$

   Wir werden im folgenden beide Summanden abschätzen und am Ende wieder diese Abschätzung zusammenfügen.

3. Zunächst betrachten wir den ersten Summanden:

   Wir erhalten für die Ableitung von φ:

   $$\frac{\partial \varphi}{\partial x_i}(x) = \frac{nc_i}{c^T x} - \frac{1}{x_i}$$

   Damit ergibt sich:

   $$\varphi'(e)(x(\alpha) - e) = \frac{nc^T(x(\alpha) - e)}{c^T e} - n(1, \ldots, 1)\underbrace{(x\alpha - e)}_{=-\alpha n d}$$

   wegen $\sum d_i = 0$ ist der hintere Term Null, also:

   $$= \frac{nc^T(x(\alpha) - e)}{c^T e}$$

   Mit dem Lemma gilt:

   $$c^T x(\alpha) = \min_{\substack{\|x-e\| \leq \alpha r \\ Ax = 0 \\ \sum_i x_i = 1}} c^T x \leq c^T x'$$

   wobei x' der Schnittpunkt der Strecke von e nach x^* und mit dem Kreis $\|x - e\| = \alpha r$.

   Per Konstruktion gilt dabei $\|x^* - e\| \leq 1$. Wir vergleichen nun, da $x' - e$ und $x^* - e$ parallel sind und damit aus der Linearität:

   $$\frac{c^T(x' - e)}{c^T(x^* - e)} \geq \frac{\alpha r}{R} \geq \alpha r$$

wobei $R \leq 1$ der Umkreisradius ist und damit gilt:

$$\frac{c^T(x' - e)}{c^T e} \leq -\alpha r$$

Und damit erhalten wir schließlich:

$$\varphi'(e)(x(\alpha) - e) = \frac{nc^T(x(\alpha) - e)}{c^T e} \leq \frac{nc^T(x' - e)}{c^T e} \leq -\alpha rn$$

4. Wir werden nun den zweiten Summanden abschätzen. Hierzu untersuchen wir zunächst die oben definierte Hilfsfunktion ψ, für die gilt:

$$\psi'(\alpha) = \varphi'(x(\alpha))x'(\alpha) - \frac{n}{c^T e} c^T x'(\alpha)$$

Wegen $x(\alpha) = e - \alpha rd$ ergibt sich $x'(\alpha) = -rd$ und damit erhalten wir mit der Definition von φ:

$$= nr \underbrace{\left(\frac{1}{c^T e} - \frac{1}{c^T x(\alpha)}\right)}_{\leq 0} \underbrace{c^T d}_{\geq 0} + r \sum_{i=1}^{n} \frac{d_i}{x_i(\alpha)}$$

Da die Konstenfunktion abnimmt, also $c^T x(\alpha) \leq c^T e$ ist, ist der erste Faktor im ersten Summanden tatsächlich negativ. Wegen $d = \frac{Pc}{\|Pc\|}$ ist $c^T Pc = c^T P^2 c = c^T P^T Pc = (Pc)^T(Pc) \geq 0$, also $c^T d \geq 0$.

Weiter gilt:

$$\begin{aligned} r \sum_{i=1}^{n} \frac{d_i}{x_i(\alpha)} &= r \sum_{i=1}^{n} \frac{d_i}{\frac{1}{n} - \alpha r d_i} \\ &= nr \sum_{i=1}^{n} \frac{d_i}{1 - \alpha rnd_i} \end{aligned}$$

Da $\sum_i d_i = 0$ ist, erhalten wir damit:

$$\begin{aligned} &= nr \sum_{i=1}^{n} d_i \left(\frac{1}{1 - \alpha rnd_i} - 1\right) \\ &= nr \sum_{i=1}^{n} d_i \frac{\alpha rnd_i}{1 - \alpha nrd_i} \\ &= \alpha (nr)^2 \sum_{i=1}^{n} \frac{d_i^2}{1 - \alpha rnd_i} \end{aligned}$$

Da $\sum_{i=1}^{n} d_i^2 = 1$ ist, gilt nun:

$$\leq \frac{\alpha (rn)^2}{1 - \alpha rn}$$

Mit dem obigen gilt damit:

$$\begin{aligned}\psi(\alpha) - \psi(0) &\leq \int_0^\alpha \frac{\beta(rn)^2}{1-\beta rn}\, d\beta \\ &= -\alpha rn - \log(1-\alpha rn)\end{aligned}$$

5. Wir setzen nun wieder beide Abschätzungen zusammen und erhalten insgesamt:

   $$\varphi(x(\alpha)) - \varphi(e) \leq -2\alpha rn - \log(1-\alpha rn)$$

   Die rechte Seite wird dabei für $\alpha rn = \frac{1}{2}$ minimal. Wegen $rn = \sqrt{\frac{n}{n-1}}$, müssen wir nun

   $$\alpha := \frac{1}{n}\sqrt{\frac{n-1}{n}}$$

   setzen und für dieses α gilt dann:

   $$\varphi(x(\alpha)) - \varphi(e) \leq -1 - \log\frac{1}{2} = -1 + \log 2 < -0.3$$

Damit gilt der Satz. □

Beweis (der Konvergenzaussage)
Wir berechnen zunächst:

$$\begin{aligned}\varphi(x^k) &= n\log(c^T x^k) - \sum_{i=1}^n \log(x_i^k) \\ \varphi(x^0) - k\delta &= n\log(c^T x^0) - \sum_{i=1}^n \log(x_i^0) - k\delta\end{aligned}$$

wobei $x_i^0 = \frac{1}{n}$ ist aufgrund der Anfangswahl.

Mit dem Hilfssatz erhalten wir $\varphi(x^k) \leq \varphi(x^0) - k\delta$, also:

$$\begin{aligned}n\log\left(\frac{c^T x^k}{c^T x^0}\right) &\leq \underbrace{\sum_{i=1}^n \log(x_i^k)}_{\leq 0} \underbrace{- \sum_{i=1}^n \log\frac{1}{n}}_{= n\log n} - k\delta \\ &\leq n\log n - k\delta\end{aligned}$$

Damit gilt:

$$\frac{c^T x^k}{c^T x^0} \leq e^{\log a - \frac{k}{n}\delta} \leq 2^{-q} = e^{q\log 2}$$

Dies gilt, wenn

$$k \geq \frac{n}{\delta}\left(q\log 2 + \log n\right)$$

und wegen $\delta > 0,3$ und $\log 2 \approx 0,69$ erhalten wir damit die Aussage des Hauptsatzes. □

5.63 Bemerkung (Praktische Wahl von alpha)
In der Praxis wählen wir unseren Parameter α nicht wie im Satz angegeben, sondern dort wird $\varphi(x(\alpha))$ unter den Nebenbedingungen $\alpha > 0$ und $e - \alpha rd > 0$ minimiert. Es reicht bereits eine Näherung des Minimums.

Typischerweise ist α deutlich näher an 1 als das im Satz angegebene α.

5.64 Bemerkung (Abbruchkriterium)

1. Falls $\phi(x^k) \leq \varphi(x^0) - q$ ist für ein vorgegebenes q, dann brechen wir den Algorithmus ab und suchen die benachbarte Ecke. Mit einem Simplex-Schritt überprüfen wir diese Ecke auf Optimalität. Sonst machen wir evtl. mit dem Simplex-Algorithmus weiter, der dann (für gute q) nicht mehr weit braucht.
2. Falls $\varphi(x^{k+1}) > \varphi(x^k) - \delta$ ist, dann gibt es keine Lösung. (Es kann sogar gezeigt werden, daß falls keine Lösung existiert, es ein solches k geben kann).

Stichwortverzeichnis