Karl-Franzens-Universität Graz
Institut für Mathematik und Wissenschaftliches Rechne
Heinrichstr. 36, A-8010 Graz, Österreich

Numerische Mathematik

für LAK

Das Skriptum Team
Patrick Ditz (Projekt Leiter), Nicole Muhr, Heidi Gräser, Johannes Kressl, Eva Haubenwaller,
Elke Gruber, Veronika Hammerl, Patricia Foditsch, Mario Scheiber,
Josef Bergthaler, Sabine Schöffmann.

Mitschrift aus dem SoSe 2005
zur Vorlesung von Ao.Univ.-Prof. Dr. Alfio Borzì

Vorwort

Die Idee eines Projektes für die Erstellung eines Skriptums für die Vorlesung Numerische Mathematik ist im Verlauf der Vorlesung entstanden mit dem bescheidenen Ziel einfach die handgeschriebenen Notizen von mir in ein elektronisches Dokument umzuwandeln. Dies ermöglicht den StudentenInnen eine einheitliche Wissensgrundlage zu haben.
Das Ergebnis ist ein Skriptum, das viel besser ist als meine Unterlagen. Dafür bin ich und alle Studenten, die diese Skriptum verwenden werden, dankbar.
Das Projekt wurde von Patrick Ditz geleitet. Er hat dafür gesorgt dass ein einheitlicher Latex Stil verwendet wurde und es zu einem vollständigen Dokument wurde. Viel wichtiger war aber, dass Er und die andere StudentenInnen inhaltliche Überarbeitungen vorgenommen haben.

Das Skriptum Team:
Patrick Ditz (Projekt Leiter)
Nicole Muhr, Heidi Gräser, Johannes Kressl, Eva Haubenwaller, Elke Gruber, Eva Wagner, Veronika Hammerl, Patricia Foditsch, Mario Scheiber, Josef Bergthaler, Sabine Schöffmann.

Alfio Borzì

Graz, 4. Juni 2007

Inhaltsverzeichnis

Kapitel 1

Mathematische Grundlagen

1.1 Vektornorm

Definition 1.0. Sei V ein Vektorraum über $\mathbb{K}$ und $\|\cdot\| : V \to \mathbb{R}^+$. Das Paar $(V, \|\cdot\|)$ heißt normierter Raum und $\|\cdot\|$ **Norm**, wenn die Abbildung $\|\cdot\|$ folgende Eigenschaften besitzt:

(N1): $\forall x \in V : \|x\| \geq 0$

$\forall x \in V : \|x\| = 0 \Leftrightarrow x = 0$ Definitheit

(N2): $\forall x \in V, \forall \lambda \in \mathbb{K} : \|\lambda x\| = |\lambda| \, \|x\|$ Homogenität

(N3): $\forall x, y \in V : \|x + y\| \leq \|x\| + \|y\|$ Dreiecksungleichung

Die Norm eines Vektors x kann als Abstand von x zum Nullvektor interpretiert werden.

Beispiel 1.1. *1. p-Norm: sei $V = \mathbb{R}^n$ für $x \in \mathbb{R}^n, x = (x_1, x_2, \ldots, x_n)^T$ definieren wir*

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

Wir wollen die so erhaltenen Normen für p=1, p=2, p=∞ näher betrachten und erhalten:

2.

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

3.

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} = \sqrt{x^T x} \qquad \textit{euklidische Norm}$$

4.

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \qquad \textit{Maximumsnorm}$$

SATZ 1.2 (Hölder Ungleichung)**.** *Für x,y $\in \mathbb{R}^n$ und $1 \leq p, q \leq \infty$ gilt:*

$$\left| x^T y \right| \leq \|x\|_p \, \|y\|_q \quad \textit{wobei } \frac{1}{p} + \frac{1}{q} = 1$$

Der Spezialfall dieser Ungleichung ist die von *Cauchy-Schwarz*.

SATZ 1.3 (Cauchy-Schwarz'sche Ungleichung)**.** *Für $x, y \in \mathbb{R}^n$ gilt:*

$$\left| x^T y \right| \leq \|x\|_2 \, \|y\|_2$$

Definition 1.4. Zwei Normen $\|\cdot\|$ und $\|\cdot\|'$ auf einem Vektorraum heißen äquivalent, wenn es Konstanten $c_1, c_2 > 0$ gibt, so dass $\forall x \in V$ die Abschätzungen

$$c_1 \|x\|' \leq \|x\| \leq c_2 \|x\|'$$

gelten.

SATZ 1.5. *Alle Normen in $\mathbb{K}^n$ sind äquivalent.*

Beispiel 1.6. *Sei $x \in \mathbb{K}^n$*

1. $$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\,\|x\|_2$$
2. $$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\,\|x\|_\infty$$
3. $$\|x\|_\infty \leq \|x\|_1 \leq n\,\|x\|_\infty$$

1.2 Matrixnorm

Definition 1.7. Eine *Matrixnorm* auf $\mathbb{K}^{m\times n}$ ist eine Abbildung $\|\cdot\| : \mathbb{K}^{m\times n} \to \mathbb{R}^+$ mit den drei Eigenschaften einer Vektornorm und, sie muss zusätzlich submultiplikativ sein:

(N1): $\forall A \in \mathbb{K}^{m\times n} : \|A\| \geq 0$
$\forall A \in \mathbb{K}^{m\times n} : \|A\| = 0 \Leftrightarrow A = 0$ Definitheit
(N2): $\forall A \in \mathbb{K}^{m\times n}, \forall \lambda \in \mathbb{K} : \|\lambda A\| = |\lambda|\,\|A\|$ Homogenität
(N3): $\forall A, B \in \mathbb{K}^{m\times n} : \|A + B\| \leq \|A\| + \|B\|$ Dreiecksungleichung
(N4): $\forall A \in \mathbb{K}^{m\times n}, \forall B \in \mathbb{K}^{n\times r} : \|AB\| \leq \|A\|\,\|B\|$ Submultiplikativität

Definition 1.8. Die Größe

$$\rho(A) := \max_{1\leq i\leq n} \left\{|\lambda_i| : \lambda_i \text{ Eigenwert von } A \in \mathbb{K}^{n\times n}\right\}$$

heißt **Spektralradius**.

SATZ 1.9. *Für jede natürliche Matrix-Norm $\|\cdot\|$ gilt*

$$\rho(A) \leq \|A\|\,.$$

Beweis 1.10. Es bezeichne λ den betragsgrößten Eigenwert von A und x den zugehörigen Eigenvektor, $Ax = \lambda x$, sowie die $\|\cdot\|$ Vektor-Norm, die die Matrix-Norm induziert. Sei nun ohne Einschränkung $\|x\| = 1$, dann gilt

$$\rho(A) = |\lambda| = |\lambda|\,\|x\| = \|\lambda x\| = \|Ax\| \leq \|A\|\,\|x\| = \|A\|$$

q.e.d.

Beispiel 1.11. *Sei $A \in \mathbb{R}^{m\times n}$*

1. *Eine wichtige Matrixnorm ist die **Frobenius-Norm**:*
$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$
2. *induzierte **p-Norm**: (Grenznorm)*
$$\|A\|_p = \sup_{\|x\|_p \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$
Eine solche Norm kann man als größte Abbildungsdehnung von A interpretieren. Sie gibt an um wieviel die Norm des Bildes ($\|Ax\|_p$), größer ist als die Norm des Urbildes ($\|x\|_p$).

3. *Die* ***Spaltensummennorm*** *ist*

$$\|A\|_1 = \max_j \sum_{i=1}^{m} |a_{ij}|$$

4. *Die* ***Zeilensummennorm*** *ist*

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$$

5. *Die* ***Spektralnorm*** *ist*

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

Wir werden später sehen, dass jede symmetrische positive definite Matrix A ein Skalarprodukt induziert: $(x, y) =< x, Ay >$ und damit die Norm $\|x\|_A = \sqrt{(x, x)} = \sqrt{< x, Ax >}$.

Definition 1.12 (Verträglichkeit)**.** Sei $A : \mathbb{R}^n \to \mathbb{R}^m$ und $A \in \mathbb{R}^{m\times n}$ dann sind $\|\cdot\|_\alpha$ auf $\mathbb{R}^n$ und $\|\cdot\|_\beta$ auf $\mathbb{R}^m$ *verträglich*, wenn gilt:

$$\|Ax\|_\beta \leq \|A\|_{\alpha,\beta} \|x\|_\alpha$$

wobei

$$\|A\|_{\alpha,\beta} = \sup_{\|x\|_\alpha \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}$$

Sei $A \in \mathbb{R}^{m\times n}$ dann gilt, dass alle Normen auf $\mathbb{R}^{m\times n}$ äquivalent sind. (Gleich wie bei Vektornorm)

Beispiel 1.13. *sei* $A \in \mathbb{R}^{m\times n}$

1. $$\|A\|_2 \leq \|A\|_F \leq \sqrt{n} \|A\|_2$$

2. $$\frac{1}{\sqrt{n}} \|A\|_\infty \leq \|A\|_2 \leq \sqrt{m} \|A\|_\infty$$

3. $$\frac{1}{\sqrt{m}} \|A\|_1 \leq \|A\|_2 \leq \sqrt{n} \|A\|_1$$

SATZ 1.14. *Sei* $A \in \mathbb{R}^{m\times n}$*, dann gilt*

$$\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$$

Lemma 1.15. Sei $A \in \mathbb{R}^{n\times n}$ mit $\|A\|_p$<1. Dann ist $(I - A)$ invertierbar und es gilt

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

mit

$$\left\|(I - A)^{-1}\right\|_p \leq \frac{1}{1 - \|A\|_p}$$

SATZ 1.16. *Sei A invertierbar und* $\left\|A^{-1}B\right\|_p$<1*. Dann ist* $(A + B)$ *invertierbar und es gilt*

$$\left\|(A + B)^{-1} - A^{-1}\right\|_p \leq \frac{\|B\|_p \left\|A^{-1}\right\|_p}{\|A^{-1}B\|_p}$$

1.3 Kondition

Definition 1.17. Sei $A \in \mathbb{R}^{n\times n}$ regulär (d.h.: invertierbar) dann heißt

$$K(A) = \|A\| \left\|A^{-1}\right\|$$

Konditionszahl von A bezüglich $\|\cdot\|$.

Die Konditionszahl einer Matrix ist ein Maß für die Stabilität einer Matrix. Je größer die Konditionszahl, umso näher liegt die Matrix bei einer singulären Matrix. Dies kann beispielsweise negative Auswirkungen beim Lösen von linearen Gleichungssystemen haben.

SATZ 1.18. *Sei $A, \delta A \in \mathbb{R}^{n\times n}$ und $b(\neq 0), \delta b \in \mathbb{R}^n$ mit*

$$Ax = b \quad und \quad (A + \delta A)y = b + \delta b$$

wobei

$$\|\delta A\| \leq \epsilon \|A\| \qquad \|\delta b\| \leq \epsilon \|b\|$$

Falls $r = \epsilon K(A) < 1$, dann ist $(A + \delta A)$ nicht singulär und es gilt:

$$\frac{\|y\|}{\|x\|} \leq \frac{1+r}{1-r}$$

SATZ 1.19. *Die Vorraussetzungen sind wie oben. Daraus ergibt sich weiters:*

$$\frac{\|y-x\|}{\|x\|} \leq \frac{2\epsilon}{1-r} K(A)$$

1.4 Spezielle Matrizen

Wichtig für dieses Kapitel ist die quadratische Form einer symmetrischen Matrix A:

$$Q(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} a_{ij} x_j \right) x_i$$

Definition 1.20. Die notwendige und hinreichende Bedingung dafür, dass eine Matrix *positiv definit* ist lautet:

$$Q(x) = x^T Ax > 0$$

für beliebige Vektoren $x \neq 0$.

SATZ 1.21. *Eine positive definite Matrix A hat positive Diagonalelemente.*

Beweis 1.22. Wir nehmen an, dass es ein negatives Hauptdiagonalelement $a_{ii} \leq 0$ gibt. Dann kann man $x = e_i$ einen Vektor konstruieren, so dass $e_i^T A e_i \leq 0$ gilt, d.h. eine derartige Matrix kann nicht positiv definit sein. q.e.d.

SATZ 1.23. *Für eine positiv definite Matrix A gilt $\forall i \neq j$:*

$$a_{ij}^2 < a_{ii} a_{jj}$$

SATZ 1.24. *Aus dem vorherigen Satz folgt, dass das absolut größtes Element einer positiv definiten Matrix notwendig auf der Hauptdiagonale liegt.*

Definition 1.25. Eine Matrix heißt *streng diagonal dominant*, falls:

$$a_{ii} > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad \text{für } i = 1, 2, \ldots$$

Für die positive Definitheit von A ist diese Bedingung hinreichend aber nicht notwendig.

SATZ 1.26. *Eine symmetrische Matrix A mit positiven Diagonalelementen ist positiv definit, falls gilt:*

$$a_{ii} a_{jj} > (\sum_{\substack{k=1 \\ k \neq i}}^{n} |a_{ik}|)(\sum_{\substack{k=1 \\ k \neq j}}^{n} |a_{jk}|) \qquad \textit{für } i, j = 1, \ldots, n; i \neq j$$

SATZ 1.27. *Eine Matrix A ist dann und nur dann positiv definit, falls die Matrix auf folgende Form gebracht werden kann*

$$A = R^T R.$$

*wobei R eine reguläre obere Dreiecksmatrix ist. Die Indefinitheit zeigt sich bei diesem Rechenverfahren zur Ermittlung von R (siehe **Cholesky -Verfahren**) durch das Auftreten eines negativen Diagonalelementes.*

SATZ 1.28. *Eine symmetrische Matrix A ist positiv definit (spd-Matrix), genau dann wenn alle Eigenwerte positiv sind (d.h. $\lambda_i > 0 \quad \forall i = 1, \ldots, n$).*

1.5 Landau-Symbole

Die Landau-Notation wird verwendet, um das asymptotische Verhalten bei Annäherung an einen endlichen oder unendlichen Grenzwert zu beschreiben. Das große O wird verwendet, um eine maximale Größenordnung anzugeben.
Es erlaubt Terme gleicher Größenordnungen bzw. Funktionen gleicher Größenordnungen zusammenzufassen.

Definition 1.29. Seien $(f_n), (g_n) \subset \mathbb{K}$ Folgen, dann ist für $n \to \infty$:

$$\begin{aligned} f_n = O(g_n) &\Leftrightarrow \exists C > 0 \exists n_0 \forall n \geq n_0 : |f_n| \leq C \cdot |g_n| \\ f_n = o(g_n) &\Leftrightarrow \forall \epsilon > 0 \exists n_0 \forall n \geq n_0 : |f_n| \leq \epsilon \cdot |g_n| \end{aligned}$$

Beispiel 1.30.

$$\begin{array}{ccccc} 3n^2 + 5n + 2 & = & 3n^2 + O(n) & = & O(n^2) \\ \frac{4}{n} + \frac{3}{n \ln(n)} & = & \frac{4}{n} + o\left(\frac{1}{n}\right) & = & O(1) \end{array}$$

Definition 1.31. Seien $f, g : \mathbb{K} \to \mathbb{K}$ und $x_0 \in \mathbb{K}$ dann ist für $x \to x_0$

$$\begin{aligned} f(x) = O(g(x)) &\Leftrightarrow \exists C > 0 \exists U(x_0) \forall x \in U(x_0) : |f(x)| \leq C \cdot |g(x)| \\ f(x) = o(g(x)) &\Leftrightarrow \forall \epsilon > 0 \exists U(x_0) \forall x \in U(x_0) : |f(x)| \leq \epsilon \cdot |g(x)| \end{aligned}$$

Beispiel 1.32. *In der Taylor-Entwicklung ergibt sich*

$$\sin(x) = x - \frac{x^3}{3!} + O(x^5) \quad (x \to 0)$$

oder für $f \in C^2(U(x))$

$$f(x + h) = f(x) + f'(h)h + O(h^2) \quad (h \to 0)$$

1.6 Konvergenzordnung

Definition 1.33. Sei $\|\cdot\|$ eine Norm auf dem $\mathbb{K}$-Vektorraum V, $x^* \in V$ und $(x_n) \subset V$ mit $x_n \to x^*$ eine gegen x^* konvergente Folge in V.

1. (x_n) konvergiert (mindestens) **linear**, wenn es ein $q < 1$ und ein n_0 gibt, so dass

$$\|x_{n+1} - x^*\| \leq q \cdot \|x_n - x^*\| \quad \forall n \geq n_0.$$

2. (x_n) konvergiert (mindestens) **superlinear**, wenn $\exists n_0, (\epsilon_n) \subset \mathbb{R}_+$ mit $\epsilon_n \to 0$, so dass

$$\|x_{n+1} - x^*\| \leq \epsilon_n \cdot \|x_n - x^*\| \quad \forall n \geq n_0.$$

3. (x_n) besitzt (mindestens) **Konvergenzordnung** α, wenn $\exists c > 0$ und n_0 mit

$$\|x_{n+1} - x^*\| \leq c \cdot \|x_n - x^*\|^\alpha \quad \forall n \geq n_0.$$

   Im Fall $\alpha = 2$ spricht mann von **quadratischer Konvergenz** bzw. von **kubischer Konvergenz** im Fall $\alpha = 3$.

4. Ein Iterationsverfahren

$$x_{n+1} := \phi(x_n) \quad n \geq 0 \text{ mit } x_0 \in V, \phi : V \to V$$

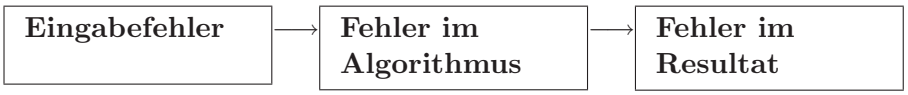
   besitzt bezüglich des Fixpunktes von ϕ, also $x^* = \phi(x^*)$, lokal (mindestens) die Konvergenzordnung $\alpha \geq 1$, wenn $\exists U(x^*)$, so dass jede Iterationsfolge mit $x_0 \in U(x^*)$ gegen x^* konvergiert und (x_n) die Konvergenzordnung (mindestens) α besitzt.

Kapitel 2

Fehleranalyse

Sei ein Problem abstrakt charakterisiert durch (f, x) mit gegebener Abbildung f und gegebenen Eingabedaten x. Das Problem lösen heißt dann, das Resultat $f(x)$ mit Hilfe eines Algorithmus bestimmen, der evtl. noch Zwischengrößen produziert:

In der praktischen Umsetzung diverser Algorithmen treten jedoch unweigerlich Fehler auf, wodurch sich folgendes Schema ergibt:



Gegenüber Eingabefehlern sind wir im Prinzip machtlos, sie gehören zum gegebenen Problem und können allenfalls durch eine Änderung der Problemstellung vermieden werden. Hingegen können Fehler im Algorithmus teilweise vermieden oder zumindest verringert werden, indem das Verfahren verändert wird. Allgemein führen diese zwei Arten von Fehlern zu den Begriffen *Kondition eines Problems* und *Stabilität eines Algorithmus*, auf welche in Folgenden genauer eingegangen wird. Zunächst wollen wir aber auf eine in beiden Bereichen vorherrschende Fehlerquelle eingehen: *eine endliche Maschine alias Computer, Taschenrechner ...*

2.1 Maschinenzahlen

Auch wenn die Eingabegrößen als exakt gegeben angesehen werden, treten durch die Darstellung nicht ganzzahliger Zahlen in einem Rechner Eingabefehler auf.
Beim numerischen Rechnen wird eine reelle Zahl $z \in \mathbb{R}$ ersetzt durch eine der endlich vielen Maschinenzahlen in Gleitkommadarstellung. Eine solche Zahl hat die Form

$$z = a \cdot d^e$$

Die Unbekannten haben folgende Bedeutung:

- a - **Mantisse**. Sie ist entweder 0 oder es gilt $d^{-1} \leq |a| < 1$ (d.h. $a_1 \neq 0$ zur Normalisierung, da ansonsten die Darstellung nicht eindeutig wäre) in der Form

  $$a = v \sum_{i=1}^{l} a_i d^{-i} = v \left(\frac{a_1}{d^1} + \ldots + \frac{a_l}{d^l} \right)$$

  mit $v \in \{\pm 1\}$ (Vorzeichen), $a_i \in \{0, \ldots, d-1\}$ und $l \in \mathbb{N}$ ist die Mantissenlänge, welche die relative Genauigkeit bestimmt.

- d - **Basis**. $d \in \mathbb{N} \backslash \{1\}$, die üblicherweise durch eine Zweierpotenz $\{2, 8, 16, 32, \ldots\}$ repräsentiert wird.

- e - **Exponent**. $e \in \{e_{min}, \ldots, e_{max}\} \in \mathbb{Z}$(=Exponentenbereich). Dieser bestimmt die betragsmäßig größte und kleinste Zahl.

2.1.1 Gleitkommazahlen nach dem IEEE Standard

Die Norm IEEE definiert Standarddarstellungen für Gleitkommazahlen in Computern und legt genaue Verfahren für die Durchführung mathematischer Operationen fest.
Es werden zwei Standarddatenformate mit 32 Bit (single precision = einfache Genauigkeit) bzw. 64 Bit (double precision = doppelte Genauigkeit) Speicherbedarf definiert. Hierzu wollen wir exemplarisch auf eines dieser Formate näher eingehen:

Single precision floating point (32 Bits)
Die allgemeine Darstellung einer Fließkommazahl besteht aus:

- einem Vorzeichenbit (1: negativ, 0: positiv),
- Charakteristikbits,
- Mantissenbits.

Die Anordnung der Bits zeigt die nachfolgende Abbildung:

0	1 8	9 31
Vorzeichen(1 Bit)	Exponent(8 Bits)	Mantisse(23 Bits)

Die Basis d (nach dem IEEE Standard for floating Point operations) ist bekanntlicherweise 2. Für $0 < e < 255$ kann man die, der Darstellung entsprechende, reelle Zahl durch

$$z = (-1)^v \cdot 1.a \cdot 2^{e-127}$$

ermitteln; wobei $v \in \{0, 1\}$, $e_{min} = -126$ $e_{max} = 127$ und $a = \sum_{i=1}^{23} a_i 2^{-i}$ mit $a_i \in \{0, 1\}$

Bemerkung 2.0. Es gibt eine kleinste und größte positive darstellbare Zahl, da es nur endlich viele Ziffern gibt. Man bezeichnet die kleinste positive Zahl als $MinReal := d^{e_{min}}$ und die größte positive Zahl als $MaxReal := \left(\sum_{i=0}^{l-1} \frac{d-1}{d^i}\right) d^{e_{max}} = \left(d - d^{1-l}\right) d^{e_{max}}$. In unserem Fall erhalten wir $MinReal \approx 1.175 \cdot 10^{-38}$ und $MaxReal \approx 3.403 \cdot 10^{38}$.

Ausnahmen:

e=255 a $\neq$ 0	$\Rightarrow$ z=NaN (Not-A-Number)	1|11111111|10011001000100111001010
e=255 a=0 v=0	$\Rightarrow$ z=Inf ($=\infty$)	0|11111111|00000000000000000000000
e=255 a=0 v=1	$\Rightarrow$ z=-Inf ($=-\infty$)	1|11111111|00000000000000000000000
e=0 a=0 v=1	$\Rightarrow$ z=-0	1|00000000|00000000000000000000000
e=0 a=0 v=0	$\Rightarrow$ z=0	0|00000000|00000000000000000000000
e=0 a $\neq$0	$\Rightarrow z = (-1)^v \cdot 0.a \cdot 2^{-126}$	0|00000000|00000000010000100001100

Nun wollen wir noch aus der Bitfolge | 0 | 10000011 | 10100110000000000000000 |
die entsprechende reelle Zahl z ermitteln:
aus den Exponential-Bits $\Rightarrow e = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 131$
aus den Mantissen-Bits $\Rightarrow a = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + 1 \cdot 2^{-6} + 1 \cdot 2^{-7} = 0.6484375$
Damit erhält man

$$z = (-1)^0 \cdot 1.6484375 \cdot 2^{131-127} = 26.375$$

Zusammenfassend könnte man noch folgende allgemeine Bemerkungen anbringen:

- Der Raster der normalisierten Gleitkomazahlen ist höchst ungleichmässig über $\mathbb{R}$ verteilt und enthält einige große Lücken bei 0(nicht normalisierte Maschinenzahl).
- Die Mantissenlänge bestimmt die Lückengrösse und damit die Genauigkeit.
- e_{min} und e_{max} liefern die Grenzen *MaxReal* und *MinReal* des Darstellungsbereichs.
- Nur die endlich vielen Zahlen des normalisierten Gleitpunkt-Systems sind exakt darstellbar, die anderen werden gerundet.

2.2 Gleitpunkt-Arithmetik und Rundungsfehler

Definition 2.1 (Maschinengenauigkeit)**.** Die **Maschinengenauigkeit eps** ist die kleinste positive Maschinenzahl welche zu 1 addiert ein Resultat $\neq 1$ ergibt. Eine neuere Definition lautet: Der Abstand zwischen zwei aufeinanderfolgenden Floating Point Numbers zwischen 1 und 2. Für single precision (32 bits) ist $eps = 2^{-23} \approx 1.192 \cdot 10^{-7}$.

SATZ 2.2. *Jede reelle Zahl x mit $MinReal \leq |x| \leq MaxReal$ läßt sich (durch Rundung auf die nächstliegende Maschinenzahl) durch eine Gleitkommazahl (floating point number) $fl(x)$ repräsentieren. Dabei gilt*

$$fl(x) := x(1+\delta) \qquad |\delta| \leq eps$$

Wir erhalten einen ***relativen Fehler*** *von*

$$\frac{|x - fl(x)|}{|x|} \leq \frac{d^{1-l}}{2} =: eps \quad x \neq 0$$

Beispiel 2.3. *Betrachten wir die Zahl $\pi = 0.314159265\ldots \cdot 10^1$ dann erhalten wir $fl(\pi) = 0.31416 \cdot 10^1 (+0.00001; 5 digits)$ mit $l = 5$(Mantissenlänge), $d = 10$(Basis) und $e = 1$(Exponent). D.h. wir machen einen absoluten Fehler $|\pi - fl(\pi)| = 0.000000735\ldots$ und der relativen Fehler $\frac{|\pi - fl(\pi)|}{|\pi|} \leq \frac{10^{1-5}}{2} = 0.00005$*

Im Regelfall werden beim numerischen Lösen die Verkettung und Ausführung einer Vielzahl von Grundoperationen benötigt. Unter einem Algorithmus (Rechenverfahren) versteht man in der Numerik eine endliche Folge von Grundoperationen, deren Reihenfolge beim Ablauf eindeutig festliegt. In Gleitpunktarithmetik ausgeführt, verfälschen Rundungsfehler die Zwischen- und Endresultate.

Definition 2.4. Der IEEE-Standard fordert, dass bei der Realisierung einer Elementaroperation $\circ \in \{+, -, \times, /\}$ durch die entsprechende Gleitkommaoperation $\hat{\circ}$ folgendes gilt

$$x \hat{\circ} y = fl(x \circ y) = (x \circ y)(1+\delta) \qquad |\delta| \leq eps$$

Bemerkung 2.5. Die Reihenfolge der Operationen kann bei der Rundungsfehleranalyse entscheidend sein, da eine **Gleitpunktarithmetik weder Assoziativ- noch Distributivgesetze erfüllt**, wie das folgende Beispiel zeigt.

Beispiel 2.6. *Wir gehen von einem normalisierten Gleitkomma-Sytem mit b=10 und l=8 aus(die Exponenten-Grenzen seien beliebig groß). Die Größen a, b, c mit*

$$\begin{aligned} a &= 0.23371258 \cdot 10^{-5} \\ b &= 0.33678429 \cdot 10^{1} \\ c &= -0.33677811 \cdot 10^{1} \end{aligned}$$

seien exakt, dann ist

$$\begin{aligned} b + c &= 0.00000618 \cdot 10^{1} \\ fl(b+c) &= 0.618 \cdot 10^{-4} \\ a + fl(b+c) &= 0.64137125|8 \cdot 10^{-4} \\ fl(a + fl(b+c)) &= 0.64137126 \cdot 10^{-4} \end{aligned}$$

aber bei einer anderen Klammerung ergibt sich

$$\begin{aligned} a + b &= 0.33678452|371258 \cdot 10^{1} \\ fl(a+b) &= 0.33678452 \cdot 10^{1} \\ fl(a+b) + c &= 0.00000614 \cdot 10^{1} \\ fl(fl(a+b)+c) &= 0.641 \cdot 10^{-4} \end{aligned}$$

Also im Allgemeinen gilt

$$fl(fl(a+b)+c) \neq fl(a+fl(b+c))$$

Frage: *Woher kommt der Verlust von ca. 5 Stellen?*
Antwort: *fl(a+b) und c sind nahezu betragsgleich und werden subtrahiert ⇒ es entstehen führende Nullen im Ergebnis. Durch Normalisierung rücken von hinten Nullen nach, die aber im Fall $fl(fl(a+b)+c)$ nicht exakt sind, weil fl(a+b) nicht exakt war, wogegen im Fall $fl(a+fl(b+c))$ sind b und c zwar auch nahezu betragsgleich, aber die durch die Normalisierung des Ergebnisses nachgerückten Nullen sind exakt, da die Ausgangsdaten als exakt vorausgesetzt waren.*
Folgerung: *Verlust an relativer Genauigkeit!*

Bemerkung 2.7. Das vorherige Beispiel liefert eine wichtige Regel für das numerische Rechnen mit fehlerbehafteten Zahlen: Die Subtraktion nahezu gleicher, fehlerbehafteter Zahlen(trifft i.A. auf alle Computer-Zahlen wegen Rundung und Konvertierung zu) sollte unbedingt vermieden werden. Dieses Phänomen heißt **Auslöschung führender Ziffern**.

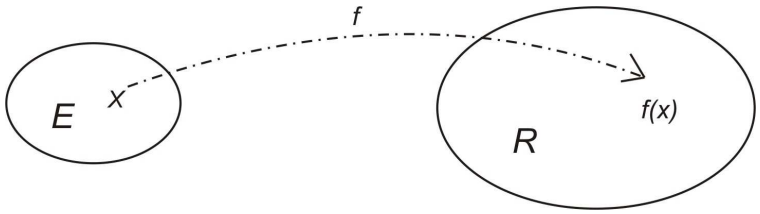
2.3 Kondition eines Problems

Problem: Wie stark ändert sich die exakte Lösung eines Problems(unabhängig vom Algorithmus) bei Störung der Eingangsdaten z.B. durch Rundung oder Konvertierung?

Der vorherige Teil hat gezeigt, dass die Eingabe x logisch ununterscheidbar ist von allen Eingaben $\tilde{x}$, die sich im Rahmen der vorgegebenen Genauigkeit befinden. Statt der „exakten“ Eingabe x sollten wir daher eine *Eingabemenge* E betrachten, die all diese gestörten Eingaben $\tilde{x}$ enthält. Eine Maschinenzahl x repräsentiert demnach die Eingabemenge

$$E = \{\tilde{x} \in \mathbb{R} : |\tilde{x} - x| \leq eps\,|x|\}$$

Die Abbildung f, die unser Problem beschreibt, überführt die Eingabemenge E in eine *Resultatmenge* $R = f(E)$. So wie wir von der vermeintlich exakten Eingabe x zur Eingabemenge E geführt wurden, so ist es bei der Analyse von Eingabefehlern daher sinnvoll, statt der *punktweisen* Abbildung $f : x \mapsto f(x)$ die *Mengenabbildung* $f : E \to R = f(E)$ zu untersuchen. Die Charakterisierung des Verhältnisses von E und R nennen wir die *Kondition* des durch (f,x) beschriebenen Problems.



Definition 2.8. Die **absolute normweise Kondition** des Problems (f, x) ist die kleinste Zahl K_{abs} so dass

$$\|f(\tilde{x}) - f(x)\| \leq K_{abs}\,\|\tilde{x} - x\| \qquad \tilde{x} \in U_x$$

und die **relative normweise Kondition** ist die kleinste Zahl K_{rel} so dass

$$\frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} \leq K_{rel}\frac{\|\tilde{x} - x\|}{\|x\|} \qquad \tilde{x} \in U_x$$

Bemerkung 2.9. Ist $K(= K_{rel})$ groß, spricht man von einem **schlecht konditionierten Problem**. Ist K nahe bei 1(und damit der relative Eingangsfehler in der Größenordnung des unvermeidbaren relativen Ergebnisfaktors), so heißt das Problem **gut konditioniert**.

Sei $f \in C^1(\mathbb{R})$ (d.h. f ist differenzierbar) so erhält man aufgrund des Mittelwertsatzes die Kondition über die Ableitung:

$$K_{abs} = \left\|f'(x)\right\| \qquad K_{rel} = \left\|f'(x)\right\| \frac{\|x\|}{\|f(x)\|}$$

wobei $\|f'(x)\|$ die Norm der Jacobi-Matrix $f'(x) \in M(m \times n; \mathbb{R})$ in der zugeordneten Matrixnorm

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

ist.

Beispiel 2.10. ***(1)*** *Für die Kondition der Addition (bzw. Subtraktion) definiert man*

$$f : \mathbb{R}^2 \longrightarrow \mathbb{R} \qquad (a,b)^T \longmapsto f(a,b) := a + b$$

$\Rightarrow f'(a,b) = (1,1)$ *; wählen wir nun auf* $\mathbb{R}^2$ *die 1-Norm* $\Rightarrow \|(a,b)\|_1 = |a| + |b|$ *und weiters ist* $\|f'(a,b)\|_1 = 2$*. Insgesamt erhält man*

$$K_{abs} = 2 \qquad K_{rel} = \frac{|a| + |b|}{|a+b|}$$

also für $a > 0$ *und* $b > 0$ *ist* $K_{rel} = 1$ *und damit das Problem gut konditioniert.*
Sind die Vorzeichen von a und b alternierend so folgt, speziell wenn $|a| \approx |b|$*, dass* $K_{rel} >> 1$ *und damit eine schlechte Konditionierung. Das Phänomen heißt* ***Auslöschung führender Ziffern****.*
(2) *Kondition eines linearen Gleichungssystems* $Ax = b$*. Sei* $b \in \mathbb{R}^n$ *die Eingabegröße dann erhält man*

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^n \qquad b \longmapsto f(b) := A^{-1}b$$

$\Rightarrow f'(b) = A^{-1}$*; und damit folgt*

$$K_{abs} = \left\|A^{-1}\right\| \qquad K_{rel} = \left\|A^{-1}\right\| \frac{\|b\|}{\|A^{-1}b\|} = \left\|A^{-1}\right\| \frac{\|Ax\|}{\|x\|} \leq \left\|A^{-1}\right\| \cdot \|A\|$$

Die allgemeine Definition der Kondition einer Matrix A ist gegeben durch

$$K(A) := \left\|A^{-1}\right\| \cdot \|A\|$$

Eine andere Darstellung für $K(A)$ *ist*

$$K(A) := \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|} \in [0, \infty].$$

Sie hat den Vorteil, dass sie auch für nicht invertierbare und rechteckige Matrizen wohldefiniert ist. Falls $A \in Gl(n)$ *symmetrisch und positiv definit ist, erhält man*

$$K(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

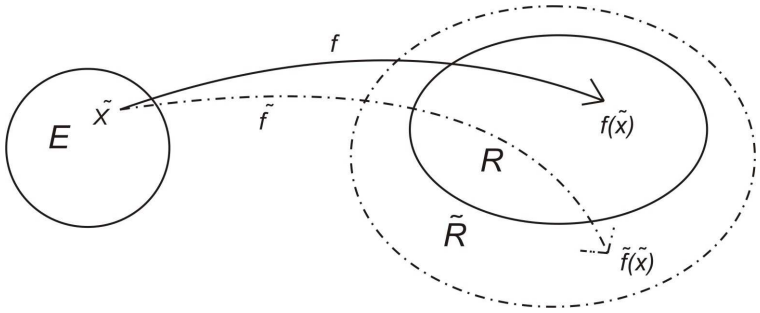
Bemerkung 2.11. Analoge Überlegungen kann man für eine **komponentenweise Konditionsanalyse** anstellen. Sie erweist sich häufig schon deshalb als günstiger, weil alle Eingaben in einen Rechner mit einem relativen Fehler in den einzelnen Komponenten behaftet sind und so einige Phänomene bei der normweisen Betrachtung nicht erklärt werden können.

2.4 Stabilität eines Algorithmus

Wir wenden uns in diesem Abschnitt der zweiten Gruppe von Fehlern zu, den Fehlern im Algorithmus. Die Abbildung f, die das gegebene Problem (f, x) beschreibt, wird bei der numerischen Lösung durch eine Abbildung $\tilde{f}$ realisiert. Diese Abbildung beinhaltet alle Rundungs- und Approximationsfehler und liefert anstelle von $f(x)$ ein gestörtes Resultat $\tilde{f}(x)$. Die Frage nach der Stabilität eines Algorithmus lautet nun:

Ist das Resultat $\tilde{f}(x)$ anstelle von $f(x)$ akzeptabel?

Wie geht man bei der Rundungsfehleranalyse vor? Man unterscheidet im wesentlichen zwei Ansätze, die **Vorwärtsanalyse** und die **Rückwärtsanalyse**. Naheliegend ist eine Vorwärtsanalyse, welche das in Gleitpunktarithmetik berechnete mit dem exakten Resultat vergleicht. Oft ist diese Technik jedoch schwierig, weswegen sich die Rückwärtsanalyse als Alternative durchgesetzt hat. Sie interpretiert das berechnete Resultat als exaktes Resultat zu geeignet veränderten Eingangsdaten. Trotzdem werden wir aus didaktischen Gründen nur auf die Vorwärtsanalyse als exemplarisches Stabilitätskonzept näher eingehen:



Die Vergrößerung von R nach $\tilde{R}$ kennzeichnet die Stabilität im Sinn der Vorwärtsanalyse. Ist $\tilde{R}$ von der gleichen Größenordnung wie R, so nennen wir den Algorithmus stabil im Sinn dieses Stabilitätkonzepts.

Definition 2.12. Sei $\tilde{f}$ die Gleitkommarealisierung eines Algorithmus zur Lösung des Problems (f, x) der relativen normweisen Kondition K_{rel}. Der Stabilitätsindikator(der normweisen Vorwärtsanalyse) ist die kleinste Zahl $\sigma \geq 0$, so dass für alle $\tilde{x} \in E$

$$\frac{\left\| \tilde{f}(\tilde{x}) - f(\tilde{x}) \right\|}{\|f(\tilde{x})\|} \leq \sigma \cdot K_{rel} \cdot eps$$

Wir nennen den Algorithmus $\tilde{f}$ *stabil* im Sinne der Vorwärtsanalyse, falls σ kleiner als die Anzahl der hintereinander ausgeführten Elementaroperationen ist.

Lemma 2.13. Für die Elementaroperationen $\{+, -, \times, /\}$ und ihre Gleitkommarealisierungen $\{\hat{+}, \hat{-}, \hat{\times}, \hat{/}\}$ gilt

$$\sigma K \leq 1$$

Beweis 2.14. Für jede elementare Gleitkommaoperation $\hat{\circ} \in \{\hat{+}, \hat{-}, \hat{\times}, \hat{/}\}$ gilt $a \hat{\circ} b = (a \circ b)(1 + \delta)$ für ein δ mit $|\delta| \leq eps$ und daher

$$\frac{|a \hat{\circ} b - a \circ b|}{|a \circ b|} = \frac{|(a \circ b)(1 + \delta) - a \circ b|}{|a \circ b|} = |\delta| \leq eps$$

q.e.d.

Kapitel 3

Lineare Gleichungssysteme

3.1 Grundlagen

Wir betrachten ein lineares Gleichungssystem mit n Gleichungen und n Unbekannnten

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

oder kurz

$$Ax = b \qquad \Leftrightarrow \qquad \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

wobei $A \in M(n \times n; \mathbb{K})$ und $b, x \in \mathbb{K}^n$ und der Körper $\mathbb{K} = \mathbb{R}$ oder $\mathbb{C}$

Definition 3.0. Die Anwendung einer elementaren Zeilenoperation auf die Einheitsmatrix ergibt eine zugehörige Elementarmatrix.

Bemerkung 3.1. Elementarmatrizen treten auf, wenn man an A Elementaroperationen durchführt (die die Lösung unverändert lassen).

Beispiel 3.2. *Sei $1 \leq r \leq s \leq n$, dann heißt*

$$P_{rs} = (e_1, e_2, \ldots, e_{r-1}, e_s, e_{r+1}, \ldots, e_{s-1}, e_r, e_{s+1}, \ldots e_n)$$

Permutationsmatrix, denn für ein $A \in M(n \times n; K)$ gilt: In dem Matrizenprodukt $P_{rs} \cdot A$ sind gegenüber A gerade die r-te und s-te Zeile vertauscht. Durch Nachrechen erkennt man weiters

$$P_{rs} = I - (e_r - e_s)(e_r - e_s)^T$$

und somit gilt

- *$det(P_{rs})$= -1 für $r \neq s$*
- *P_{rs} ist symmetrisch*
- *P_{rs} ist orthogonal*

Beispiel 3.3. *Sei $1 \leq k \leq n$. Eine Elementarmatrix der Form*

$$L_k = I - l_k e_k^T$$

mit $l_k = (0, \ldots, 0, l_{k+1}, \ldots, l_n)^T \in \mathbb{K}^n \setminus \{0\}$ *heißt Gauß-Matrix oder auch Frobenius-Matrix; d.h.* L_k *ist eine untere Dreiecksmatrix mit folgender Gestalt*

$$L_k = \begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & -l_{k+1,k} & 1 & \ddots & \vdots \\ \vdots & \vdots & 0 & 1 & 0 \\ 0 & -l_{n,k} & 0 & 0 & 1 \end{pmatrix}$$

Weitere Eigenschaften sind

$$det\,(L_k) = 1$$

sowie

$$\left(I - l_k e_k^T\right) \cdot \left(I + l_k e_k^T\right) = I - \underbrace{l_k e_k^T l_k e_k^T}_{=0} = I$$

d.h. L_k *ist regulär mit*

$$L_k^{-1} = I + l_k e_k^T$$

Bemerkung 3.4. Elementarmatrizen spielen bei direkten Verfahren zur Lösung von linearen Gleichungssystemen eine wichtige Rolle (im Gegensatz zu iterativen Verfahren). Dabei wird $Ax = b$ sukzessive mit P_{rs} und L_k multipliziert und nach endlich vielen Schritten in ein äquivalentes LGS umgewandelt, das durch Rücksubstitution gelöst wird.

SATZ 3.5. *Sei* $A \in M(n \times n, \mathbb{R})$ *mit* $\det(A) \neq 0$ *und* $b \in \mathbb{R}^n$*. Dann existiert genau ein* $x \in \mathbb{R}^n$*, so dass* $Ax = b$*.*

Im Prinzip läßt sich die Lösung $x = A^{-1}b$ mit der Cramerschen Regel berechnen. Der Rechenaufwand ist leider sehr groß. Selbst mit Hilfe des Laplace'schen Entwicklungssatzes

$$\det(A) = \sum_{i=1}^{n} (-1)^{i+1} a_{1i}\, det A_{1i}$$

sind 2^n Operationen auszuführen.
Ideal wäre ein Verfahren dessen Aufwand zur Berechnung eines Problems mit n Variablen ($\sim$Komplexität) linear proportional zu n, also $O(n)$ wäre. Wir werden direkte und iterative Verfahren untersuchen.

3.2 Direkte Verfahren

3.2.1 Auflösung gestaffelter Systeme

Hier betrachten wir den Fall eines *gestaffelten Gleichungssytems*

$$Rx = z \qquad \Leftrightarrow \qquad \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ 0 & r_{22} & \ldots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & r_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}$$

wobei R eine obere Dreiecksmatrix ist, d.h.$r_{ij} = 0$ für alle $i > j$. Beginnend mit der Zeile n, berechnet man x durch Rekursion:

$$\begin{array}{rcccl} x_n & := & z_n/r_{nn} & \text{falls} & r_{nn} \neq 0 \\ x_{n-1} & := & (z_{n-1} - r_{n-1n}x_n)/r_{n-1n-1} & \text{falls} & r_{n-1n-1} \neq 0 \\ \vdots & & \vdots & & \\ x_1 & := & (z_1 - r_{12}x_2 - r_{13}x_3 - \ldots - r_{1m}x_m)/r_{11} & \text{falls} & r_{11} \neq 0 \end{array}$$

Nun gilt für die obere Dreiecksmatrix R, dass $\det(R) = r_{11} \cdot \ldots \cdot r_{nn}$ und daher

$$\det(R) \neq 0 \Leftrightarrow r_{ii} \neq 0 \quad \forall i = 1, \ldots, n.$$

Für den Rechenaufwand ergibt sich:

- für die i-te Zeile: je $n - i$ Additionen und Multiplikationen, eine Division
- insgesamt für die Zeilen n bis 1:

$$\sum_{j=1}^{n}(n-j) = n^2 - \frac{n(n+1)}{2} = \frac{n(n-1)}{2} = O\left(\frac{n^2}{2}\right)$$

  Multiplikationen und ebensoviele Additionen und n Divisionen.

Bemerkung 3.6. Analog läßt sich ein gestaffeltes Gleichungs-System der Form

$$Lx = z$$

mit einer unteren Dreiecksmatrix L lösen, indem man jetzt in der ersten Zeile beginnt und sich bis zur letzten Zeile durcharbeitet. Diese Auflösung gestaffelter Syteme heißt *Vorwärtssubstitution*, und das zuvor vorgestellte analoge Verfahren mit der oberen Dreiecksmatrix wird *Rückwärtssubstitution* bezeichnet.

Algorithmus(Vorwärts- und Rückwärtssubstitution)

```
% Rückwärtssubstitution(Rx = z):
for j=n:1
   x_j = (z_j − Σ_{k=j+1}^{n} r_jk x_k)/r_jj
end
% Vorwärtssubstitution(Lx = z):
for i=1:n
   x_i = (z_i − Σ_{k=1}^{i−1} l_ik x_k)/l_ii
end
```

3.2.2 Gaußsche Eliminationsmethode

Idee: Transformiere $Ax = b$ mit $det\,(A) \neq 0$ auf ein oberes Dreieckssytem $Rx = z$, ohne die Lösung zu verändern (Löse dann $Rx = z$ durch Rückwärts-Substitution).

Vorgehensweise: Das lineare Gleichungssytem soll also in ein gestaffeltes umgeformt werden. Die erste Zeile muß nicht verändert werden. Sie wird verwendet um die Koeffizienten von x_1 in den restlichen Gleichungen verschwinden zu lassen. Man setzt $a_{11} \neq 0$ voraus (Pivotelement). Um den Term $a_{j1}x_1$ in Zeile j, $j = 2, ..., n$ zu eliminieren, subtrahieren wir von der Zeile j ein Vielfaches der Zeile 1 (Pivotzeile), d.h.

$$Zeilej_{(neu)} = Zeilej_{(alt)} - l_{j1}Zeile1$$

es folgt sofort $l_{j1} = a_{j1}/a_{11}$ und die neue Zeile j ist

$$0 + \underbrace{(a_{j2} - l_{j1}a_{12})}_{a'_{j2}} x_2 + \ldots + \underbrace{(a_{jn} - l_{j1}a_{1n})}_{a'_{jn}} x_n = \underbrace{b_j - l_{j1}b_1}_{b'_j}$$

In den Zeilen 2 bis n bleibt eine $(n-1 \times n-1)$-Restmatrix. Wenden wir auf die Restmatrix die Eliminationsschritt erneut an, erhalten wir die Folge

$$A = A^{(1)} \to A^{(2)} \to \ldots \to A^{(n)} =: R$$

$$b = b^{(1)} \to b^{(2)} \to \ldots \to b^{(n)} =: z$$

Da jeder Eliminationsschritt eine lineare Operation auf den Zeilen von A ist, lässt sich der Übergang von $A^{(k)}$ und $b^{(k)}$ zu $A^{(k+1)}$ und $b^{(k+1)}$ als Multiplikation mit einer Matrix L_k von *links* darstellen, d.h.

$$A^{(k+1)} = L_k A^{(k)} \qquad b^{(k+1)} = L_k b^{(k)}$$

wobei

$$L_k = \begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & -l_{k+1,k} & \ddots & \ddots & \vdots \\ \vdots & \vdots & 0 & 1 & 0 \\ 0 & -l_{n,k} & 0 & 0 & 1 \end{pmatrix}$$

Die Matrix L_k hat die Eigenschaft, dass die Inverse L_k^{-1} aus L_k durch einen Vorzeichenwechsel in den Elementen l_{ik} entsteht. Darüber hinaus gilt

$$L := L_1^{-1} \cdot \ldots \cdot L_{n-1}^{-1} = \begin{pmatrix} 1 & 0 & \ldots & \ldots & 0 \\ l_{21} & 1 & \ddots & & \vdots \\ l_{31} & l_{32} & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n1} & \ldots & \ldots & l_{n,n-1} & 1 \end{pmatrix}, \quad L^{-1} = L_{n-1} \cdot \ldots \cdot L_1.$$

Zusammengefasst erhalten wir auf diese Weise das zu $Ax = b$ äquivalente gestaffelte Gleichungssytem $Rx = z$ mit

$$R = L^{-1}A \qquad z = L^{-1}b$$

Bemerkung 3.7. Eine untere oder obere Dreiecksmatrix, deren Diagonalelemente alle gleich eins sind, heißt *unipotent*. Die obige Darstellung $A = LR$ der Matrix A als Produkt einer unipotenten unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix R heißt **Gaußsche Dreieckszerlegung** oder auch **LR-Zerlegung** von A. Für den Rechenaufwand, gezählt in Multiplikationen, ergibt sich

$$\sim \sum_{k=1}^{n-1} k^2 = O\left(\frac{n^3}{3}\right)$$

Algorithmus:

```
for k=1:n-1
   bestimme r(k) ∈ {k,...,n} mit a_{r(k),k} ≠ 0
   % Die Auswahl des Pivotelements a_{r(k),k} sollte numerisch günstig erfolgen
   % Als Beispiel wählen wir |a_{r(k),k}| = max_{k≤i≤n} {|a_ik|}
   (A|b) = P_{k,r(k)} · (A|b)
   % Hierdurch wird die k-te und r(k)-te Zeile vertauscht;
   % danach ist a_kk ≠ 0
   for i=k+1:n
      l_ik = a_ik / a_kk
   end
   l_k = (0,...,0,l_{k+1,k},...,l_{n,k})^T
   L_k = I - l_k e_k^T      % Gauss-Elementar-Matrix
   (A|b) = L_k · (A|b)
end
```

Bemerkung 3.8. Die Vertauschungsmatrix (P_{rs}) wird benötigt um sicherzustellen, dass in der Diagonalen von A kein Eintrag 0 enstehen kann, was zu einer Division durch 0 und somit zu einem Fehler führen würde! Der Verzicht auf diese Matrix heißt diagonale Pivot-Wahl. Auch für reguläres A ist dies nicht immer möglich[1]. Daher sollte man i.A. eine Pivotisierung durchführen. Ein übliches Verfahren ist die Spaltenpivotisierung

$$|a_{r(k),k}| = \max_{k \leq i \leq n} \{|a_{ik}|\}$$

In der Praxis werden für sehr große Matrizen A nicht alle Elemente der Spalte sondern nur eine definierte Zahl durchsucht, da diese Maximumssuche relativ aufwendig ist. Selbst wenn möglich sollte man i.A. nicht auf Pivotisierung verzichten, da Zwischenergebnisse unnötig groß werden können.

SATZ 3.9. *Für reguläre Matrizen ist der Gauß-Algorithmus durchführbar.*

Beweis 3.10. Nach dem ersten Schritt erhält man als Matrix

$$L_1 P_{1p(1)} A \quad = \quad \begin{pmatrix} a_{p(1)1} & * & \dots & * \\ 0 & & & \\ \vdots & & A^{(1)} & \\ 0 & & & \end{pmatrix}$$

und somit

$$|det\left(L_1 P_{1p(1)} A\right)| = |det\left(A\right)|$$

Reguläre Matrizen kann man immer auf Dreiecksgestalt bringen, also ist der Gauß-Algorithmus anwendbar. q.e.d.

SATZ 3.11. *Der Gauß-Algorithmus liefert für $A \in \mathbb{R}^{n \times n}$ eine Permutationsmatrix P, eine Matrix $L \in \Delta_-$ und $R \in \Delta^+$, so dass*

$$PA = LR$$

SATZ 3.12. *Ist A streng diagonaldominant, d.h.*

$$|a_{ii}| > \sum_{j=1 j \neq i}^{n} |a_{ij}|$$

funktioniert die Gauss-Elimination ohne Pivotisierung.

3.2.3 LR-Zerlegung

Manchmal hat man ein lineares Gleichungssystem mit verschiedenen rechten Seiten zu lösen, dann erspart die LR-Zerlegung die wiederholte Faktorisierungsphase und es schliesst sich lediglich die Lösungsphase an. Hierzu wird A in L und R zerlegt.

$$Ax = L \underbrace{Rx}_{=:c} = b$$

und somit

$$\begin{cases} Lc = b \\ Rx = c \end{cases}$$

[1] z.B. $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{2,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \dots & l_{n,n-1} & 1 \end{pmatrix}}_{=:L} \underbrace{\begin{pmatrix} r_{1,1} & \dots & \dots & r_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & r_{n,n} \end{pmatrix}}_{=:R}$$

Dieses Matrixenprodukt besteht aus n^2 Gleichungen. L hat $\frac{n^2-n}{2}$ und R hat $\frac{n^2+n}{2}$ Unbekannte, d.h. insgesamt n^2 Unbekannte. Somit können die Matrizen durch Substitution ermittelt werden:

```
for i=1:n
   L_{i,i} = 1
   for k=1:i-1
      L_{i,k} = (A_{i,k} - sum_{j=1}^{k-1} L_{i,j} R_{j,k}) / R_{k,k}
   end
   for k=1:i
      R_{k,i} = A_{k,i} - sum_{j=1}^{k-1} L_{k,j} R_{j,i}
   end
end
```

Bemerkung 3.13. Um die Zahl der Unbekannten zu vermindern haben wir die Diagonalelemente(l_{ii}) von L selbst bestimmt, $l_{ii} = 1$. Im Allgemeinen werden unterschiedliche Bedingungen für diese n Unbekannten angegeben:

- $l_{ii} = 1 \quad \forall i = 1, \dots, n$ Doolittle-Verfahren
- $r_{ii} = 1 \quad \forall i = 1, \dots, n$ Crout-Verfahren
- $l_{ii} = r_{ii} \quad \forall i = 1, \dots, n$ Cholesky-Verfahren

3.2.4 Cholesky-Verfahren

Wir wollen nun die Gauß-Elimination auf die eingeschränkte Klasse von Gleichungssystemen mit **symmetrisch positiv definiten Matrizen** anwenden. Wir erinnern daran, dass eine symmetrische Matrix $A = A^T \in M(n \times n; \mathbb{R})$ genau dann positiv definit ist, falls $\langle x, Ax \rangle > 0$ für alle $x \neq 0$. Wir nennen solche Matrizen auch kurz *spd-Matrizen.*

SATZ 3.14. *Für jede spd-Matrix $A \in M(n \times n; \mathbb{R})$ gilt:*

(a) A ist invertierbar

(b) $a_{ii} > 0$ für $i = 1, \dots, n$

(c) $max_{i,j=1,\dots,n} |a_{ij}| = max_{i=1,\dots,n} a_{ii}$

(d) Bei der Gauss-Elimination ohne Pivotsuche ist jede Restmatrix wiederum spd.

Offensichtlich besagen (c) und (d), dass eine Spalten- oder Zeilenpivotsuche bei der LR-Zerlegung von A unnötig, ja sogar unsinnig wäre, da sie evtl. die spezielle Struktur von A zerstören würde.

SATZ 3.15. *Für jede spd Matrix A existiert eine eindeutig bestimmte Zerlegung der Form*

$$A = LDL^T$$

wobei L eine unipotente untere Dreiecksmatrix ist und D eine positive Diagonalmatrix ist.

Cholesky-Algorithmus(liefert L,D):

```
for i = 1:n
   for j = 1:i-1
      v_j = a_ij d_j
   end
   d_i = a_ii − Σ_{j=1}^{i-1} a_ij v_j
   for j = i+1:n
      l_ji = (a_ji − Σ_{k=1}^{i-1} a_jk v_k)/d_i
   end
end
```

Lemma 3.16. Da $D = diag(d_{ii})$ positiv ist (d.h. $d_{ii} > 0 \quad \forall i = 1, \ldots, n$), existiert $D^{\frac{1}{2}} = diag(\sqrt{d_{ii}})$ und daher die Cholesky-Zerlegung

$$A = \bar{L}\bar{L}^T$$

wobei $\bar{L} = LD^{\frac{1}{2}}$ ist.

Durch Benutzung der Matrix $\bar{L} = (l_{ij})$ erhalten wir somit das klassische Cholesky-Verfahren, das sich kompakt wie folgt darstellen lässt:

```
for k =1:n
   l_kk = (a_kk − Σ_{j=1}^{k-1} l_kj^2)^{1/2}
   for i = k+1:n
      l_ik = ( a_ik − Σ_{j=1}^{k-1} l_ij l_kj ) / l_kk
   end
end
```

Die Herleitung dieses Algorithmus ist die elementweise Auswertung von

$$\begin{pmatrix} l_{11} & & \\ \vdots & \ddots & \\ l_{n1} & \ldots & l_{nn} \end{pmatrix} \quad \begin{pmatrix} l_{11} & \ldots & l_{n1} \\ & \ddots & \vdots \\ & & l_{nn} \end{pmatrix} \quad = \quad \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix}$$

daraus folgt

$$\begin{aligned} i = k: \quad a_{kk} &= l_{k1}^2 + \ldots + l_{k,k-1}^2 + l_{kk}^2 \\ i > k: \quad a_{ik} &= l_{i1}l_{k1} + \ldots + l_{i,k-1}l_{k,k-1} + l_{ik}l_{kk} \end{aligned}$$

Die Raffinesse der Methode steckt in der Reihenfolge der Berechnung der Elemente von $\bar{L}$. Für den Rechenaufwand zählen wir

$$O\left(\frac{n^3}{6}\right) \text{ Multiplikationen und } n \text{ Quadratwurzeln.}$$

3.2.5 LR-Zerlegung von Bandmatrizen

Definition 3.17. Eine Matrix $A = (a_{ik}) \in \mathbb{K}^{n\times n}$ heißt **Bandmatrix** mit **Bandbreite** d, falls $a_{ik} = 0$ für $|i-k| > d$. Eine Matrix heißt **Tridiagonalmatrix**, wenn sie eine Bandmatrix mit Bandbreite 1 ist:

$$T \quad = \quad \begin{pmatrix} a_1 & b_1 & 0 & \ldots & 0 \\ c_2 & a_2 & b_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & c_{n-1} & a_{n-1} & b_{n-1} \\ 0 & \ldots & 0 & c_n & a_n \end{pmatrix}$$

Bemerkung 3.18. Bandmatrizen spielen in der Anwendung eine große Rolle (Diskretisierung von Differentialgleichungen, ...). Wir wollen zeigen, dass man unter gewissen Vorausetzungen für Tridiagonalmatrizen eine LR-Zerlegung mit einem Aufwand von $O(n)$ findet.

Sei nun T eine reguläre Tridiagonalmatrix. Wir betrachten

$$T = \begin{pmatrix} a_1 & b_1 & 0 & \dots & 0 \\ c_2 & a_2 & b_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & c_{n-1} & a_{n-1} & b_{n-1} \\ 0 & \dots & 0 & c_n & a_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ l_2 & 1 & \ddots & & \vdots \\ 0 & l_3 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & l_n & 1 \end{pmatrix} \begin{pmatrix} r_1 & b_1 & 0 & \dots & 0 \\ 0 & r_2 & b_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & b_{n-1} \\ 0 & \dots & \dots & 0 & r_n \end{pmatrix}$$

Falls so eine Darstellung möglich ist, d.h. falls es derartige l_j, r_j gibt, so muss gelten

$$\begin{cases} r_1 = a_1 \\ l_i = \frac{c_i}{r_{i-1}} \quad i = 2, \dots, n \\ r_i = a_i - l_i \cdot b_{i-1} \quad i = 2, \dots, n \end{cases}$$

Bemerkung 3.19. Damit dieser Algorithums durchführbar ist, muss gelten $r_1, \dots, r_{n-1} \neq 0$. Damit R regulär ist, muss zusätzlich folgen $r_n \neq 0$. Eine hinreichende Bedingung für die Durchführbarkeit des Verfahrens ergibt der folgende Satz.

SATZ 3.20. *Sei T Tridiagonalmatrix und es gelte zusätzlich:*

$$\begin{cases} |a_1| > |b_1| > 0 \\ |a_i| > |b_i| + |c_i| \quad b_i \cdot c_i \neq 0 \quad i = 2, \dots, n-1 \\ |a_n| \geq |c_n| > 0 \end{cases}$$

Dann ist der Algorithmus durchführbar, und es gilt in der vorherigen Bezeichnung $r_i \neq 0$. Mit L, R gilt $T = LR$ und somit ist T regulär.

Algorithmus zur Lösung von $Tx = d$:

```
% Faktorisierung der Tridiagonalmatrix
% Das Gleichungssystem wird zerstört, d.h. a wird durch r und
% c durch l überschrieben.
% a_1 = a_1
for i = 2:n
   % c wird durch Faktoren l ueberschrieben:
   c_i = c_i / a_{i-1}
   % a wird durch Faktoren r ueberschrieben:
   a_i = a_i - c_i b_{i-1}
end
% Lösen der faktorisierten Tridiagonalmatrix:
% Vorwärtseinsetzen:
x_1 = d_1
for i = 2:n
   x_i = d_i - c_i x_{i-1}
end
% Rueckwärtseinsetzen:
x_n = x_n / a_n
for i = (n-1):-1:1
   x_i = (x_i - b_i x_{i+1}) / a_i
end
```

Falls der Algorithmus durchführbar ist, entsteht in x die Lösung des Sytems, in c stehen die l-Werte, in a die r-Werte und der Aufwand ist $O(n)$!

3.3 Iterative Verfahren

Klassische Iterationsverfahren basieren auf der Idee der Fixpunktverfahren. Man definiert eine Abbildung $\Phi(x)$, sodass die Lösung $x = \Phi(x)$ der Fixpunktiteration $x_{k+1} = \Phi(x_k)$ auch die Lösung von $Ax = b$ ist. Am einfachsten erreichen wir das, indem wir die Gleichung $Ax = b$ in eine Fixpunktgleichung umformen,

$$\begin{aligned} Ax = b &\Leftrightarrow Q^{-1}(b - Ax) = 0 \\ &\Leftrightarrow \Phi(x) := \left(I - Q^{-1}A\right)x + Q^{-1}b = x, \end{aligned}$$

wobei Q eine beliebige reguläre Matrix sein kann. Q muss aber so gewählt werden, dass die Iteration $x_{k+1} = \left(I - Q^{-1}A\right)x_k + Q^{-1}b$ konvergiert. Wir verwenden im Folgenden die Konvention

$$M := I - Q^{-1}A \text{ und } N := Q^{-1}.$$

folgen und die Fixpunktinteration $x_{k+1} = Mx_k + Nb$ untersuchen.

SATZ 3.21. *Das Iterationsverfahren $x_{k+1} = Mx_k + Nb$ konvergiert genau dann für jeden Startwert $x_0 \in R^n$, wenn $\rho(M) < 1$, wobei $\rho(M) := \max_j |\lambda_j(M)|$ der Spektralradius von M ist.*

Beweis 3.22. Wir beschränken uns auf den Fall einer symmetrischen Matrix $M = M^T$, den wir im folgenden ausschließlich benötigen. Dann gibt es eine orthogonale Matrix O, so dass

$$OMO^T = \Lambda = diag(\lambda_1, ..., \lambda_n)$$

die Diagonalmatrix der Eigenwerte von M ist. Da $|\lambda_i| \leq \rho(M) < 1$ für alle i , gilt

$$lim_{k\to\infty} M^k = lim_{k\to\infty} O^T \Lambda^k O = 0$$

Sei x die Lösung von $x = Mx + Nb$ und $x_{k+1} = Mx_k + Nb$. Dann ist

$$(x - x_{k+1}) = M(x - x_k) = ... = M^{k+1}(x - x_0).$$

Daraus erhalten wir weiters

$$\|x - x_k\| \leq \left\|M^k\right\| \|x - x_0\| \leq \|M\|^k \|x - x_0\|$$

und schließlich $lim_{k\to\infty} \|x - x_k\| = 0$ woraus die Konvergenz folgt. q.e.d.

Bemerkung 3.23. Da $\rho(M) \leq \|M\|$ für jede Matrixnorm, ist $\|M\| < 1$ hinreichend für die Konvergenz.

Bemerkung 3.24. Es gilt folgende Fehlerabschätzung:

$$\|x - x_k\| \leq \frac{\|M\|^k}{1 - \|M\|} \|x_1 - x_0\|$$

Um diese Abschätzung herzuleiten haben wir $\|x - x_1\| \leq \|M\| \|x - x_0\|$ und

$$\|x - x_0\| \leq \|x - x_1\| + \|x_1 - x_0\| \leq \|M\| \|x - x_0\| + \|x_1 - x_0\|$$

daher $(1 - \|M\|) \|x - x_0\| \leq \|x_1 - x_0\|$. Da $\|x - x_k\| \leq \|M\|^k \|x - x_0\|$, folgt die Aussage.
Die am besten invertierende Matrix ist zweifellos die Identität $Q = I$. Das so entstehende Verfahren mit $M = I - A$

$$x_{k+1} = x_k + (b - Ax_k) \Leftrightarrow x_{k+1} = (I - A)x_k + b$$

ist das sogenannte **Richardson-Verfahren**. Gehen wir von einer spd-Matrix A aus, so ergibt sich für den Spektralradius von M gerade

$$\rho(M) = \rho(I - A).$$

Eine notwendige Bedingung für die Konvergenz der Richardson-Iteration ist daher $\lambda_{max} < 2$. Für sich genommen ist diese Iteration demnach nur selten verwendbar. Für die weiteren Verfahren verwenden wir die Zerlegung

$$A = L + D + R \qquad \text{(Splitting)}$$

wobei $D = diag\{a_{11}, ..., a_{nn}\}$ und

$$L := \begin{pmatrix} 0 & \dots & \dots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn-1} & 0 \end{pmatrix}, R := \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1n} \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

3.3.1 Jacobi-Verfahren

Man wählt hier $Q = D$. Das zugehörige Verfahren

$$x_{k+1} = \left(I - D^{-1}A\right) x_k + D^{-1}b = -D^{-1}(L + R) x_k + D^{-1}b$$

heißt *Jacobi-Verfahren*.

SATZ 3.25. *Die Jakobi-Iteration konvergiert für jeden Startwert x_0 gegen die Lösung $x = A^{-1}b$, falls A strikt diagonaldominant ist, d.h.*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \qquad i = 1, ..., n$$

Beweis 3.26.

$$\rho\left(D^{-1}(L + R)\right) \leq \left\|D^{-1}(L + R)\right\|_\infty = \max_i \sum_{j \neq i} \left|\frac{a_{ij}}{a_{ii}}\right| < 1.$$

q.e.d.

Um einen Algorithmus für dieses Verfahren formulieren zu können, schauen wir uns das Verfahren aufgespaltet in die einzelnen Zeilen an:

$$x_{k+1} = -D^{-1}(L + R) x_k + D^{-1}b \Leftrightarrow$$

$$\begin{pmatrix} x_{(1)k+1} \\ \vdots \\ \vdots \\ x_{(n)k+1} \end{pmatrix} = - \begin{pmatrix} \frac{1}{a_{11}} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{a_{nn}} \end{pmatrix} \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{pmatrix} \begin{pmatrix} x_{(1)k} \\ \vdots \\ \vdots \\ x_{(n)k} \end{pmatrix}$$

$$+ \begin{pmatrix} \frac{1}{a_{11}} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{a_{nn}} \end{pmatrix} \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\Rightarrow x_{(i)k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_{(j)k} \right)$$

Damit erhalten wir das Jacobi-Verfahren:

```
% Input: A, b, x_0, k_max
%    A ...strikt-diagonaldominante-Matrix
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Iterationszahl
for k=0:k_max
   x(neu) = x
   for i=1:n
      x(neu)_i = 1/a_ii ( - sum_{j=1,j≠i}^{n} a_ij x_j + b_i )
   end
   if ||x(neu) - x|| < ε return
   else x = x(neu)
end
```

3.3.2 Gauss-Seidel-Verfahren

Wir setzen hier $Q := D + L$ (die untere Dreieckshälfte von A) an, so erhalten wir das Gauss-Seidel-Verfahren

$$x_{k+1} = \left(I - (D+L)^{-1} A\right) x_k + (D+L)^{-1} b = -(D+L)^{-1} R x_k + (D+L)^{-1} b.$$

Es konvergiert für jede spd-Matrix A. Um dies nachzuweisen, leiten wir eine einfach zu überprüfende hinreichende Bedingung für die Kontraktionseigenschaft $\rho(M) < 1$ von $M = I - Q^{-1}A$ her. Dazu betrachten wir, dass jede spd-Matrix A ein Skalarprodukt $(x, y) := \langle x, Ay\rangle$ auf dem $\mathbb{R}^n$ induziert. Für jede Matrix $B \in \mathbb{R}^{n\times n}$ ist $B^* := A^{-1}B^T A$ die bezüglich $(\cdot,\cdot)$ adjungierte Matrix, d.h.

$$(Bx, y) = (x, B^* y) \quad \forall x, y \in \mathbb{R}^n$$

Eine selbstadjungierte Matrix $B = B^*$ heißt dann positiv bezüglich $(\cdot,\cdot)$ falls

$$(Bx, x) > 0 \quad \forall x \neq 0.$$

SATZ 3.27. *Sei $M \in \mathbb{R}^{n\times n}$ und M^* die bezüglich eines Skalarproduktes $(\cdot,\cdot)$ adjungierte Matrix von M. Ist dann $B = I - M^*M$ positiv selbstadjungiert, dann gilt:*

$$\rho(M) < 1$$

Beweis 3.28. Da B positiv ist, gilt für alle $x \neq 0$

$$(Bx, x) = (x, x) - (M^*Mx, x) = (x, x) - (Mx, Mx) > 0$$

und daher, mit der Norm $\|x\| = \sqrt{(x, x)}$, dass $\|x\| > \|Mx\|$ für alle $x \neq 0$.Daraus folgt:

$$\rho(M) \leq \|M\| := \sup_{\|x\|\neq 0} \frac{\|Mx\|}{\|x\|} < 1$$

q.e.d.

SATZ 3.29. *Das Gauss-Seidel-Verfahren konvergiert für jede spd-Matrix A.*

Beweis 3.30. Wir müssen zeigen, dass $B := I - M^*M$ mit $M = I - (D+L)^{-1} A$ eine positive Matrix bzgl. $(\cdot,\cdot) := \langle\cdot, A\cdot\rangle$ ist. Nun ist wegen $R^T = L$

$$M^* = I - A^{-1}A^T (D+L)^{-T} A = I - (D+R)^{-1} A$$

und daher

$$\begin{aligned}
B &= I - M^*M \\
&= I - \left[I - (D+R)^{-1} A\right] \left[I - (D+L)^{-1} A\right] \\
&= I - \left[(D+R)^{-1} [D+R-A]\right] \left[(D+L)^{-1} [D+L-A]\right] \\
&= I - \left[(D+R)^{-1} (-L)\right] \left[(D+L)^{-1} (-R)\right] \\
&= I - (D+R)^{-1} L (D+L)^{-1} R \\
&= (D+R)^{-1} \left[(D+R) - L (D+L)^{-1} (A - (D+L))\right] \\
&= (D+R)^{-1} \left[D+R - L (D+L)^{-1} A + L\right] \\
&= (D+R)^{-1} \left[A - L (D+L)^{-1} A\right] \\
&= (D+R)^{-1} \left[I - L (D+L)^{-1}\right] A \\
&= (D+R)^{-1} [D+L-L] (D+L)^{-1} A \\
&= (D+R)^{-1} D (D+L)^{-1} A
\end{aligned}$$

Damit folgt für alle $x \neq 0$, dass

$$\begin{aligned}
(Bx, x) &= \left\langle (D+R)^{-1} D (D+L)^{-1} Ax, Ax \right\rangle \\
&= \left\langle D^{\frac{1}{2}} (D+L)^{-1} Ax, D^{\frac{1}{2}} (D+L)^{-1} Ax \right\rangle > 0,
\end{aligned}$$

d.h. B ist positiv und $\rho(M) < 1$. q.e.d.

Für den Algorithmus schauen wir uns das Verfahren wieder etwas genauer an, indem wir es in die einzelnen Zeilen zerlegen:

$$\begin{aligned}
x_{k+1} &= -(D+L)^{-1} R x_k + (D+L)^{-1} b \\
(D+L) x_{k+1} &= b - R x_k
\end{aligned}$$

$$\begin{pmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ a_{n1} & \dots & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_{(1)k+1} \\ \vdots \\ \vdots \\ x_{(n)k+1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix} - \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{pmatrix} \begin{pmatrix} x_{(1)k} \\ \vdots \\ \vdots \\ x_{(n)k} \end{pmatrix}$$

$$\Rightarrow x_{i(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{j(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_{j(k)} \right)$$

Damit erhalten wir den Gauss-Seidel-Algorithmus:

```
% Input: A, b, x_0, k_max
%    A ...spd-Matrix
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Iterationszahl
for k=0:k_max
   x(neu) = x
   for i=1:n
      x(neu)_i = 1/a_ii ( b_i - sum_{j=1}^{i-1} a_ij x(neu)_j - sum_{j=i+1}^{n} a_ij x_j )
   end
   if ||x(neu) - x|| < ε return
   else x = x(neu)
end
```

SATZ 3.31 (Stein-Rosenberg)**.** *Wenn* $a_{ij} \leq 0$ *für alle* $i \neq j$, $a_{ii} > 0$ *für* $i = 1, \ldots, n$ *und* $M_J = -D^{-1}(L+R)$, $M_{GS} = -(D+L)^{-1}R$, *dann gilt, dass eine der folgenden Situationen zutrifft:*

1. $0 < \rho(M_{GS}) < \rho(M_J) < 1$
2. $1 < \rho(M_J) < \rho(M_{GS})$
3. $\rho(M_J) = \rho(M_{GS}) = 1$

3.3.3 Konvexkombinationen

Die Konvergenzgeschwindigkeit einer Fixpunktiteration hängt stark von dem Spektralradius $\rho(M)$ ab. Für jedes konkret gewählte M gibt es jedoch ein einfaches Mittel, diesen zu verbessern, nämlich die sogenannte **Extrapolation** oder besser **Relaxation**. Dazu betrachten wir Konvexkombinationen,

$$x_{k+1} = \omega(Mx_k + Nb) + (1-\omega)x_k$$

wobei $\omega \in (0,1)$ ein sogenannter *Dämpfungsparameter* ist. Auf diese Weise gewinnen wir eine Familie von *relaxierten Fixpunktiterationsverfahren* mit Parameter ω. Durch geeignete Wahl von ω kann man versuchen Konvergenz zu erzwingen, obwohl die Ausgangsiteration i.a. nicht konvergiert, oder das Iterationsverfahren zu optimieren.
Falls M konvergiert kann man versuchen $\omega > 1$ zu wählen, um die Konvergenz zu beschleunigen. Insbesondere verwendet man $\omega > 1$ zusammen mit dem Gauss-Seidel-Verfahren. Dies ergibt das SOR-Verfahren.

SOR-Verfahren

Für das SOR-Verfahren (successive over relaxation) definieren wir $Q = \frac{1}{\omega}D + L$ mit $\omega \in (0,2)$. Daraus erhalten wir folgendes Verfahren

$$x_{k+1} = \left(I - \left(\frac{1}{\omega}D + L\right)^{-1} A\right) x_k + \left(\frac{1}{\omega}D + L\right)^{-1} b.$$

Um einen Algorithmus für dieses Verfahren formulieren zu können, schauen wir uns das Verfahren aufgespaltet in die einzelnen Zeilen an:

$$\begin{aligned} x_{k+1} &= \left(I - \left(\frac{1}{\omega}D + L\right)^{-1} A\right) x_k + \left(\frac{1}{\omega}D + L\right)^{-1} b \\ x_{k+1} - x_k &= \left(\frac{1}{\omega}D + L\right)^{-1} b - \left(\frac{1}{\omega}D + L\right)^{-1} Ax_k \\ \left(\frac{1}{\omega}D + L\right)(x_{k+1} - x_k) &= b - Ax_k \end{aligned}$$

$$\begin{pmatrix} \frac{a_{11}}{\omega} & 0 & \dots & 0 \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n1} & \dots & a_{n,n-1} & \frac{a_{nn}}{\omega} \end{pmatrix} \left[\begin{pmatrix} x_{(1)k+1} \\ \vdots \\ \vdots \\ x_{(n)k+1} \end{pmatrix} - \begin{pmatrix} x_{(1)k} \\ \vdots \\ \vdots \\ x_{(n)k} \end{pmatrix} \right] = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix} - \begin{pmatrix} a_{11} & \dots & \dots & a_{1n} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ a_{n1} & \dots & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_{(1)k} \\ \vdots \\ \vdots \\ x_{(n)k} \end{pmatrix}$$

$$\begin{aligned} &\Rightarrow \quad \frac{a_{ii}}{\omega}\left(x_{(i)k+1} - x_{(i)k}\right) + \sum_{j=1}^{i-1} a_{ij}\left(x_{(j)k+1} - x_{(j)k}\right) = b_i - \sum_{j=1}^{n} a_{ij}x_{(j)k} \\ &\Leftrightarrow \quad \frac{1}{\omega}\left(x_{(i)k+1} - x_{(i)k}\right) = \underbrace{\frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_{(j)k+1} - \sum_{j=i+1}^{n} a_{ij}x_{(j)k}\right)}_{=:y_{(i)k+1}(=Gauss-Seidel)} - x_{(i)k} \\ &\Leftrightarrow \quad x_{(i)k+1} = \omega y_{(i)k+1} + (1-\omega)x_{(i)k} \end{aligned}$$

Damit haben wir die Konvexkombination des Gauss-Seidel-Verfahrens verifiziert, um die Herleitung einmal gesehen zu haben. Natürlich hätten wir auch einfach in die Darstellung einsetzen können. Für den Algorithmus erhalten wir

```
% Input: A, b, x_0, k_max, ω
%    A ... spd-Matrix
%    b ... rechte Seite
%    x_0 ... Startwert
%    k_max ... maximale Iterationszahl
%    ω ∈ (0,2) ... Dämpfungsparameter
for k=0:k_max
   x(neu) = x
   for i=1:n
      x(neu)_i = 1/a_ii ( b_i - Σ_{j=1}^{i-1} a_ij x(neu)_j - Σ_{j=i+1}^{n} a_ij x_j )
      x(neu)_i = ω x(neu)_i + (1-ω) x_i
   end
   if ||x(neu) - x|| < ε return
   else x = x(neu)
end
```

SATZ 3.32. *Für das SOR-Verfahren gilt* $M = I - \left(\frac{1}{\omega}D + L\right)^{-1} A$. *Falls* $a_{ii} \neq 0, i = 1, ..., n$, *dann gilt*

$$\rho(M) \geq |\omega - 1|$$

Das heißt $\rho(M) < 1$ *nur wenn* $0 < \omega < 2$.

SATZ 3.33. *Das SOR-Verfahren konvergiert für jede spd-Matrix A, mit* $0 < \omega < 2$.

SATZ 3.34. *Sei A eine spd - Matrix mit tridiagonaler Gestalt. Dann gilt: $\rho(M_{GS}) = [\rho(M_J)]^2 < 1$ und das optimale ω für das SOR - Verfahren ist:*

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(M_{GS})}}$$

Damit erhält man $\rho(M_{SOR}) = \omega - 1$.

3.3.4 Alternativer Ansatz

Für die Lösung von $Ax = b$, sei $\overline{x}$ eine Approximation zu x und $r = b - A\overline{x}$ das **Residuum** und $e = x - \overline{x}$ der Fehler.
Das Lösen des Systems $Ax = b$ ist äquivalent mit dem Lösen von $Ae = r$, denn dann gilt:

$$x = \overline{x} + e$$

Statt das System $Ae = r$ zu lösen, betrachten wir das Problem $Se = r$, wobei S eine Approximation von A ist. S sollte so gewählt werden, dass S^{-1} leichter zu berechnen ist als A^{-1}. Ein iteratives Verfahren ist gegeben durch:

```
% Input: A, S, b, x_0, k_max
%    A...Matrix des zu lösenden Systems
%    S...Approximierte Matrix von A
%    b...rechte Seite
%    x_0...Startwert
%    k_max...maximale Iterationszahl
x^alt = x_0
for k=0:k_max
   r = b - A x^alt   % Residuum
   e = S^-1 r    % Lösen von Se = r
   x^neu = x^alt + e
   if ||r|| < ε return
   else x^alt = x^neu
end
```

Aus dem bereits formulierten wollen wir nun genauere Angaben über die Vorgehensweise geben:

$$\begin{aligned} x^{neu} &= x^{alt} + e \\ &= x^{alt} + S^{-1}r \\ &= x^{alt} + S^{-1}\left(b - Ax^{alt}\right) \\ &= \left(I - S^{-1}A\right)x^{alt} + S^{-1}b \end{aligned}$$

Also haben wir für $T = S - A$ erhalten:

$$x^{neu} = S^{-1}Tx^{alt} + S^{-1}b$$

Man erkennt sofort, dass S genau die Rolle Q, welches in den Kapiteln zuvor definiert wurde, spielt. D.h. setzt man $S = Q$ kann man die zuvor besprochenen Algorithmen neuerlich anschreiben, was wir hier aber nur andeuten. Es sei $A = D + L + R$, dann gilt

- (Jakobi): $S = D$
- (Gauss-Seidel): $S = D + L$
- (SOR): $S = \frac{1}{\omega}D + L$

Definition 3.35 (Asymptotischer Konvergenzfaktor)**.**

$$\alpha = \sup_{x^0 \neq 0} \inf_k \sup_{n \geq k} \|x^n - x^*\|^{\frac{1}{n}}$$

Wegen der Norm Äquivalenz und $\lim_{n\to\infty} c^{1/n} = 1$ für jede Konstante $c \neq 0$, ist α Norm unabhängig.

Theorem 3.36. Wenn $\rho(M) < 1$, dann gilt

$$\alpha = \rho(M)$$

Definition 3.37. Die Zerlegung $A = S - T$ ist *regulär*, wenn gilt

$$S^{-1} \geq 0 \text{ und } T \geq 0$$

SATZ 3.38. *Wenn $A^{-1} \geq 0$ und $A = S - T$ eine reguläre Zerlegung ist, dann gilt*

$$\rho\left(S^{-1}T\right) < 1$$

Definition 3.39. Eine nicht singuläre Matrix A ist eine M-Matrix, falls gilt:

$$a_{ij} \leq 0 \text{ für } i \neq j \text{ und } A^{-1} \geq 0$$

SATZ 3.40. *Sei A eine M-Matrix, dann gilt:*
Die Jakobi-Iteration und die Gauss-Seidel-Iteration konvergieren (d.h. die Jakobi-Zerlegung und die Gauss-Seidel-Zerlegung sind regulär).

Bemerkung 3.41. All diese Iterationen sind von der Form:

$$x^{k+1} = x^k + \alpha_k p_k$$

wobei $\alpha_k = \frac{\langle p_k, r_k\rangle}{\langle Ap_k, p_k\rangle}$ die Schrittlänge und p_k die Richtung ist (z.B., $p_k = e_k$ Koordinate Richtung, ergibt Jacobi Iteration). Dieser allgemeine Ansatz führt uns zu weiteren Verfahren, die das folgende Kapitel prägen.

3.3.5 Gradientenverfahren

SATZ 3.42. *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann ist das Lösen von $Ax = b$ äquivalent zur Minimierung der quadratischen Form $Q : \mathbb{R}^n \to \mathbb{R}$*

$$Q(x) = \frac{1}{2}\langle x, Ax\rangle - \langle x, b\rangle.$$

Beweis 3.43. Sei $x \in \mathbb{R}^n, v \in \mathbb{R}^n \setminus \{0\}$ und $t \in \mathbb{R}$. Wir betrachten Q auf der Geraden $x + tv$:

$$\begin{aligned} Q(x+tv) &= Q(x) + t\langle Ax, v\rangle + \frac{1}{2}t^2\langle Av, v\rangle - t\langle v, b\rangle \\ &= Q(x) + t\langle Ax - b, v\rangle + \frac{1}{2}t^2 \underbrace{\langle Av, v\rangle}_{>0} =: f(t), f : \mathbb{R} \to \mathbb{R} \end{aligned}$$

Die Funktion f ist also eine nach oben geöffnete Parabel, diese hat ein *Minimum* bei

$$f'(t) = \langle Ax - b, v\rangle + t\langle Av, v\rangle = 0 \Leftrightarrow t = \frac{\langle b - Ax, v\rangle}{\langle Av, v\rangle}$$

Auf der Geraden $x + tv$ hat Q also in $x + t_0 v$ den Minimalwert

$$\begin{aligned} Q(x + t_0 v) &= Q(x) + t_0(\langle Ax - b, v\rangle + \frac{1}{2}\langle b - Ax, v\rangle) \\ &= Q(x) - \frac{(\langle b - Ax, v\rangle)^2}{2\langle Av, v\rangle} \begin{cases} = Q(x), & v \perp b - Ax \\ < Q(x), & sonst \end{cases} \end{aligned}$$

Ist also x keine Lösung von $Ax = b$, dann gibt es ein v, so dass

$$\langle b - Ax, v\rangle \neq 0$$

und somit kann x nicht Minimum von Q sein.
Löst andererseits x die Gleichung $Ax = b$, gibt es keine Gerade durch x, auf der Q einen kleineren Wert annimmt als $Q(x)$. Ein solches x muss also ein Minimum sein. Weil darüberhinaus $A = Q''$ und A positiv definit ist, ist x ein globales Minimum. Tatsächlich, für jede v

$$Q(x + v) = Q(x) + \frac{1}{2}\langle Av, v\rangle \geq Q(x).$$

q.e.d.

Bemerkung 3.44. **Anwendung auf iterative Lösungen** von $Ax = b$:
Idee: Starte mit x_0, wähle mit einer geeigneten Strategie - wir lernen verschiedene kennen - eine **Suchrichtung** v_0 und setze

$$x_1 := x_0 + t_0 v_0, \quad t_0 = \frac{(\langle b - Ax_0, v_0\rangle)}{\langle Av_0, v_0\rangle}.$$

Wiederhole diesen Vorgang mit $x_1, v_1, t_1, \ldots$
Allgemein: $x_{k+1} := x_k + t_k v_k$ für eine Abstiegsrichtung v_k und t_k wie oben.

Bemerkung 3.45. Es gilt $\langle r_{k+1}, v_k\rangle = 0$. Wir haben $r_{k+1} = A(x - x_{k+1})$ und

$$\langle r_{k+1}, v_k\rangle = \langle A(x - x_{k+1}), v_k\rangle = \langle A(x - x_k) + A(x_k - x_{k+1}), v_k\rangle = \langle r_k, v_k\rangle - t_k\langle Av_k, v_k\rangle = 0.$$

Gradientenverfahren-Steilste Abstieg

Sei $\nabla Q(x)$ der Gradient von Q und v_k ein Vektor, wir haben

$$\langle \nabla Q(x), v_k\rangle = \|\nabla Q(x)\|_2 \cdot \|v_k\|_2 \cdot \cos(\theta)$$

Der Vektor v_k definiert eine **Abstiegsrichtung** genau dann, wenn $\angle(\nabla Q(x), v_k) < 0$, d.h., dass $\cos(\theta) < 0$ ist. Für $\theta = -\pi$ ist der Abstieg am steilsten, d.h. wenn v_k parallel und mit gleiche Orientierung zu $-\nabla Q(x)$ ist.
Daher nimmt man als Richtung den steilsten Abstieg

$$-\nabla Q(x) = -\frac{1}{2}(A + A^T)x + b = (b - Ax)$$

also für x, die (noch) keine Lösung von $Ax = b$ sind.
Nach Konstruktion ist

$$x_{k+1} := x_k + t_k v_k, \quad k = 0, 1, \ldots \quad \text{(Gradientenschritte)}$$

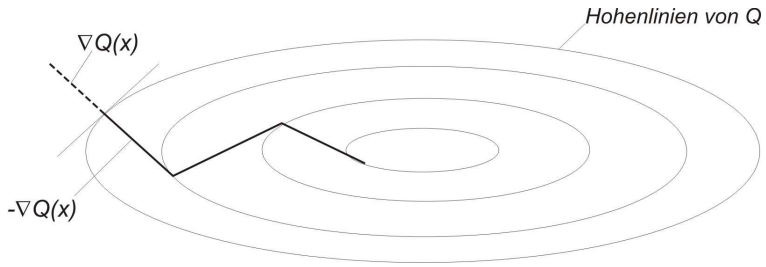
mit

$$t_k := \frac{\langle v_k, v_k\rangle}{\langle Av_k, v_k\rangle}$$

optimal bezüglich der Richtung

$$v_k := b - Ax_k \quad \text{(Residuum)}$$

Geometrische Interpretation:



Nachteil: Unter Umständen langsame Konvergenz.

Die Wiederholung dieser Vorgangsweise bis ein Abbruch-Kriterium erreicht ist, d.h. v klein genug oder die maximale Iterationsschrittzahl erreicht ist, liefert uns folgenden Algorithmus:

```
% Input: A, b, x_0, k_max
%    A ...spd-Matrix
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Interationszahl
for k=0:k_max
   v_k = b - A x_k
   t_k = (v_k^T · v_k) / (v_k^T · (A v_k))
   x_{k+1} = x_k + t_k · v_k
   if ||v_k|| < ε return
end
```

In diesem Algorithmus brauchen wir zwei (Matrix-Vektor) Multiplikationen Ax_k und Av_k. Eine kann erspart werden durch (hier ist $v_k = r_k$):

$$\begin{aligned} x_{k+1} &= x_k + t_k v_k \\ -Ax_{k+1} &= -Ax_k - t_k Av_k \\ \underbrace{b - Ax_{k+1}}_{r_{k+1}} &= \underbrace{b - Ax_k}_{r_k} - t_k Av_k \\ r_{k+1} &= r_k - t_k Av_k \end{aligned}$$

Daher brauchen wir nur Av_k und erhalten somit einen verbesserten Algorithmus für das Gradienten-Verfahren:

```
% Input: A, b, x_0, k_max
%    A ...spd-Matrix
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Interationszahl
v_0 = b − Ax_0
for k=0:k_max
   c_k = Av_k
   t_k = ⟨v_k,v_k⟩ / ⟨v_k,c_k⟩
   x_{k+1} = x_k + t_k v_k
   v_{k+1} = v_k − t_k c_k
   if ‖v_k‖ < ϵ return
end
```

Für die Fehlerabschätzung gilt

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

Konjugiertes Gradientenverfahren (Conjugate gradient method)

Definition 3.46. Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann heißen $p_0, \ldots, p_{n-1} \in \mathbb{R}^n$ **paarweise konjugiert** oder **A-orthogonal**, wenn gilt

$$\langle p_i, Ap_j\rangle = \begin{cases} 0 & 0 \leq i \neq j \leq n-1 \\ \|p_i\|_A^2 & i = j \end{cases}$$

$p_0, \ldots, p_{n-1} \in \mathbb{R}^n$ heißen **A-orthonormiert**, wenn gilt

$$\langle p_i, Ap_j\rangle = \delta_{ij} \qquad 0 \leq i, j \leq n-1$$

Idee: Erweiterung der Optimalität von x_{k+1} bezüglich *span* $\{p_0, \ldots, p_{n-1}\}$ für linear unabhängige Suchrichtungen v_i. Sind die Suchrichtungen p_i speziell A-orthonormiert, so folgt $U^T AU = I$ mit $U = (p_0, \ldots, p_{n-1})$ regulär, d.h. $p_0, \ldots, p_{n-1}$ bilden eine *Basis*. Wir werden zeigen, dass man im n-ten Schritt die Lösung von $Ax = b$ findet, wenn man diese p_i als Suchrichtung bei der schrittweisen Minimierung der quadratischen Funktion Q wählt.

SATZ 3.47. *Seien $p_0, \ldots, p_{n-1} \in \mathbb{R}^n$ A-orthonormiert und $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Es sei weiters*

$$x_{k+1} := x_k + t_k p_k \quad t_k := \langle b - Ax_k, p_k\rangle \quad 0 \leq k \leq n-1, \quad x_0 \in \mathbb{R}^n \text{ beliebig}$$

Dann gilt $Ax_n = b$.

Beweis 3.48. Es gilt

$$Ax_{k+1} = Ax_k + t_k Ap_k \qquad k = 0, \ldots, n-1.$$

$\Rightarrow$

$$\begin{aligned} Ax_n &= Ax_{n-2} + t_{n-1}Ap_{n-1} + t_{n-2}Ap_{n-2} \\ &= \ldots \\ &= Ax_0 + t_0 Ap_0 + \ldots + t_{n-1}Ap_{n-1} \end{aligned}$$

Damit ergibt sich

$$\begin{aligned} t_k &= \langle b - Ax_k, p_k\rangle \\ &= \langle b - Ax_0, p_k\rangle + \langle Ax_0 - Ax_1, p_k\rangle + \langle Ax_1 - Ax_2, p_k\rangle + \ldots + \langle Ax_{k-1} - Ax_k, p_k\rangle \\ &= \langle b - Ax_0, p_k\rangle - t_0 \underbrace{\langle Ap_0, p_k\rangle}_{=0} - \ldots - t_{k-1} \underbrace{\langle Ap_{k-1}, p_k\rangle}_{=0} \\ &= -\langle Ax_0 - b, p_k\rangle \end{aligned}$$

Daraus folgt

$$\langle Ax_n - b, p_k\rangle = \langle Ax_0 - b, p_k\rangle + t_k = 0 \qquad k = 1, \ldots, n.$$

Weil $p_0, \ldots, p_{n-1}$ Basis im $\mathbb{R}^n$ ist, folgt die Behauptung. q.e.d.

Es stellt sich jetzt die Frage wie man ein A-orthonormierte basis $p_1, \ldots, p_n \in \mathbb{R}^n$ konstruieren kann. Eine Antwort liefert folgende

Gram-Schmidt-Orthogonalisierung:

In einem Prähilbertraum H mit Norm $\|x\|_2^2 := \langle x, x\rangle$ und linear unabhängigen $a_0, \ldots, a_{n-1}$ wird durch

$$h_i = a_i - \frac{\sum_{i<j}\langle a_i, n_j\rangle n_j}{\left\|\sum_{i<j}\langle a_i, n_j\rangle n_j\right\|_2} \qquad 0 \le i \le k$$

ein Orthonormalsystem für $span\,\{a_0, \ldots, a_{n-1}\}$ definiert. Das Gram-Schmidt-Verfahren angewandt auf die Standardeinheitsvektoren $e_1, \ldots, e_n$ liefert

$$p_i := \frac{v_i}{\|v_i\|_A}$$

mit

$$v_i = e_i - \sum_{j<i}\langle e_j, p_j\rangle_A\, p_j = e_i - \sum_{j<i}(Ap_j)_i\, p_j.$$

Die resultierenden $p_0, \ldots, p_{n-1} \in \mathbb{R}^n$ sind A-orthogonal.

Jedes p_j ist also eine Linearkombination der Einheitsvektoren und $U = (p_0, \ldots, p_{n-1})$ eine rechte obere Dreiecksmatrix. Aus

$$U^T A U = I$$

folgt

$$A = \underbrace{\left(U^T\right)^{-1}}_{L}\underbrace{(U)^{-1}}_{R},$$

d.h. man kann dieses Verfahren auch als eine spezielle Gauss-Variante zur Lösung von $Ax = b$ auffassen.

SATZ 3.49. *Seien $v_0, \ldots, v_{n-1} \in \mathbb{R}^n \setminus \{0\}$ A-orthogonal, $A \in \mathbb{R}^{n\times n}$ symmetrisch, positiv definit*

$$x_{k+1} := x_k + \frac{\langle b - Ax_k, v_k\rangle}{\langle Av_k, v_k\rangle} v_k, \quad 0 \le k \le n-1, \quad x_0 \in \mathbb{R}^n \text{ beliebig}$$

Dann gilt $Ax_n = b$.

Das Problem in der Gram-Schmidt-Orthogonalisierung ist, dass alle Richtungen p_j zu verfügung (gespeichert) sein müssen. Es gibt einen Ausweg durch folgende Ansatz:

$$p_{k+1} := r_{k+1} + s_k\, p_k$$

und $p_0 = r_0$.

Bemerkung 3.50. Es gilt $\langle r_{k+1}, p_{k+1}\rangle = \langle r_{k+1}, r_{k+1}\rangle$.
Nachdem $p_{k+1} := r_{k+1} + s_k p_k$ und $\langle r_{k+1}, p_k\rangle = 0$.

Daraus folgt insbesondere dass

$$t_k = \frac{\langle r_k, p_k\rangle}{\langle Ap_k, p_k\rangle} = \frac{\langle r_k, r_k\rangle}{\langle Ap_k, p_k\rangle}$$

und weiter erhalten wir dass $\langle r_{k+1}, r_k\rangle = 0$. Das kann man folgenden zeigen

$$\langle r_{k+1}, r_k\rangle = \langle r_k, r_k\rangle - t_k\langle Ap_k, r_k\rangle = \langle r_k, r_k\rangle - t_k\langle Ap_k, p_k - s_{k-1}\, p_{k-1}\rangle = 0.$$

Hier wird s_k so gewählt dass $\langle p_{k+1}, Ap_k\rangle = 0$. Wir haben

$$0 = \langle Ap_{k+1}, p_k\rangle = \langle Ar_{k+1}, p_k\rangle + s_k\,\langle Ap_k, p_k\rangle$$

und weiter

$$s_k = -\frac{\langle Ar_{k+1}, p_k\rangle}{\langle Ap_k, p_k\rangle} = -\frac{\langle r_{k+1}, t_k Ap_k\rangle}{\langle p_k, t_k Ap_k\rangle} = -\frac{\langle r_{k+1}, r_k - r_{k+1}\rangle}{\langle p_k, r_k - r_{k+1}\rangle} = \frac{\langle r_{k+1}, r_{k+1}\rangle}{\langle r_k, r_k\rangle}$$

Für die Durchführung erhalten wir folgenden Algorithmus für das Konjugierte Gradientenverfahren:

```
% Input: A, b, x_0, k_max
%    A ...spd-Matrix
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Interationszahl
p_0 = r_0 = b - A x_0
for k=0:k_max
   t_k = <r_k,r_k> / <Ap_k,p_k>
   x_{k+1} = x_k + t_k p_k
   r_{k+1} = r_k - t_k A p_k
   if |r_{k+1}| < ε return
   s_k = <r_{k+1},r_{k+1}> / <r_k,r_k>
   p_{k+1} = r_{k+1} + s_k p_k
end
```

SATZ 3.51. *Der Approximationsfehler $x - x_k$ des konjugierten Gradientenverfahrens läßt sich in der Energienorm folgendermaße abschätzen*

$$\|x - x_k\|_A \leq 2\left(\frac{\sqrt{K_2(A)} - 1}{\sqrt{K_2(A)} + 1}\right)^k \|x - x_0\|_A$$

$\|x\|_A = \sqrt{\langle Ax, x\rangle}$

Vorkonditionierung

Die Abschätzung der Konvergenzgeschwindigkeit für das konjugierte Gradientenverfahren (wie auch bei anderen) hängt von der Kondition ab. Es ist möglich die Kondition des Problems $Ax = b$ zu verringern.
Sei B eine invertierbare Matrix. Dann ist das Lösen von $Ax = b$ äquivalent zu $\overline{A}\overline{x} = b$ wobei $\overline{x} = B^{-1}x$ und $\overline{A} = AB$. Seien A und B spd. Dann ist die Matrix $\overline{A} = AB$ zwar nicht mehr selbstadjungiert bzgl. $\langle\cdot,\cdot\rangle$, wohl aber bzgl. $(\cdot,\cdot)_B$=$\langle\cdot, B\cdot\rangle$.

$$(x, ABy)_B = \langle x, BABy\rangle = \langle ABx, By\rangle = (ABx, y)_B$$

Daher sind das Gradientenverfahren sowie auch das konjugierte Gradientenverfahren wieder anwendbar, wenn $(\cdot,\cdot)_B$ die Rolle von $\langle\cdot,\cdot\rangle$ übernimmt. Deshalb verwenden wir

$$\left(\overline{A}\cdot,\cdot\right)_B = \langle AB\cdot, B\cdot\rangle = (\cdot,\cdot)_{AB}$$

anstatt von $\langle A\cdot,\cdot\rangle$. B nennen wir Vorkonditionierer (Preconditioner).
Vorkonditioniertes konjugiertes Gradientenverfahren Algorithmus I PCG:

```
% Input: A, B, b, x_0, k_max
%    A ...spd-Matrix
%    B ...Vorkonditionierer
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Interationszahl
p_1 = r_0 = b - ABx_0
for k=1:k_max
   t_k = (r_{k-1},r_{k-1})_B / (p_k,p_k)_AB = <r_{k-1},Br_{k-1}> / <ABp_k,Bp_k>
   x_k = x_{k-1} + t_k p_k
   r_k = r_{k-1} - α_k ABp_k
   if |r_k| < ε return
   s_{k+1} = (r_k,r_k)_B / (r_{k-1},r_{k-1})_B = <r_k,Br_k> / <r_{k-1},Br_{k-1}>
   p_{k+1} = r_k + s_{k+1} p_k
end
```

Eine sparsamere Version des vorkonditionierten konjugierten Gradientenverfahrens erhält man, wenn man mit $q_k = Bp_k$ arbeitet:

Vorkonditioniertes konjugiertes Gradientenverfahren Algorithmus II PCG:

```
% Input: A, B, b, x_0, k_max
%    A ...spd-Matrix
%    B ...Vorkonditionierer
%    b ...rechte Seite
%    x_0 ...Startwert
%    k_max ...maximale Interationszahl
r_0 = b - Ax_0
q_1 = Br_0
for k=1:k_max
   t_k = <r_{k-1},Br_{k-1}> / <q_k,Aq_k>
   x_k = x_{k-1} + t_k q_k
   r_k = r_{k-1} - t_k Aq_k
   if |r_k| < ε return
   s_{k+1} = <r_k,Br_k> / <r_{k-1},Br_{k-1}>
   q_{k+1} = Br_k + s_{k+1} q_k
end
```

SATZ 3.52. *Für diesen Algorithmus gilt die Abschätzung:*

$$\|x - x_k\|_{AB} \leq 2\left(\frac{\sqrt{K_{2B}(AB)} - 1}{\sqrt{K_{2B}(AB)} + 1}\right)^k \|x - x_0\|_{AB}$$

Kapitel 4

Ausgleichsprobleme

4.1 Gaußsche Methode der kleinsten Fehlerquadrate

4.1.1 Problemstellung

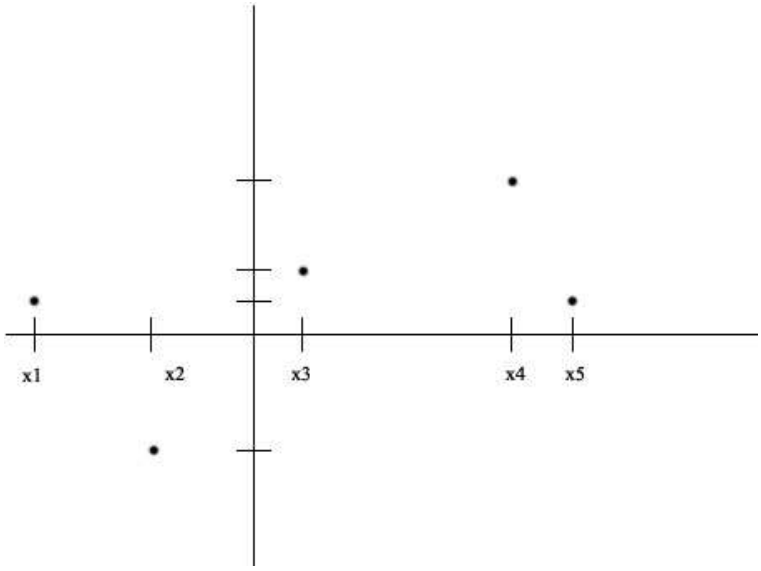
Wir gehen von folgender Problemstellung aus: Gegeben seien m Punkte

$$(x_i, f_i) \quad x_i, f_i \in \mathbb{R} \quad i = 1, \ldots, m.$$

Wir nehmen an, dass die Werte f_i von x_i abhängen und dass diese Abhängigkeit durch eine Modellfunktion φ gegeben ist mit

$$f(x) = \varphi(x; p_1, \ldots, p_n)$$

Wobei in die Modellfunktion n unbekannte Parameter p_i eingehen.



Wenn f_i Meßwerte mit Meßfehler sind, dann können wir nur fordern, dass

$$f_i \approx \varphi(x_i; p_1, \ldots, p_n) \quad i = 1, \ldots, m.$$

oder, dass die Abweichungen

$$\Delta_i = f_i - \varphi(x_i; p_1, \ldots, p_n)$$

so klein wie möglich sind.

Dies führt zu der Aufgabe, die Parameter $p_1, \ldots, p_n$ so zu bestimmen, dass

$$\Delta^2 = \sum_{i=1}^{n} \Delta_i^2 \quad i = 1, \ldots, m.$$

minimal wird.

Wir betrachten den Fall, dass die Modellfunktion φ linear in p_i ist. Das heißt

$$\varphi(x; p_1, \ldots, p_n) = a_1(x)\, p_1 + \ldots + a_n(x)\, p_n$$

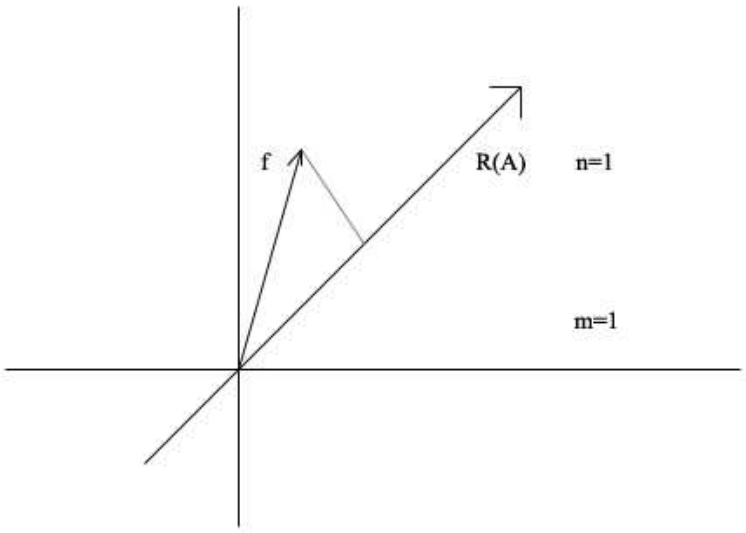
wobei $a_1, \ldots, a_n : \mathbb{R} \longrightarrow \mathbb{R}$ beliebige (gegebene) Funktionen sind.
Sei $\|\cdot\|$ die euklidische Norm. Im linearen Fall läßt sich das Ausgleichsproblem in der Kurzfassung

$$min\, \|f - Ap\| \qquad f \in \mathbb{R}^m, \quad p \in \mathbb{R}^n$$

schreiben, wobei $A = (a_{ij}) \in M(m \times n; \mathbb{R})$ mit $a_{ij} = a_j(x_i)$. Wir betrachten hier nur den überbestimmten Fall $m \geq n$, d.h. es liegen mehr Daten als Parameter vor.

4.1.2 Normalengleichung

Geometrisch gesprochen suchen wir nach einem Punkt $z = Ap$ aus dem Bildraum R von A, der den kleinsten euklidischen Abstand zu dem gegebenen Punkt f hat. Mit anderen Worten: Ax ist die orthogonale Projektion von f auf $R(A)$:

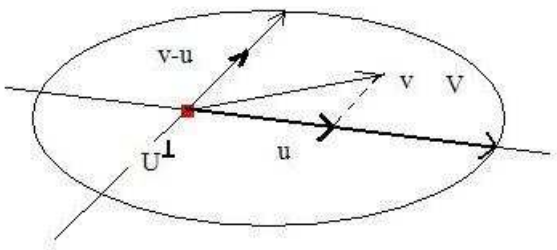


SATZ 4.0. *Sei V ein endlich dimensionaler Euklidischer Vektorraum mit Skalarprodukt $\langle\cdot,\cdot\rangle$, $U \subset V$ ein Unterraum und*

$$U^{\perp} = \{v \in V \,|\, \langle v, u\rangle = 0 \quad \forall u \in U\}$$

sein orthogonales Komplement in V. Dann gilt für alle $v \in V$ bezüglich $\|v\| = \sqrt{\langle v, v\rangle}$,dass

$$\|v - u\| = \min_{w \in U} \|v - w\| \iff v - u \in U^{\perp}$$



Beweis 4.1. $\Leftarrow$: Sei $u \in U$ der eindeutig bestimmte Punkt, so dass $v-u \in U^{\perp}$. Dann gilt $\forall w \in U$

$$\begin{aligned} \|v-w\|^2 &= \|v-u\|^2 + 2\langle v-u, u-w\rangle + \|u-w\|^2 \\ &= \|v-u\|^2 + \|u-w\|^2 \geq \|v-u\|^2 \end{aligned}$$

wobei Gleichgewicht gilt wenn $u = w$.

$\Rightarrow$: Sei

$$\|v-u\| = min_{w\in U}\|v-w\| \, d.h. \, \|v-u\| \leq \|v-w\|$$

nehmen wir $w = u + \alpha\delta \quad \delta \in U$ dann ist bei $\alpha = 0$ das Minimum von $\|v-u-\alpha\delta\|^2$. Daraus folgt

$$\begin{aligned} \frac{\partial}{\partial\alpha}\|v-u-\alpha\delta\|^2 &= \frac{\partial}{\partial\alpha}\left(\|v-u\|^2 + \alpha^2\|\delta\|^2 + 2\alpha\langle v-u,\delta\rangle\right) \\ &= 2\alpha\|\delta\|^2 + 2\langle v-u,\delta\rangle = 0 \quad \forall\delta \in U \text{ und } \alpha = 0 \\ \Longrightarrow \quad \langle v-u,\delta\rangle &= 0 \quad \forall\delta \in U \\ &\Longrightarrow \quad v-u \in U^{\perp} \end{aligned}$$

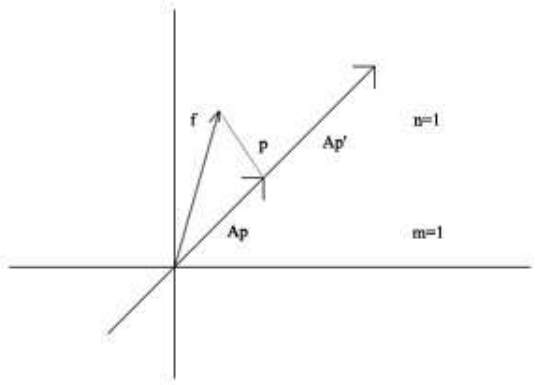
Damit ist die Lösung $u \in U$ von $\min_{w\in U}\|v-u\|$ eindeutig bestimmt und heißt orthogonale Projektion von v an U. q.e.d.

Bemerkung 4.2. Der Satz gilt auch, wenn U durch einen affinen Unterraum $W = w_0 + U \subset V$ ersetzt ist. Dieser Satz erlaubt uns eine Aussage über die Existenz und Eindeutigkeit des linearen Ausgleichsproblems.

SATZ 4.3. *Der Vektor $p \in \mathbb{R}^m$ ist genau dann Lösung des linearen Ausgleichsproblems* $\min\|f-Ap\|$, *falls er die Normalgleichung*

$$A^TAp = A^Tf$$

erfüllt. Inbesondere ist das lineare Ausgleichsproblem genau dann eindeutig *lösbar, wenn der Rang von A maximal ist, das heißt $RangA = n$.*



Beweis 4.4. Setzen wir den vorigen Satz auf $V = \mathbb{R}^m$ und $U = \mathbb{R}(A)$

$$\begin{aligned} \min\|f-Ap\| &\Leftrightarrow \langle f-Ap, Ap'\rangle = 0 \quad \forall p' \in \mathbb{R}^n \\ &\Leftrightarrow \langle A^T(f-Ap), p'\rangle = 0 \quad \forall p' \in \mathbb{R}^n \\ &\Leftrightarrow A^T(f-Ap) = 0 \\ &\Leftrightarrow A^TAp = A^Tf \end{aligned}$$

Eindeutigkeit: A^TA ist invertierbar wenn Rang $A = n$. Geometrisch gesehen, die Lösung der Normalgleichung ergibt $(f-Ap) \perp R(A)$. q.e.d.

4.1.3 Lösung der Normalgleichungen

Bemerkung 4.5. Sei das lineare Ausgleichsproblem eindeutig lösbar, das heißt Rang $A = n$. So ist A^TA eine spd-Matrix. Daher bietet sich das Cholesky - Verfahren an.
Aufwand:

1. Berechnung von $A^TA \quad \approx \frac{1}{2}n^2m \qquad \Longrightarrow$ für $m \gg n$ erhalten wir $\approx \frac{1}{2}n^2m$
2. Cholesky - Zerlegung von $A^TA \approx \frac{1}{6}n^3 \quad \Longrightarrow$ für $m \approx n$ haben wir $\approx \frac{2}{3}n^3$

Dazu kommt die Verstärkung der Fehler von A^Tb, beschrieben durch die Kondition (hier Definition für A $m \times n$-Matrix):

$$K_2\left(A^TA\right) = K_2\left(A\right)^2$$

Das heißt, falls $K_2\left(A\right) \gg 1$, kann der Übergang zu A^TA nicht hingenommen werden.
Im Fall $K_2\left(A\right) \gg 1$ kann die Gauss-Elimination zur Instabilität führen. Wählt man statt L_i , orthogonale Transformationen Q_j für die Elimination

$$K_2\left(Q_j\right) = \|Q_j\| \cdot \left\|Q_j^{-1}\right\| = \|Q_j\| \cdot \left\|Q_j^T\right\| = 1$$

so gibt ein stabiles **Orthogonalisierungsverfahren**.

4.2 Orthogonalisierungsverfahren

SATZ 4.6. *Sei $A \in M(m \times n; \mathbb{R})$, $m \geq n$, $Rang(A) = n$ und $Q \in O(m)$ orthogonale Matrix mit*

$$Q^T\ A = \begin{pmatrix} * & \dots & * \\ & \ddots & \vdots \\ & & * \\ & & \\ & 0 & \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

wobei $R \in M(n \times n; \mathbb{R})$ eine (invertierbare) obere Dreiecksmatrix ist. Sei $Q^Tf = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$
Dann ist $p = R^{-1}f_1$ die Lösung von $min\|f - Ap\|$. Man beachte, dass das Residuum $r = f - Ap$ im allgemeinen nicht verschwindet und dass $\|r\| = \|f_2\|$.

Bemerkung 4.7. Es sei zu bemerken, dass orthogonale Transformationen die Norm $\|x\|_2$ eines Vektors x unverändert lassen:$\|Qx\|_2 = \sqrt{x^TQ^TQx} = \|x\|_2$. Sei

$$Q^Tf = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

so sehen wir, dass

$$\|f - Ap\|_2^2 = \|Q^T(f - Ap)\|_2^2 = \| \begin{pmatrix} R \\ 0 \end{pmatrix} p - \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \|_2^2 = \|Rp - f_1\|_2^2 + \|f_2\|_2^2$$

Folglich ist $\|f - Ap\|_2$ minimal, wenn p so gewählt wird, dass $Rp = f_1$.
Die in Frage kommenden orthogonalen Transformationen lassen sich für $m = 2$ leicht geometrisch ableiten, nämlich als Drehungen (Rotationen) bzw. Spiegelungen (Reflexionen):
Die erste Möglichkeit ist, a um den Winkel θ auf αe_1 zu drehen. Wir erhalten

$$a \longmapsto \alpha e_1 = Qa \text{ mit } Q = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}.$$

Als zweite Möglichkeit können wir a an der auf dem Vektor v senkrecht stehenden Gerade l spiegeln, d.h.

$$a \longmapsto \alpha e_1 = a - 2\frac{\langle v, a\rangle}{\langle v, v\rangle}v$$

Diese zwei Fälle werden nun in den nächsten beiden Kapitel verallgemeinert:

4.2.1 Givens-Rotationen

Als **Givens-Rotationen** bezeichnet man Matrizen der Form

$$\Omega_{kl} := \begin{pmatrix} I & & & & \\ & c & & s & \leftarrow k \\ & & I & & \\ & -s & & c & \leftarrow l \\ & & & & I \end{pmatrix} \qquad \Omega_{kl} \in \mathbb{R}^{m\times m}$$

wobei I jeweils die Einheitsmatrix der passenden Dimension ist und $c^2 + s^2 = 1$. Geometrisch beschreibt die Matrix eine Drehung um den Winkel θ ($c = \cos\theta$ und $s = \sin\theta$) in der (k,l)-Ebene. Wenden wir Ω_{kl} auf einen Vektor $x \in \mathbb{R}^m$ an, so folgt

$$x \longmapsto y = \Omega_{kl}x \text{ mit } y_i = (\Omega_{kl}x)_i = \begin{cases} cx_k + sx_l & , \quad i = k \\ -sx_k + cx_l & , \quad i = l \\ x_i & , \quad i \neq k \ i \neq l \end{cases}$$

Man bestimmt c und s so dass zum Beispiel die Komponente x_l von x Null wird:

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} x_k \\ x_l \end{pmatrix} = \begin{pmatrix} \overline{x_k} \\ 0 \end{pmatrix} \Leftrightarrow \overline{x_k} = \pm\sqrt{x_k^2 + x_l^2},\ c = x_k/\overline{x_k} \ \ s = x_l/\overline{x_k}$$

Unsere Matrix A schreiben wir in der Form $A = [a_1, a_2, \ldots, a_n]$ wobei $a_1, \ldots, a_n \in \mathbb{R}^m$ die Spalten von A sind. Und $\Omega_{kl}A = [\Omega_{kl}a_1, \Omega_{kl}a_2, \ldots, \Omega_{kl}a_n]$.

Beispiel 4.8.

$$A = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \overset{\Omega_{4,3}}{\to} \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \\ 0 & * & * \end{pmatrix} \overset{\Omega_{3,2}}{\to} \begin{pmatrix} * & * & * \\ * & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix} \overset{\Omega_{2,1}}{\to} \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$$

$$\overset{\Omega'_{4,3}}{\to} \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & * & * \\ 0 & 0 & * \end{pmatrix} \cdots \overset{\Omega''_{4,3}}{\to} \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & * \\ 0 & 0 & 0 \end{pmatrix}$$

Der Aufwand für die QR-Zerlegung einer vollbestzten Ausgangsmatrix $A \in M(m \times n; \mathbb{R})$ beträgt $\sim 4n^3/3$. Gegenüber der Alternative der Gaußschen Dreieckszerlegung ist dies ein 4 mal höherer Aufwand, jedoch erreichen wir eine höhere Stabilität.

Beispiel 4.9. *Givens-Rotation:*

$$A_1 = A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad f = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\Omega_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{pmatrix} \qquad x_2 = 0 \quad x_3 = 1 \quad \overline{x} = 1 \quad c = 0 \quad s = 1$$

$$\Omega_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \qquad A_2 = \Omega_{23}A = \begin{pmatrix} 1 & 2 \\ 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\Omega_{12} = \begin{pmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad x_1 = 1 \quad x_2 = 1 \quad \overline{x} = \sqrt{2} \quad c = 1/\sqrt{2} \quad s = 1/\sqrt{2}$$

$$\Omega_{12} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad A_3 = \Omega_{12} A_2 = \begin{pmatrix} \sqrt{2} & \sqrt{2} \\ 0 & -\sqrt{2} \\ 0 & -1 \end{pmatrix}$$

$$\Omega'_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{pmatrix} \quad x_2 = -\sqrt{2} \quad x_3 = -1 \quad \overline{x} = \sqrt{3} \quad c = -\sqrt{2/3} \quad s = -1/\sqrt{3}$$

$$\Omega_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sqrt{2/3} & -1/\sqrt{3} \\ 0 & 1/\sqrt{3} & -\sqrt{2/3} \end{pmatrix} \quad A_2 = \Omega_{23} A = \begin{pmatrix} 1 & 2 \\ 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$A_4 = \Omega'_{23} A_3 = \begin{pmatrix} \sqrt{2} & \sqrt{2} \\ 0 & \sqrt{3} \\ 0 & 0 \end{pmatrix} = Q^T A = R$$

4.2.2 Householder-Reflexionen

Für **Householder-Reflexionen** betrachtet man die Matrizen $Q \in M(n \times n; \mathbb{R})$ der Form

$$Q = I - 2\frac{vv^T}{\langle v, v \rangle}$$

mit $v \in \mathbb{R}^n$. Solche Matrizen beschreiben die Reflexion an der auf v senkrecht stehenden Ebene. Wenden wir Q auf einen Vektor $y \in \mathbb{R}^n$ an, so gilt

$$y \longmapsto Qy = (I - 2\frac{vv^T}{v^T v})y = y - 2\frac{\langle v, y \rangle}{\langle v, v \rangle} v$$

Soll y auf ein Vielfaches αe_1 des ersten Einheitsvektors e_1 abgebildet werden, d.h. so wählt man

$$v = (y_1 - \alpha, y_2, \ldots, y_n)^T \quad \alpha = \pm \|y\|^2$$

(usw. auf $e_j \quad v = (y_1, y_2, \ldots, y_j - \alpha, \ldots, y_n)^T$) Sei dann $y \equiv a_1$ erste Spalte von A, so können wir die Householder-Reflexionen anwenden und erhalten

$$A \longrightarrow A' := Q_1 A = \begin{pmatrix} \alpha_1 & & & \\ 0 & & & \\ \vdots & a'_2 & \ldots & a'_n \\ 0 & & & \end{pmatrix}$$

wobei

$$Q_1 = I - 2\frac{v_1 v_1^T}{v_1^T v_1} \quad v_1 = a_1 - \alpha_1 e_1 \quad \alpha_1 = -sgn(a_{11}) \|a_1\|_2$$

Nach dem k-ten Schritt haben wir die Ausgangsmatrix A bis auf eine Restmatrix $T^{k+1} \in M(m-k \times n-k; \mathbb{R})$ auf obere Dreiecksgestalt gebracht

$$A^{(k)} = \begin{pmatrix} * & \dots & \dots & \dots & * \\ & \ddots & & & \vdots \\ & & * & \dots & * \\ & & 0 & & \\ & & \vdots & T^{k+1} & \\ & & 0 & & \end{pmatrix}$$

Bilden wir nun die orthogonale Matrix

$$Q_{k+1} = \begin{pmatrix} I_k & 0 \\ 0 & \overline{Q}_{k+1} \end{pmatrix}$$

wobei $\overline{Q}_{k+1} \in O(m-k)$ wie im ersten Schritt mit $T^{(k+1)}$ anstelle von A konstruiert wird, so können wir die nächste Subspalte unterhalb der Diagonalen eliminieren. Insgesamt erhalten wir so nach $p = min(m-1, n)$ Schritten die obere Dreiecksmatrix

$$R = Q_p \dots Q_1 A.$$

Für den Aufwand erhalten wir bei dieser Methode falls $m \approx n$ $\frac{2}{3}n^3$ Multiplikationen, falls $m \gg n$ $2n^2m$ Multiplikationen.

4.3 SVD - singuläre Wertzerlegung

Gegeben seien $A \in \mathbb{R}^{m \times n}$, $A^TA \in \mathbb{R}^{n \times n}$ ist symmetrisch und positiv semi-definit. Somit hat A^TA orthogonalen Eigenvektoren und alle Eigenwerte von A^TA sind nicht negativ. Für die positiven Eigenwerte von A^TA schreiben wir $\sigma_1^2 \geq \sigma_2^2 \geq \dots \geq \sigma_r^2 > 0$.

Lemma 4.10. Die beiden Matrizen A^TA und AA^T haben dieselben positiven Eigenwerte. Bezeichnet man mit $u_j \in \mathbb{R}^n$ den normalisierten Eigenvektor von A^TA mit dem Eigenwert σ_j^2, dann ist $Au_j \in \mathbb{R}^m$ ein Eigenvektor von AA^T dessen Eigenwert σ_j^2 entspricht. Wählen wir $v_j = \frac{1}{\sigma_j}Au_j$ dann ist v_j normalisiert.

Beweis 4.11. Ist $A^TAu_j = \sigma_j^2 u_j$ dann ist $(AA^T)Au_j = \sigma_j^2 Au_j$ u_j normalisiert, $u_j^T A^T Au_j = \|Au_j\|_2^2\ \sigma_j^2$. q.e.d.

Definition 4.12. Die Werte $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma \geq \sigma_{r+1} = \dots = 0$ nennt man singuläre Werte von A. Und
$\{u_j\}_j$ ... rechts singuläre Vektoren von A
$\{v_j\}_j$ ... links singuläre Vektoren von A.

Lemma 4.13. Sei $U = [u_1 \dots u_n] \in \mathbb{R}^{n \times n}$ und $V = [v_1 \dots v_m] \in \mathbb{R}^{m \times m}$ gilt:

1. $v_j = \frac{1}{\sigma_j}Au_j \quad j = 1, \dots, r$
2. $\{v_j\}_{j=r+1}^m$ sind orthonormale Eigenvektoren bezüglich des Null-Eigenwerts von AA^T

Dann sind U und V orthogonale Matrizen.

SATZ 4.14 (SVD)**.** *Sind U und V gegeben wie im vorherigen Lemma, haben wir*

$$A = V \begin{bmatrix} \sum & 0 \\ 0 & 0 \end{bmatrix} U^T$$

*und $\sum = diag\{\sigma_1 \dots \sigma_r\}$ Diese Darstellung wird **singuläre Wert Zerlegung** (SVD) von A genannt.*

Beweis 4.15. sei

$$U = \begin{bmatrix} U_1 & U_2 \end{bmatrix} V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$$

$$V^T AU = \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} A \begin{bmatrix} U_1 & U_2 \end{bmatrix} = \begin{bmatrix} V_1^T AU_1 & V_1^T AU_2 \\ V_2^T AU_1 & V_2^T AU_2 \end{bmatrix}$$

hier ist nun $AU_2 = 0, \quad V_2^T AU_2 = V_2^T V_1 \sum = 0, \quad V_1^T AU_1 = \sum$ mit gewähltem V. q.e.d.

SATZ 4.16. *Sei $A \in \mathbb{R}^{m \times n}$, dargestellt in SVD Form (oben). Dann*

$$\widetilde{x} = U \begin{bmatrix} \sum^{-1} & 0 \\ 0 & 0 \end{bmatrix} V^T b$$

minimiert $\|b - Ax\|_2$ über alle $x \in \mathbb{R}^n$. Ferner, wenn $\widehat{x}$ eine weitere Lösung des Ausgleichsproblems für ein Minimum ist, so gilt: $\|\widetilde{x}\|_2 \leq \|\widehat{x}\|$.

Beweis 4.17.

$$\|b - Ax\|_2^2 = \left\|V^T (b - Ax)\right\|_2^2 = \left\|V^T AUU^T x - V^T b\right\|_2^2 = \left\| \begin{bmatrix} \sum & 0 \\ 0 & 0 \end{bmatrix} z - c \right\|_2^2$$

mit $z = U^T x \quad c = V^T b. \quad z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ und $c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ für $z_1, c_1 \in \mathbb{R}^r$. Dann

$$\|b - Ax\|_2^2 = \left\| \begin{bmatrix} \sum z_1 - c_1 \\ -c_2 \end{bmatrix} \right\|_2^2 = \left\| \sum z_1 - c_1 \right\|_2^2 + \|c_2\|_2^2$$

Dann ist $\|b - Ax\|_2^2$ minimiert wenn $z_1 = \sum^{-1} c_1$ (z_2 kann beliebig sein).

Wähle $\widetilde{z} = \begin{bmatrix} \sum^{-1} c_1 \\ 0 \end{bmatrix}$ dann folgt

$$\widetilde{x} = U\widetilde{z} = U \begin{bmatrix} \sum^{-1} c_1 \\ 0 \end{bmatrix} = U \begin{bmatrix} \sum^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = U \begin{bmatrix} \sum^{-1} & 0 \\ 0 & 0 \end{bmatrix} V^T b$$

Für jede andere Lösung des Ausgleichsproblems $\widehat{x}$ muß das entsprechende $\widehat{z}$ die Form $\widehat{z} = \begin{bmatrix} \sum^{-1} c_1 \\ z_2 \end{bmatrix}$ haben. Dann erhalten wir

$$\|\widehat{x}\|_2^2 = \|U\widehat{z}\|_2^2 = \|\widehat{z}\|_2^2 = \left\|\sum^{-1} c_1\right\|_2^2 + \|z_2\|_2^2 \geq \|\widetilde{z}\|_2^2 = \|\widetilde{x}\|_2^2$$

q.e.d.

4.4 Nichtlineare Ausgleichsprobleme

Definiert man die Abbildung

$$F : \mathbb{R}^n \to \mathbb{R}^m, \qquad F_i(p) = \varphi(x_i; p) - f_i, \quad i = 1, \ldots, m$$

kann das nichtlineare Ausgleichsproblem wie folgt formuliert werden.
Bestimme $p^* \in \mathbb{R}^n$, so dass

$$\|F(p^*)\|_2 = \min_{p \in \mathbb{R}^n} \|F(p)\|_2$$

oder äquivalent

$$\phi(p^*) = \min_{p \in \mathbb{R}^n} \phi(p)$$

wobei $\phi : \mathbb{R}^n \to \mathbb{R}$, $\phi(p) = \|F(p)\|_2^2/2 = F(p)^T F(p)/2$.
Die Funktion $\phi(p)$ hat in einem Punkt p^* ein lokales Minimum genau dann, wenn die folgenden zwei Bedingungen erfüllt sind.

$$\nabla\phi(p^*) = 0 \text{ und } \nabla^2\phi(p^*) \in \mathbb{R}^{n\times n} \text{ ist spd.}$$

Wir haben dass

$$\nabla\phi() = F'(p)^T\, F(p) \text{ und } \nabla^2\phi(p) = F'(p)^T\, F'(p) + \sum_{i=1}^{m} F_i(p)^T\, F_i''(p).$$

Durch Taylor Entwicklung $F(p) = F(p^k) + F'(p^k)(p - p^k) + \mathcal{O}(\|p - p^k\|_2^2)$ erhalten wir eine Folge von lineare Ausgleichsprobleme
Finde $s^k \in \mathbb{R}^n$ mit minimaler 2-Norm, so dass

$$\|F'(p^k)\, s^k + F(p^k)\|_2 = \min_{s\in\mathbb{R}^n} \|F'(p^k)\, s + F(p^k)\|$$

setze $p^{k+1} = p^k + s^k$. Wenn $Rang(F'(p)) = n$ ist die Bedingung mmit minimaler 2-Normnnicht notwendig. Insgesamt erhalten wir folgendes
Gauß–Newton Algorithmus:

```
% Input: Startwert p^0
Für k=1:k_max
   Berechne F(p^k), F'(p^k)
   Löse das lineare Ausgleichsproblem min_{s∈R^n} ||F'(p^k) s + F(p^k)||
   Setze p^{k+1} = p^k + s^k
   falls ||F'(p^{k+1})F(p^{k+1})||_2 ≤ ε dann stopp
```

Sei p^* ein kritischer Punkt von ϕ, der in einer Umgebung U eindeutig ist und sei $Rang(F'(p)) = n$ für alle $p \in U$. Dann hat das lineare Ausgleichsproblem die eindeutige Lösung

$$s^k = -\left[F'(p^k)^T\, F'(p^k)\right]^{-1} F'(p^k)^T\, F(p^k)$$

Deshalb gilt für die Gauß–Newton Iteration

$$p^{k+1} = p^k - \left[F'(p^k)^T\, F'(p^k)\right]^{-1} F'(p^k)^T\, F(p^k).$$

Wir definieren $\Phi(p^k) = p^k - \left[F'(p^k)^T\, F'(p^k)\right]^{-1} F'(p^k)^T\, F(p^k)$. Für die Gauß–Newton Iteration gilt Falls die Gauß-Newton-Methode konvergiert, ist die Konvergenz im allgemeinen nicht schneller als linear. Lokale Konvergenz des Verfahrens ist gesichert, falls das Residuum $\|F(p^*)\|$ hinreichend klein ist.
Wenn der kritische Punkt p^* ein lokales Maximum oder ein Sattelpunkt ist, ist solcher Punkt für das Gauß–Newton-Verfahren abstoßend. Das Verfahren bewahrt uns also davor, einen falschen kritischen Punkt zu finden.
Beispiel (x statt p):
Sei

$$F(x) = \begin{pmatrix} a + r\cos x \\ r\sin x \end{pmatrix}$$

mit $a > r > 0$, $x \in [0, 2\pi]$. Wir haben $\|F(x)\|_2 = \sqrt{a^2 + 2ar\cos x + r^2}$, $F'(x) = \mathrm{r} \begin{pmatrix} -\sin x \\ \cos x \end{pmatrix}$, $F'(x)^T F'(x) = r^2$, $\nabla\varphi(x) = -ra\sin x$. Es gibt zwei kritische Punkte von φ: $x^* = 0$ (lokales Maximum), $x^* = \pi$ (lokales Minimum). Die Iterationsfunktion zu F ist $\Phi(x) = x + \frac{a}{r}\sin x$. In den

kritischen Punkten gilt $|\Phi'(x^*)| = |1 + \frac{a}{r}\cos x^*|$. Für $x^* = 0$ gilt $|\Phi'(x^*)| > 1$ und für $x^* = \pi$ gilt $|\Phi'(x^*)| = \frac{a}{r} - 1$.
Bemerkungen:
Die Gauß-Newton-Methode hat in diesem Beispiel folgende Eigenschaften: 1. Das lokale Maximum ist abstosend. 2. Die Methode ist linear konvergent in einer Umgebung des lokalen Minimums (wenn $a < 2r$), oder 3. das lokale Minimum ist auch abstosend (wenn $a > 2r$). Man kann zeigen, das ähnliche Eigenschaften in einem allgemeinen Rahmen güultig sind.
Jetzt besprechen wir das Levenberg-Marquardt Verfahren als weiter Entwicklung des Gauß-Newton-Methode. Man betrachte das Problem:
Finde $s^k \in \mathbb{R}^n$, so dass

$$\|F'(p^k)\, s^k + F(p^k)\|_2^2 + \mu^2 \|s^k\|_2^2 = \min_{s\in\mathbb{R}^n} \|F'(p^k)\, s + F(p^k)\|_2^2 + \mu^2 \|s\|_2^2$$

wobei $\mu > 0$ ein Parameter ist. Dann folgt $p^{k+1} = p^k + s^k$.
Eine äquivalente Formulierung ist folgende: Finde s^k, so dass folgender Norm minimiert wird

$$\| \begin{pmatrix} F'(p^k) \\ \mu\, I \end{pmatrix} s^k + \begin{pmatrix} F(p^k) \\ 0 \end{pmatrix} \|_2^2$$

So haben wir der Vorteil dass die Matrix $\begin{pmatrix} F'(p^k) \\ \mu\, I \end{pmatrix}$ immer vollen Rang hat.
Für die Korrektur s^k gilt $\|s^k\|_2 \le \|F(p^k)\|_2/\mu$. Also kann man durch eine geeignete Wahl von μ die Korrektur kontrollieren. Auch das Levenberg-Marquardt-Verfahren kann man als Fixpunktiteration formulieren, wobei

$$p^{k+1} = p^k - \left[F'(p^k)^T\, F'(p^k) + \mu^2\, I\right]^{-1} F'(p^k)^T\, F(p^k).$$

Levenberg-Marquardt Algorithmus:

```
% Input: Startwert p^0
Für k=1:k_max
   Berechne F(p^k), F'(p^k)
   Löse das lineare Ausgleichsproblem min_{s∈R^n} ‖ (F'(p^k); μ I) s^k + (F(p^k); 0) ‖_2^2
   Setze p^{k+1} = p^k + s^k
   falls ‖F'(p^{k+1})F(p^{k+1})‖_2 ≤ ε dann stopp
```

Kapitel 5

Lineare Eigenwertprobleme

Wir diskutieren die numerische Lösung des Eigenwertproblems

$$A\underline{x} = \lambda \underline{x}$$

wobei A eine $n \times n$ - Matrix ist, und $\underline{x}$ einen **Eigenvektor** zum **Eigenwert** $\lambda \in \mathbb{C}$ bezeichnet. Für eine beliebige komplexe Matrix $A \in M(n \times n; \mathbb{C})$ sind die Eigenwerte $\lambda_1, \ldots, \lambda_n$ die Nullstellen des charakteristischen Polynoms

$$P_A(\lambda) = det(A - \lambda I).$$

Praktisch ist die Konstruktion von $P_A(\lambda)$ jedoch zu aufwändig ($\approx n!$) und deshalb suchen wir Lösungsverfahren mit geringerer Komplexität.

SATZ 5.0. *Seien $\lambda_1, \ldots, \lambda_k$ mit $k \leq n$ unterschiedliche Eigenwerte der Matrix A mit Eigenvektoren $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_k$. Dann sind $\underline{x}_1, \ldots, \underline{x}_k$ linear unabhängig.*

Definition 5.1. Die Vektoren $\underline{x}_1, \ldots, \underline{x}_n$ sind **orthogonal** wenn $\underline{x}_i^T \underline{x}_j = 0$ für $i \neq j$. Man nennt sie **orthonormal** falls $\underline{x}_i^T \underline{x}_j = \delta_{ij}$. Dann ist die Matrix $P = \begin{pmatrix} \underline{x}_1 & \underline{x}_2 & \ldots & \underline{x}_n \end{pmatrix}$ orthogonal: $P^{-1} = P^T$.

SATZ 5.2. *Sei A eine symmetrische $n \times n$ - Matrix und $D = diag\{\lambda_1, \ldots, \lambda_n\}$ mit $\lambda_i \in \mathbb{R}$. Dann existieren n Eigenvektoren von A, die orthonormal sind, und es gilt:*

$$D = P^{-1}AP = P^T AP$$

Falls A eine spd-Matrix ist, dann gilt $\lambda_i > 0$ für $i = 1, \ldots, n$.

SATZ 5.3 (Satz von Gerschgorin)**.** *Sei $A \in M(n \times n; \mathbb{C})$ und seien R_i die Kreise auf $\mathbb{C}$ mit Zentrum a_{ii} und Radius $\sum_{j=1, j\neq i}^{n} |a_{ij}|$, d.h.*

$$R_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}|\} \quad \text{für } i = 1, \ldots, n.$$

Dann liegen alle Eigenwerte λ_i von A in $R = \bigcup_{i=1}^{n} R_i$.

Beweis 5.4. Sei λ ein Eigenwert von A und $\underline{x} = (x_1, \ldots, x_n)$ der zugehörige Eigenvektor. Wählen wir $i = 1, \ldots, n$ so, dass $|x_i| = \|\underline{x}\|_\infty$. Für die i-te Gleichung gilt dann

$$\begin{aligned} a_{i1}x_1 + \ldots + a_{ii}x_i + \ldots + a_{in}x_n &= \lambda x_i \\ (a_{ii} - \lambda) x_i &= -\sum_{j=1, j\neq i}^{n} a_{ij}x_j \end{aligned}$$

Dann folgt

$$|a_{ii} - \lambda| \leq \sum_{j=1, j\neq i}^{n} |a_{ij}| \frac{|x_j|}{|x_i|} \leq \sum_{j=1, j\neq i}^{n} |a_{ij}|$$

und daraus folgt $\lambda \in R_i$.

q.e.d.

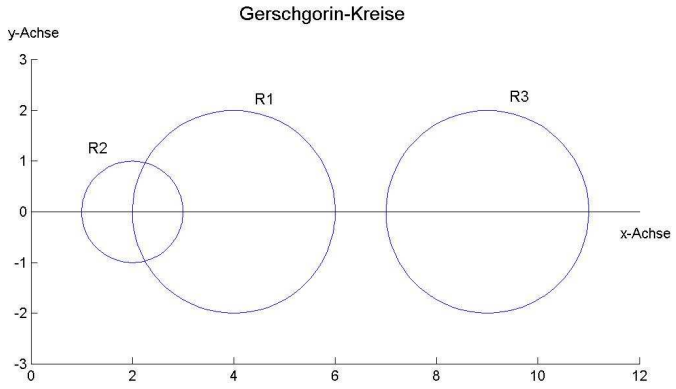
Bemerkung 5.5. Die Vereinigung von k Kreisen R_i, die nicht die restlichen Kreise schneidet, enthält genau k Eigenwerte.

Beispiel 5.6.

$$A = \begin{pmatrix} 4 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 0 & 9 \end{pmatrix}$$

$$R_1 = \{z \in \mathbb{C} : |z-4| \leq 2\}$$
$$R_2 = \{z \in \mathbb{C} : |z-2| \leq 1\}$$
$$R_3 = \{z \in \mathbb{C} : |z-9| \leq 2\}$$

Zwei Eigenwerte liegen in der Vereinigung $R_1 \cup R_2$, der dritte liegt in R_3.



5.1 Iterationen zur Berechnung von Eigenwerten

5.1.1 Direkte Vektoriteration (Power method)

Diese Iteration liefert den betragsgrößten einfachen Eigenwert von A.

SATZ 5.7. *Sei λ_1 ein einfacher Eigenwert der symmetrischen Matrix $A \in M(n \times n; \mathbb{R})$ und λ_1 sei betragsmäßig echt größer als alle anderen Eigenwerte von A, d.h.*

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n| .$$

Sei $\underline{x}_0 \in \mathbb{R}^n$ ein Vektor, der nicht senkrecht auf den Eigenraum von λ_1 steht.
Dann konvergiert die Folge

$$\underline{y}_k := \frac{\underline{x}_k}{\|\underline{x}_k\|} \quad \text{mit} \quad \underline{x}_{k+1} = A\underline{x}_k$$

gegen einen normierten [1] Eigenvektor von A zum Eigenwert λ_1.

Beweis 5.8. Sei $\underline{\eta}_1, \ldots, \underline{\eta}_n$ eine Orthonormalbasis von Eigenvektoren von A mit $A\underline{\eta}_i = \lambda_i \underline{\eta}_i$.
Dann gilt $\underline{x}_0 = \sum_{i=1}^n \alpha_i \underline{\eta}_i$ mit $\alpha_1 = \left\langle \underline{x}_0, \underline{\eta}_1 \right\rangle \neq 0$.
Folglich ist

$$\underline{x}_k = A^k \underline{x}_0 = \sum_{i=1}^n \alpha_i \lambda_i^k \underline{\eta}_i = \alpha_1 \lambda_1^k \underbrace{\left(\underline{\eta}_1 + \sum_{i=2}^n \frac{\alpha_i}{\alpha_1} \left(\frac{\lambda_i}{\lambda_1} \right)^k \underline{\eta}_i \right)}_{=:\underline{z}_k}$$

[1] normiert bezüglich $\|\cdot\|_\infty$ oder $\|\cdot\|_2$

Da $|\lambda_i| < |\lambda_1|$ für alle $i = 2, \ldots, n$, gilt $lim_{k\to\infty}\underline{z}_k = \underline{\eta}_1$ und daher

$$\underline{y}_k = \frac{\underline{x}_k}{\|\underline{x}_k\|} = \frac{\underline{z}_k}{\|\underline{z}_k\|} \longrightarrow \underline{\eta}_1, \quad \text{für } k \to \infty$$

q.e.d.

Algorithmus für die Direkte Vektoriteration:

```
% Input: A, x_0, k_max
%    A ...symmetrische-Matrix
%    x_0 ...Startwert
%    k_max ...maximale Iterationszahl
for k = 1 : k_max
   x_k = A x_{k-1}
   λ_k = (x_k)_i    so dass |(x_k)_i| = ||x_k||_∞
   x_k = x_k / ||x_k||_∞
end
```

Man kann für die Berechnung des Eigenwertes auch den sogenannten Rayleigh Quotienten verwenden: Dann liefert die Vektoriteration nach k Schritten einen approximierten Eigenvektor x_k und einen approximierten Eigenwert

$$\mu(x_k) = \frac{\langle x_k, Ax_k\rangle}{\langle x_k, x_k\rangle}.$$

Die Funktion $\mu : \mathbb{R}^n \longrightarrow \mathbb{R}$ heißt **Rayleigh-Quotient** von A.

Beispiel 5.9.

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

Die Matrix A hat die Eigenwerte $\lambda_1 = 6, \lambda_2 = 3$ *und* $\lambda_3 = 1$.
Wählen wir als Startwert $x_0 = (1, 0, 0)$, *so erhalten wir:*

k	x_k	y_k	$\mu(x_k)$
0	*(1,0,0)*	*(1,0,0)*	*4*
1	*(4,-1,1)*	*(0.94,-0.23,0.23)*	*4*
2	*(4.24,-2.12,2.12)*	*(0.81,-0.40,0.40)*	*5*
$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	*(3.47,-3.46,3.46)*	*(0.57,-0.57,0.57)*	*5.99*

Hier wurde für $y_k = \frac{x_k}{\|x_k\|}$ *die euklidische Norm* $\|\cdot\|_2$ *verwendet.*

Nachteile der direkten Vektoriteration:

- Wir erhalten nur den Eigenvektor zum betragsgrößten Eigenwert λ_1 von A.
- Die Konvergenzgeschwindigkeit hängt von $\rho = \left|\frac{\lambda_2}{\lambda_1}\right|$ ab. Liegen die Eigenwerte λ_1 und λ_2 betragsmäßig dicht zusammen, so konvergiert die direkte Vektoriteration nur sehr langsam.

Diese Nachteile werden bei der von WIELANDT (1945) entwickelten inversen Vektoriteration vermieden.

5.1.2 Inverse Vektoriteration (Inverse power method)

Angenommen wir hätten einen Schätzwert $\bar{\lambda} \approx \lambda_i$ eines beliebigen Eigenwertes λ_i der Matrix A zur Verfügung, so dass

$$\left|\lambda_i - \bar{\lambda}\right| < \left|\lambda_j - \bar{\lambda}\right|, \quad j \neq i.$$

Dann ist $\left(\lambda_i - \bar{\lambda}\right)^{-1}$ der betragsgrößte Eigenwert der Matrix $\left(A - \bar{\lambda}I\right)^{-1}$. Konsequenterweise konstruiert man deshalb die Vektoriteration für diese Matrix:

$$\underline{x}_{k+1} = \left(A - \bar{\lambda}I\right)^{-1} \underline{x}_k$$

Daraus folgt:

$$\left(A - \bar{\lambda}I\right) x_{k+1} = x_k, \quad \text{für } k = 0, 1, \ldots$$

Bei jedem Iterationsschritt muss dieses lineare Gleichungssystem gelöst werden, jedoch nur für verschiedene rechte Seiten x_k. Die Matrix $A - \bar{\lambda}I$ muss daher nur einmal zerlegt werden. Nach obigem Satz konvergiert die Folge $y_k := \frac{x_k}{\|x_k\|}$ gegen einen normierten Eigenvektor von A zum Eigenwert λ_i. Der Konvergenzfaktor ist dabei $max_{j \neq i} \left|\frac{\lambda_i - \bar{\lambda}}{\lambda_j - \bar{\lambda}}\right| < 1$. Der Satz von Gerschgorin hilft bei der Schätzung von λ_i. Algorithmus für die Inverse Vektoriteration:

```
% Input: A, x_0, k_max, λ̄
%    A ... symmetrische-Matrix
%    x_0 ...Startwert
%    k_max ...maximale Iterationszahl
%    λ̄ ... Schätzwert des Eigenwertes
A − λ̄I = LR        % LR-Zerlegung
for k = 1 : k_max
   LR · x_k = x_{k−1}      % löse das Gleichungssytem
   λ_k = λ̄ + 1/(x_k)_i     so dass |(x_k)_i| = ‖x_k‖_∞
   x_k = x_k / ‖x_k‖_∞
end
```

5.1.3 Deflation

Sei der betragsmäßig größte Eigenwert bereits berechnet worden. Mit der Deflation wollen wir die übrigen Eigenwerte berechnen.

SATZ 5.10. *Seien $\lambda_1, \ldots, \lambda_n$ Eigenwerte von A (wobei λ_1 einfacher Eigenwert) und seien $\underline{x}_1, \ldots, \underline{x}_n$ die zugehörigen Eigenvektoren. Sei $\underline{y} \in \mathbb{R}^n$ so dass $\underline{y}^T \underline{x}_1 = 1$. Dann hat die Matrix*

$$B = A - \lambda_1 \underline{x}_1 \underline{y}^T$$

die Eigenwerte $0, \lambda_2, \lambda_3, \ldots, \lambda_n$ mit Eigenvektoren $\underline{x}_1, \underline{w}_2, \ldots, \underline{w}_n$, wobei

$$\underline{x}_i = (\lambda_i - \lambda_1)\, \underline{w}_i + \lambda_1 \left(\underline{y}^T \underline{w}_i\right) \underline{x}_1, \quad \textit{für } i = 2, 3, \ldots, n.$$

Deflation von Wielandt:

$$\underline{y} = \frac{1}{\lambda_1 \left(\underline{x}_1\right)_i} \begin{pmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{pmatrix}, \quad \left(\underline{x}_1\right)_i \neq 0$$

Hier ist $(a_{i1}, a_{i2}, \ldots, a_{in})$ die i-te Zeile von A und $\left(\underline{x}_1\right)_i$ die i-te Komponente des Vektors $\underline{x}_1$. Für dieses $\underline{y}$ gilt:

$$\underline{y}^T \underline{x}_1 = \frac{1}{\lambda_1 \left(\underline{x}_1\right)_i} \sum_{j=1}^{n} a_{1j} \left(\underline{x}_1\right)_j = \frac{1}{\lambda_1 \left(\underline{x}_1\right)_i} \lambda_1 \left(\underline{x}_1\right)_i = 1$$

Die i-te Zeile von B ist eine Nullzeile, denn aus $B = A - \lambda_1 \underline{x}_1 \underline{y}^T$ ergibt sich:

$$\begin{aligned}(b_{i1}, b_{i2}, \ldots, b_{in}) &= (a_{i1}, a_{i2}, \ldots, a_{in}) - \lambda_1 (x_1)_i \frac{1}{\lambda_1 (x_1)_i} (a_{i1}, a_{i2}, \ldots, a_{in}) \\ &= (0, 0, \ldots, 0)\end{aligned}$$

In $Bw = \lambda w$ spielt daher auch die i-te Spalte keine Rolle. Wir erhalten eine $(n-1)\times(n-1)$-Matrix B' indem wir die i-te Zeile und die i-te Spalte von B streichen. Die direkte Vektoriteration kann nun auf B' angewendet werden, wenn $|\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$.

Beispiel 5.11.

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

Die Eigenwerte von A sind $\lambda_1 = 6, \lambda_2 = 3$, *und* $\lambda_3 = 1$. *Angenommen es wären bereits* $\lambda_1 = 6$ *und der zugehörige Eigenvektor* $\underline{x}_1 = (1, -1, 1)^T$ *bekannt.*
Wir führen die Deflation für λ_2 *durch, mit* $i = 1$*: Zunächst berechnen wir den Vektor* $\underline{y}$*:*

$$\underline{y} = \frac{1}{6 \cdot (\underline{x}_1)_1} \begin{pmatrix} 4 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ -\frac{1}{6} \\ \frac{1}{6} \end{pmatrix}$$

Um die Matrix B bestimmen zu können, benötigen wird das dualische Produkt $\underline{x}_1 \underline{y}^T$*:*

$$\underline{x}_1 \underline{y}^T = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} \frac{2}{3} & -\frac{1}{6} & \frac{1}{6} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & -\frac{1}{6} & \frac{1}{6} \\ -\frac{2}{3} & \frac{1}{6} & -\frac{1}{6} \\ \frac{2}{3} & -\frac{1}{6} & \frac{1}{6} \end{pmatrix}$$

Also erhalten wir:

$$B = A - 6 \cdot \left(\underline{x}_1 \underline{y}^T\right) = \begin{pmatrix} 0 & 0 & 0 \\ 3 & 2 & -1 \\ -3 & -1 & 2 \end{pmatrix}$$

Aus B lesen wir die Matrix B' *ab, indem wir die i-te Zeile und Spalte weglassen:*

$$B' = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

Nun erhalten wir die Eigenwerte λ_2 *und* λ_3 *indem wir die Eigenwerte der Matrix* B' *berechnen:*

$$\begin{aligned}(2-\lambda)^2 - 1 &= 0 \\ \pm(2-\lambda) &= 1 \\ \lambda_2 &= 3 \\ \lambda_3 &= 1\end{aligned}$$

Wir berechnen die zugehörigen Eigenvektoren:

$$\begin{aligned}\left(B' - \lambda_2 I\right) w_2' &= 0 \\ \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix} w_2' &= 0 \\ w_2' = \begin{pmatrix} 1 \\ -1 \end{pmatrix}\end{aligned}$$

Ebenso:

$$\begin{aligned}\left(B' - \lambda_3 I\right) w_3' &= 0\\ \begin{pmatrix} 1 & -1\\ -1 & 1\end{pmatrix} w_3' &= 0\\ w_3' &= \begin{pmatrix}1\\1\end{pmatrix}\end{aligned}$$

Daraus erhalten wir die Eigenvektoren w_2 und w_3 von B:

$$w_2 = \begin{pmatrix}0\\1\\-1\end{pmatrix}, \quad w_3 = \begin{pmatrix}0\\1\\1\end{pmatrix}$$

Schließlich können wir die Eigenvektoren x_2 und x_3 der Matrix A bestimmen:

$$\begin{aligned}\underline{x}_2 &= (\lambda_2 - \lambda_1)\,\underline{w}_2 + \lambda_1 \left(\underline{y}^T \underline{w}_2\right) \underline{x}_1\\ &= (3-6)\begin{pmatrix}0\\1\\-1\end{pmatrix} + 6 \cdot \left(-\frac{2}{6}\right)\begin{pmatrix}1\\-1\\1\end{pmatrix} = \begin{pmatrix}-2\\-1\\1\end{pmatrix}\\ \underline{x}_3 &= (\lambda_3 - \lambda_1)\,\underline{w}_3 + \lambda_1 \left(\underline{y}^T \underline{w}_3\right) \underline{x}_1\\ &= (1-6)\begin{pmatrix}0\\1\\1\end{pmatrix} + 6 \cdot (0)\begin{pmatrix}1\\-1\\1\end{pmatrix} = \begin{pmatrix}0\\-5\\-5\end{pmatrix}\end{aligned}$$

5.2 QR-Algorithmus für symmetrische Eigenwertprobleme

In diesem Abschnitt interessieren wir uns nun für die Frage, wie wir effektiv sämtliche Eigenwerte einer reellen symmetrischen Matrix $A \in M\left(n \times n, \mathbb{R}\right)$ gleichzeitig berechnen können. Wir wissen, dass A nur reelle Eigenwerte $\lambda_1, ..., \lambda_n \in \mathbb{R}$ besitzt und eine Orthonormalbasis $\eta_1, ..., \eta_n \in \mathbb{R}$ aus Eigenvektoren $A\eta_i = \lambda_i \eta_i$ existiert, d.h.

$$Q^T A Q = \Lambda = diag\left\{\lambda_1, ..., \lambda_n\right\} \text{ mit } Q = [\eta_1, ..., \eta_n] \in O(n).$$

Um die Eigenwerte λ_i zu berechnen, wird der so genannte **QR-Algorithmus** angewandt. Dieser Algorithmus liefert sämtliche Eigenwerte von *symmetrischen, tridiagonalen Matrizen*. Das bedeutet, die Matrix A muss in *Tridiagonalgestalt* gebracht werden. Dies liefert uns der folgenden Satz:

SATZ 5.12. *Sei $A \in M\left(n \times n; \mathbb{R}\right)$ symmetrisch. Dann existiert eine orthogonale Matrix $P \in O(n)$, welche das Produkt von (n-2)* Householder-Reflexionen *darstellt, so dass PAP^T Tridiagonalgestalt hat:*

$$A = \begin{pmatrix} * & \dots & \dots & *\\ \vdots & & & \vdots\\ \vdots & & & \vdots\\ * & \dots & \dots & *\end{pmatrix} \xrightarrow{P_1 \cdot} \begin{pmatrix} * & * & \dots & *\\ * & \vdots & & \vdots\\ 0 & \vdots & & \vdots\\ 0 & * & \dots & *\end{pmatrix} \xrightarrow{\cdot P_1^T} \begin{pmatrix} * & * & 0 & 0\\ * & * & \dots & *\\ 0 & \vdots & & \vdots\\ 0 & * & \dots & *\end{pmatrix}$$

Wir iterieren denn gezeigten Prozess und erhalten so Householder-Reflexionen $P_1, \dots, P_{n-2}$ derart, dass

$$\underbrace{P_{n-2} \cdot \ldots \cdot P_1}_{=P} A \underbrace{P_1^T \cdot \ldots \cdot P_{n-2}^T}_{=P^T} = \begin{pmatrix} * & * & & \\ * & \ddots & \ddots & \\ & \ddots & \ddots & *\\ & & * & *\end{pmatrix}$$

Damit haben wir somit unser Ausgangsproblem auf die Bestimmung der Eigenwerte einer symmetrischen Tridiagonalmatrix transformiert.

Die erste Idee eines sogenannten LR-Algorithmus für symmetrische Tridiagonalmatrizen geht auf H.Rutishauser zurück. Er hatte zunächst ausprobiert, was passiert, wenn man die Faktoren der LR-Zerlegung einer Matrix $A = LR$ vertauscht, $A' = RL$, und diesen Prozess von Zerlegen und Vertauschen iteriert. Dabei zeigt sich, dass in vielen Fällen die Folge der so konstruierten Matrizen gegen die Diagonalmatrix Λ der Eigenwerte konvergiert. Bei dem auf J.G.F.Francis(1959) und V.N.Kublanovskaja(1961) zurückgehende QR-Algorithmus verwendet man statt einer LR-Zerlegung eine QR-Zerlegung. Diese exisitiert immer(keine Permutation nötig) und ist vor allen Dingen stabil. Wir definieren daher eine Folge $(A_k)_{k=1,2,\ldots}$ von Matrizen durch

1. $A_1 = A$
2. $A_k = Q_k R_k$, QR-Zerlegung
3. $A_{k+1} = R_k Q_k$.

Lemma 5.13. Die Matrizen A_k haben folgende Eigenschaften:

- Die Matrizen A_k sind alle konjugiert zu A, denn es gilt $A_{k+1} = Q_k^T A_k Q_k$.
- Ist A symmetrisch, so auch alle A_k.
- Ist A symmetrisch und tridiagonal, so auch alle A_k.

SATZ 5.14. *Sei $A \in M(n \times n; \mathbb{R})$ symmetrisch mit den Eigenwerten $\lambda_1, \ldots, \lambda_n$, so dass*

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0$$

und A_k, Q_k, R_k wie zuvor definiert. Dann gilt mit $A_k = \left(a_{ij}^{(k)}\right)$:

- $\lim_{k\to\infty} Q_k = I$,
- $\lim_{k\to\infty} R_k = \Lambda$,
- $a_{i,j}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right)$ *für $i > j$.*

5.3 Lanczos-Verfahren

Dies ist ein Verfahren um große dünn besetzte Matrizen in Tridiagonale Gestalt zu bringen. Sei Q eine orthogonale Matrix, so dass

$$A = Q\,T\,Q^T \text{ mit } Q = (q_1, q_2, \ldots, q_n)$$

wobei T eine Tridiagonalmatrix der Form

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \ldots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ddots & \vdots \\ 0 & \beta_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_{n-1} \\ 0 & \ldots & 0 & \beta_{n-1} & \alpha_n \end{pmatrix}$$

ist. Aus $A = Q\,T\,Q^T \Rightarrow AQ = QT$. Wenn wir die Spalten der beiden Seiten gleichsetzen, erhalten wir

$$\begin{aligned} Aq_1 &= \alpha_1 q_1 + \beta_1 q_2 \\ Aq_i &= \beta_{i-1} q_{i-1} + \alpha_i q_i + \beta_i q_{i+1}, \quad i = 2, \ldots, n-1 \qquad (*) \\ Aq_n &= \beta_{n-1} q_{n-1} + \alpha_n q_n \end{aligned}$$

Die Orthonormalität von Q ergibt

$$q_i^T q_i = 1 \text{ und } q_i^T q_j = 0 \text{ für } i \neq j$$

so dass die Multiplikation von $(*)$ mit q_i^T die Elemente

$$\alpha_i = q_i^T A q_i$$

liefert. Um β_1 zu erhalten schreiben wir

$$\beta_1 q_2 = Aq_1 - \alpha_1 q_1$$

und nehmen auf beiden Seiten die Norm, woraus sich

$$\beta_1 = \|Aq_1 - \alpha_1 q_1\|_2$$

ergibt. Durch gleiche Vorgangsweise gilt weiters

$$\beta_i = \|Aq_i - \alpha_i q_i - \beta_{i-1} q_{i-1}\|_2$$

Dies liefert uns folgenden Lanczos-Algorithmus

```
Wähle q_1 mit q_1^T · q_1 = 1 z.B. (1,0,...,0)  β_0 = 0 und q_0 = 0
for i = 1,...,n-1
   p_i = A q_i
   α_i = q_i^T p_i
   w_i = p_i - α_i q_i - β_{i-1} q_{i-1}
   β_i = ||w_i||_2
   q_{i+1} = β_i^{-1} w_i
end
α_n = q_n^T A q_n
```

Man kann leicht durch Nachrechnen zeigen, dass

$$q_i^T q_i = 1 \text{ und } q_i^T q_j = 0 \text{ für } i \neq j$$

gilt. Beispielsweise gilt

$$q_1^T q_2 = \beta_1^{-1} q_1^T \left(Aq_1 - \alpha_1 q_1\right) = \beta_1^{-1} \left(\alpha_1 - \alpha_1\right) = 0$$

und

$$q_i^T q_{i+1} = \beta_i^{-1} q_i^T \left(Aq_i - \alpha_i q_i - \beta_{i-1} q_{i-1}\right) = \beta_i^{-1} \left(\alpha_i - \alpha_i\right) = 0$$

Angenommen wird, dass $\beta_i \neq 0$ ist. Falls $\beta_i = 0$ ist, dann gilt $Aq_i = \alpha_i q_i$. Also ist α_i ein Eigenwert und q_i ein Eigenvektor. Sobald $\beta_i = 0$ aufritt, erhalten wir die Gleichung

$$AQ_i = Q_i T_i \quad Q_i = (q_1, \ldots, q_i)$$

was signalisiert, dass wir i Eigenwerte von A in $T_i$auffinden können. Dann wählen wir $q_{i+1} \perp \{q_1, \ldots, q_i\}$ und starten den Algorithmus erneut. Unterbrechen wir die Iteration im Algorithmus bei k(statt $n-1$) Schritten erhalten wir eine $k \times k$-Tridiagonalmatrix T_k, deren Eigenwerte folgende Eigenschaften besitzen

$$\begin{aligned} \lambda_{min}(A) &\leq \lambda_{min}(T_k) \\ \lambda_{max}(A) &\geq \lambda_{max}(T_k) \end{aligned}$$

Die Eigenwerte von T_k können wir nun beispielsweise mit dem QR-Algorithmus berechnen.

5.4 Verallgemeinerte symmetrische Eigenwertprobleme

In vielen Anwendungen(z.B. Strukturmechanik) trifft man auf das sogenannte *verallgemeinerte symmetrische Eigenwertproblem*,

$$Ax = \lambda Bx,$$

wobei die Matrizen $A, B \in M(n \times n, \mathbb{R})$ beide symmetrisch sind und B zusätzlich positiv definit. Setzen wir die Cholesky-Zerlegung $B = LL^T$ von B ein, so gilt

$$\begin{aligned} Ax = \lambda Bx &\Leftrightarrow Ax = \lambda LL^T x \\ &\Leftrightarrow \underbrace{\left(L^{-1}AL^{-T}\right)}_{=:\bar{A}} \underbrace{L^T x}_{=:\bar{x}} = \lambda L^T x. \end{aligned}$$

Da $\bar{A} = L^{-1}AL^T$ wieder symmetrisch ist, ist das verallgemeinerte Eigenwertproblem $Ax = \lambda Bx$ äquivalent zu dem symmetrischen Eigenwertproblem $\bar{A}\bar{x} = \lambda\bar{x}$.

Kapitel 6

Interpolation

6.1 Lineare Interpolationsprobleme

Sei $f : \mathbb{R} \to \mathbb{R}$ in Form experimenteller Daten gegeben, d. h. es liegen nur einige diskrete Funktionswerte $f(x_i)$ und eventuell noch Ableitungen $f^{(j)}(x_i)$ an endlich vielen Punkten x_i vor.
Unsere Aufgabe ist es nun eine Funktion zu finden, die diese Messpunkte beinhaltet (sie müssen explizit auf der Funktion zu liegen kommen)und eine gute Vorhersage für die Lücken zwischen den gegebenen Punkten liefert. Gegeben sei eine Funktion einer Variablen x

$$\varphi(x; a_0, \ldots, a_n),$$

die von $n+1$ weiteren reellen oder komplexen Parametern $a_0, \ldots, a_n$ abhängt. Ein Interpolationsproblem liegt dann vor, wenn die Parameter a_i so bestimmt werden sollen, dass für $n+1$ gegebene Paare von reellen oder komplexen Zahlen $(x_i, f_i), i = 0, \ldots, n$ mit $x_i \neq x_k$ für $i \neq k$ gilt

$$f_i = f(x_i) = \varphi(x_i; a_0, \ldots, a_n), \quad i = 0, \ldots, n.$$

Die Paare (x_i, f_i) werden als *Stützpunkte* bezeichnet; die x_i heißen *Stützabzissen*, die f_i heißen *Stützordinaten* oder *Stützwerte*. Manchmal werden auch Werte der Ableitungen von φ an den Stützabzissen x_i vorgeschrieben. Ein **lineares Interpolationsproblem** liegt vor, wenn φ linear von den Parametern a_i abhängt:

$$\varphi(x; a_0, \ldots, a_n) \equiv a_0\varphi_0(x) + a_1\varphi_1(x) + \ldots + a_n\varphi_n(x).$$

zu diesen linearen Problemen gehören das Problem der *Interpolation durch Polynome*

$$\varphi(x; a_0, \ldots, a_n) \equiv a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$$

und der *trigonometrischen Interpolation*

$$\varphi(x; a_0, \ldots, a_n) \equiv a_0 + a_1 \exp^{xi} + a_2 \exp^{2xi} + \ldots + a_n \exp^{nxi} \quad \left(i^2 = -1\right).$$

Zu den **nicht linearen Interpolationsproblemen** gehören die Interpolation durch *rationale Funktionen*

$$\varphi(x; a_0, \ldots, a_n, b_0, \ldots, b_m) \equiv \frac{a_0 + a_1x + a_2x^2 + \ldots + a_nx^n}{b_0 + b_1x + b_2x^2 + \ldots + b_mx^m},$$

und durch Exponentialsummen

$$\varphi(x; a_0, \ldots, a_n, \lambda_0, \ldots, \lambda_m) \equiv a_0 \exp^{\lambda_0 x} + a_1 \exp^{\lambda_1 x} + \ldots + a_n \exp^{\lambda_n x}.$$

6.2 Interpolation durch Polynome

Im Allgemeinen suchen wir eine Approximation zu f durch ein Polynom P. Grund dafür ist folgender Satz:

SATZ 6.0 (Weierstrass Approximation Satz). *Sei f eine stetige Funktion auf $[a, b]$ und $\epsilon > 0$ gegeben. Dann existiert ein Polynom P so, dass $|f(x) - P(x)| < \epsilon \quad \forall x \in [a, b]$. (Dieser Satz sagt uns leider nicht wie man P konstruieren muss.)*

6.2.1 Interpolation mit Taylor

Zunächst betrachten wir die Konstruktion von P durch die Taylor-Entwicklung:

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)\frac{(x - x_0)^2}{2!} + \ldots + f^{(n)}(x_0)\frac{(x - x_0)^n}{n!}$$

Dafür brauchen wir $f \in C^{n+1}$ so gilt für den Approximationsfehler, dass

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}$$

mit $\xi(x) \in U_{x0}$. Der Taylor Ansatz ist nur anwendbar, falls alle Ableitungen $f^{(j)} \quad j = 1, \ldots, n$ zur Verfügung stehen und die Approximation in der Nähe von x_0 gesucht wird. Tatsächlich wächst der Fehler für x das weit von x_0 entfernt ist wie der Fehler $(x - x_0)^{n+1}$.

6.2.2 Interpolationsformel von Lagrange

In Wirklichkeit ist der Fall jedoch realistischer, in dem f (und noch wenige andere Ableitungen) auf $x_i \quad i = 0, \ldots, n$ vorhanden ist. Sei $f_i = f(x_i) \quad i = 0, \ldots, n$ an den Knoten $x_0, \ldots, x_n \quad (x_i \neq x_j \quad i \neq j)$ gegeben. Wir suchen nun ein Polynom P vom Grad $\leq n$:

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$$

so dass

$$P(x_i) = f_i, \quad i = 0, \ldots, n$$

dieser Ansatz resultiert in einem eindeutig bestimmten Polynom, dessen Koeffizienten a_k, $k = 0, \ldots, n$ die Lösung von folgendem Vandermonde System ist

$$V \cdot \underline{a} = \underline{f}.$$

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

Die Matrix V heißt Vandermonde Matrix. Ihre Determinante ist gegeben durch

$$det(V) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_i - x_j).$$

Das bedeutet, dass V singulärer wird, je kleiner die Differenz $(x_i - x_j)$ ist. Das Vandermonde System lösen wir mit der Gauss-Elimination ohne Pivotsuche. Der Aufwand beträgt $\approx n^3$.
Eine alternative Basis zur Darstellung des Interpolationspolynoms bilden die sogenannten Lagrange-Polynome $L_0, \ldots L_n$. Sie sind definiert als die eindeutig bestimmten Polynome $L_i \in \mathbb{P}_n$ mit

$$L_{ni}(x_j) = \delta_{ij}.$$

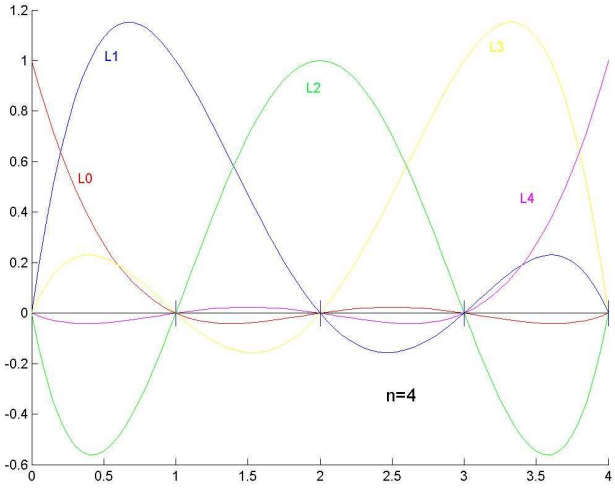
In expliziter Form

$$L_{ni}(x) = \prod_{j=0, j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)}.$$

Das Interpolationspolynom für beliebige Stützwerte $f_0, \ldots, f_n$ lässt sich aus den Lagrange-Polynomen durch Superposition aufbauen:

$$P(x) = \sum_{i=0}^{n} f_i L_{ni}(x)$$

und es gilt offensichtlich $P(x_j) = \sum_{i=0}^{n} f_i L_{ni}(x_j) = f_j$



SATZ 6.1. *Sei $f \in C^{n+1}[a,b]$ und $x_0, x_1, \ldots, x_n \in [a,b]$ mit $x_i \neq x_j$ für $i \neq j$. Dann existieren $\xi(x) \in [a,b]$ so, dass*

$$f(x) - P_L(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n).$$

6.2.3 Der Algorithmus von Neville

SATZ 6.2 (Rekursionsformel, Satz von Aitken)**.** *Sei $f(x)$ gegeben auf $x_0, \ldots x_k$ und x_j, x_i sind zwei Stützpunkte davon mit $x_j \neq x_i$. Dann ist*

$$P(x) = \frac{1}{(x-x_j)}\left[(x-x_j)P_{0,1,\ldots,j-1,j+1,\ldots,k}(x) - (x-x_i)P_{0,1,\ldots,i-1,i+1,\ldots,k}(x)\right]$$

das K Lagrange-Polynom auf $x_0, \ldots x_k$.

Mit Hilfe dieses Satzes wird die rekursive Berechnung von $P(x)$ für ein festes $\bar{x}$ möglich. Zum Beispiel seien $f_0, f_1, \ldots, f_n$ auf $x_0, \ldots, x_n$ gegeben. Sei $P_{i,k} = P(f|x_{i-k}, \ldots, x_i)(\bar{x})$ der Wert vom Interpolationspolynom K-ten Grades an der Stelle x_i. Wir haben das **Schema von Neville**:

$$P_{i,0} = f_i \quad \text{für} \quad i = 0, \ldots, n$$

$$P_{i,k} = P_{i,k-1} + \frac{\bar{x} - x_i}{x_i - x_{i-k}}(P_{i,k-1} - P_{i-1,k-1}) \quad i \geq k$$

Graphisch:

$$\begin{array}{cccccccccc}
f_0 & = & P_{0,0} & \searrow & & & & & & \\
f_1 & = & P_{1,0} & \rightarrow & P_{1,1} & \searrow & & & & \\
\vdots & = & \vdots & & & & & & & \\
f_{n-1} & = & P_{n-1,0} & \rightarrow & P_{n-1,1} & \rightarrow & \ldots \rightarrow & P_{n-1,n-1} & \searrow & \\
f_n & = & P_{n,0} & \rightarrow & P_{n,1} & \rightarrow & \ldots \rightarrow & P_{n,n-1} & \rightarrow & P_{n,n} \\
 & & \textit{(k=0)} & & \textit{(k=1)} & & & \textit{(k=n-1)} & & \textit{(k=n)}
\end{array}$$

(Iterated Interpolation)

Beispiel 6.3.

x_n	x	$f(x)$
x_0	*1.0*	*0.765*
x_1	*1.3*	*0.620*
x_2	*1.6*	*0.455*
x_3	*1.9*	*0.281*
x_4	*2.2*	*0.110*

für $\bar{x} = 1.5 \Rightarrow$
$P_{0,0} = 0.765 \quad P_{1,0} = 0.620 \quad P_{2,0} = 0.455 \quad P_{3,0} = 0.281 \quad P_{4,0} = 0.110$
$P_{1,1}(1.5) = P_{1,0} + \frac{(1.5)-(1.3)}{(1.3)-(1.0)}(P_{1,0} - P_{0,0}) = 0.523$
$P_{2,1}(1.5) = P_{2,0} + \frac{(1.5)-(1.6)}{(1.6)-(1.3)}(P_{2,0} - P_{1,0}) = 0.510$
$P_{3,1}(1.5) = 0.513$
$P_{4,1}(1.5) = 0.510$
$P_{2,2} = P_{2,1} + \frac{\bar{x}-x_2}{x_2-x_0}(P_{2,1} - P_{1,1}) = 0.512...$

1.0	*0.765*				
1.3	*0.620*	*0.523*			
1.6	*0.455*	*0.510*	*0.512*		
1.9	*0.281*	*0.513*	*0.511*	*0.51181*	
2.2	*0.110*	*0.510*	*0.513*	*0.51183*	*0.51182*

d.h wir erhalten aus den Stützpunkten für $\bar{x} = 1.5$ *den Funktionswert 0.51182*

6.3 Dividierte Differenzen

Mit der iterierten Interpolation haben wir die Werte immer höheren Grades errechnet. Wir werden durch die Polynome für ein festes $\bar{x}$ dividieren. Mit den dividierten Differenzen berechnen wir explizit die Koeffizienten von

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

der angegebenen Daten. Wir brauchen folgende Definitionen:

1) $f[x_i] = f(x_i)$
2) $f[x_i, x_{i+1}] = \frac{f[x_{i+1}]-f[x_i]}{x_{i+1}-x_i}$

$\vdots \quad \vdots = \vdots$

k) $f[x_i, \ldots, x_{i+k}] = \frac{f[x_{i+1},\ldots,x_{i+k}]-f[x_i,\ldots,x_{i+k-1}]}{x_{i+k}-x_i}$

Durch die Substitution erkennt man, dass

$$a_k = f[x_0, x_1, \ldots, x_k]$$

mit

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)\cdots(x - x_{k-1})$$

Man kann sich nun folgende Tabelle vorstellen:

x	$f(x)$	$f[x_i, x_j]$	$f[x_i, x_j, x_k]$	$f[x_i, x_j, x_k, x_l]$
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0}$		
x_2	$f[x_2]$	$f[x_1, x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1}$	$f[x_0, x_1, x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$	
x_3	$f[x_3]$	$f[x_2, x_3] = \frac{f[x_3]-f[x_2]}{x_3-x_2}$	$f[x_1, x_2, x_3] = \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	$f[x_0, x_1, x_2, x_3] = \frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$

6.4 Hermite-Polynome

Bei einer verallgemeinerten Interpolationsaufgabe sind neben den Funktionswerten $f(x_i)$ noch Ableitungen an den Knoten gegeben.

SATZ 6.4. *Sei $f \in C^1[a,b]$ und $x_0, \ldots, x_n \in [a,b] \quad x_i \neq x_j \quad i \neq j$. Dann ist das eindeutig bestimmte Polynom H mit (dem kleinst möglichen) Grad $\leq 2n+1$ mit*

$$H(x_i) = f(x_i) \qquad H'(x_i) = f'(x_i) \qquad i = 0, \ldots, n$$

gegeben durch

$$H(x) = \sum_{j=0}^{n} f(x_j) \cdot H_{nj}(x) + \sum_{j=0}^{n} f'(x_j) \cdot \hat{H}_{nj}(x)$$

wobei

$$H_{nj}(x) = \left[1 - 2(x - x_j) L'_{nj}(x_j)\right] L^2_{nj}(x)$$

und

$$\hat{H}_{nj}(x) = (x - x_j) L^2_{nj}(x)$$

mit

$$L_{nj}(x) = \prod_{i=0, i\neq j}^{n} \frac{x - x_i}{x_j - x_i}.$$

Weiters ist $f \in C^{2n+2}[a,b]$, dann folgt

$$f(x) - H(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} (x - x_0)^2 \cdots (x - x_n)^2 \quad \textit{mit} \quad \xi \in [a,b].$$

Zu zeigen ist dass:

$$H(x_k) = f(x_k) \quad \text{und} \quad H'(x_k) = f'(x_k)$$

Dazu verwendet man folgende Eigenschaften:

a)
$$H_{nj}(x_k) = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$$

b)
$$H'_{nj}(x_k) = 0 \quad \forall k$$

c)
$$\hat{H}_{nj}(x_k) = 0 \quad \forall k$$

d)
$$\hat{H}'_{nj}(x_k) = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$$

Beispiel 6.5. $m = 2$

k	x_k	$f(x_k)$	$f'(x_k)$
0	*1.3*	*0.620*	*-0.522*
1	*1.6*	*0.455*	*-0.569*
2	*1.9*	*0.281*	*0.581*

Bessel Funktion 1.Grades

Zuerst berechnen wir $L_{2,j}$ *und* $L'_{2,j}$

$$\begin{array}{l|l} L_{20}(x) = \frac{50}{9}x^2 - \frac{175}{9}x + \frac{152}{9} & L'_{20}(x) = \frac{100}{9}x - \frac{175}{9} \\ L_{21}(x) = \frac{-100}{9}x^2 - \frac{320}{9}x + \frac{247}{9} & L'_{21}(x) = \frac{-200}{9}x - \frac{320}{9} \\ L_{22}(x) = \frac{50}{9}x^2 - \frac{145}{9}x + \frac{104}{9} & L'_{22}(x) = \frac{100}{9}x - \frac{145}{9} \end{array}$$

$\Longrightarrow$

$$\begin{array}{l|l} H_{20}(x) = (10x - 12)\left(\frac{50}{9}x^2 - \frac{175}{9}x + \frac{152}{9}\right)^2 & \hat{H}_{20}(x) = (x - 1.3)\, L_{20}^2 \\ H_{21}(x) = (1)\left(\frac{-100}{9}x^2 - \frac{320}{9}x + \frac{247}{9}^2\right) & \hat{H}_{21}(x) = (x - 1.6)\, L_{21}^2 \\ H_{22}(x) = 10\,(2 - x)\left(\frac{50}{9}x^2 - \frac{145}{9}x + \frac{104}{9}^2\right) & \hat{H}_{22}(x) = (x - 1.9)\, L_{22}^2 \end{array}$$

$\Longrightarrow$

$$H(x) = 0.620 \cdot H_{20}(x) + 0.455 \cdot H_{21}(x) + 0.281 \cdot H_{22}(x) - 0.522 \cdot \hat{H}_{20}(x) - 0.569 \cdot \hat{H}_{21}(x) - 0.581 \cdot \hat{H}_{22}(x)$$

Approximation für $x = 1.5 \quad \rightarrow \quad H(1.5) = 0.5118277.$

Beispiel 6.6. *Gegeben sind folgende Werte-Paare:* $f(x_0) = 0 \quad f(x_1) = 1 \quad f'(x_0) = 1 \quad f'(x_1) = 0$, *mit n=1.*

$$H(x) = \sum_{j=0}^{n} f(x_j)\, H_{nj}(x) + \sum_{j=0}^{n} f'(x_j)\, \hat{H}_{nj}(x)$$

$$\Rightarrow \quad H(x) = f(x_1)\, H_{n1}(x) + f'(x_0)\, \hat{H}_{n0}(x)$$

da $f(x_0) = 0 \quad f'(x_1) = 0$ *(wir haben den Aufwand dadurch halbiert). Die Larange-Polynome sind allgemein gegeben durch*

$$L_{ni}(x) = \prod_{j=0 j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

In unserem Fall erhalten wir

$$L_{10} = \frac{x - x_1}{x_0 - x_1} = 1 - x \quad L_{11} = \frac{x - x_0}{x_1 - x_0} = x \quad mit \quad x_0 = 0 \quad x_1 = 1$$

weiters folgt

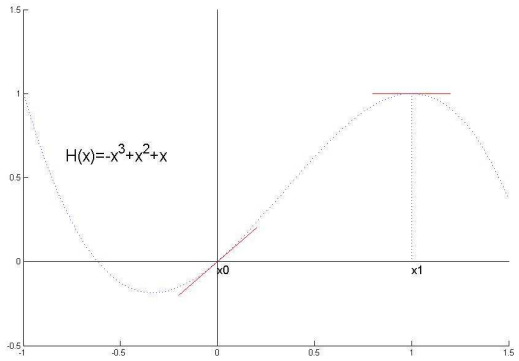
$$H_{n1}(x) = \left[1 - 2\,(x - x_1)\, L'_{11}(x_1)\right] L_{11}^2(x) = \left[1 - 2\,(x - 1) \cdot 1\right] x^2 = (3 - 2x)\, x^2$$

$$\hat{H}_{n0}(x) = (x - x_0)\, L_{10}^2(x) = x\,(1 - x)^2$$

woraus die gesuchte Funktion gegeben ist durch

$$H(x) = 1 \cdot (3 - 2x)\, x^2 + 1 \cdot x\,(1 - x)^2 = 3x^2 - 2x^3 + x - 2x^2 + x^3 = -x^3 + x^2 + x$$

Graphische Auswertung:



6.5 Splines

Bei einer großen Anzahl von äquidistanten Knoten ist die klassische Polynominterpolation ungeeignet, weil Polynome höheren Grades zu starken Oszillationen neigen. Deswegen werden wir jetzt lokale Polynome niedrigen Grades verwenden und diese an den Stützpunkten miteinander verheften. Wir konzentrieren uns auf **kubische Splines**, das heißt, Polynomen dritten Grades stückweise definiert durch $S = S_j$ wobei

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3 \quad j = 0, \ldots, n-1$$

in $[x_j, x_{j+1}]$.
Durch die $4n$ Parameter (a_j, b_j, c_j, d_j) können wir S_j, $j = 0, \ldots, n-1$, so konstruieren, dass $S(x), S'(x)$ und $S''(x)$ auf x_j stetig sind und $S(x_j) = f(x_j)$. Dies führt zu folgenden

Bedingungen für kubische-Splines

(a) S ist ein kubisches Polynom mit S_j auf $[x_j, x_{j+1}] \quad j = 0, \ldots, n-1$

(b) $S(x_j) = f(x_j), \quad j = 0, 1, \ldots, n$

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1}), \quad j = 0, \ldots, n-2$

(d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1}), \quad j = 0, \ldots, n-2$

(e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1}), \quad j = 0, \ldots, n-2$
So haben wir $(n+1) + 3(n-1) = 4n-2$ Gleichungen. Die zusätzlich notwendige zwei Bedingungen werden gegeben in Form von

(f) Randbedingungen:
(1) $S''(x_0) = S''(x_n) = 0$ Freie R.B./Natural Splines
(2) $S'(x_0) = f'(x_0), \quad S'(x_n) = f'(x_n)$ Feste R.B./Clamped Splines

Wir verwenden (a)-(f) um die Koeffizienten a_j, b_j, c_j, d_j, $j = 0, \ldots, n-1$ zu bestimmen.
Um die Notation zu erleichtern, definieren wir $h_j = x_{j+1} - x_j$.

a_j)
Mit (a) und (b) haben wir $S(x_j) = S_j(x_j) = a_j = f(x_j)$, $j = 0, \ldots, n-1$.
Weiters haben wir $S(x_n) = S_{n-1}(x_n) = a_{n-1} + b_{n-1}h_{n-1} + c_{n-1}h_{n-1}^2 + d_{n-1}h_{n-1}^3 = f(x_n) =: a_n$.
Daher schreiben wir

$$a_j = f(x_j), \quad j = 0, \ldots, n.$$

Im Allgemein, durch (c) folgt

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3, \quad j = 0, \ldots, n-1.$$

b_j)
Die Ableitung $S'(x)$ auf $[x_j, x_{j+1}]$ ist gegeben durch $S'(x) = S'_j(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$, mit $S'(x_0) = b_0$. Weiters haben wir $S'(x_n) = S'_{n-1}(x_n) = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2 = f'(x_n) =: b_n$. Und im Allgemein durch (d) folgt

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2, \quad j = 0, \ldots, n-1.$$

c_j)
Die zweite Ableitung $S''(x)$ auf $[x_j, x_{j+1}]$ ist gegeben durch $S''(x) = S''_j(x) = 2c_j + 6d_j(x - x_j)$, mit $S''(x_0) = 2c_0$.
Weiters haben wir $S''(x_n) = S''_{n-1}(x_n) = 2c_{n-1} + 6d_{n-1}h_{n-1} = f''(x_n) =: 2c_n$. Und im Allgemein durch (e) folgt

$$c_{j+1} = c_j + 3d_j h_j, \quad j = 0, \ldots, n-1.$$

Jetzt zeigen wir dass es möglich ist die Koeffizienten b_j und d_j durch a_j und c_j zu definieren.
Von der letzte Gleichung erhalten wir

$$d_j = \frac{1}{3h_j}(c_{j+1} - c_j) \quad j = 0, \ldots, n-1.$$

Verwenden wir dieses Resultat in $a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$ haben wir

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}), \quad j = 0, \ldots, n-1.$$

Mit diese Ergebniss und $b_j = b_{j-1} + 2c_{j-1}h_{j-1} + 3d_{j-1}h_{j-1}^2$ und $d_{j-1} = \frac{1}{3h_{j-1}}(c_j - c_{j-1})$ wobei $j = 1, \ldots, n-1$, finden wir eine Gleichung für c_j mit gegebenen a_j:

$$\begin{aligned}
b_j &= b_{j-1} + 2c_{j-1}h_{j-1} + 3d_{j-1}h_{j-1}^2 \\
\frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) &= \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) + 2c_{j-1}h_{j-1} + 3d_{j-1}h_{j-1}^2 \\
c_{j-1}h_{j-1} + 2c_j h_{j-1} + 2c_j h_j + c_{j+1}h_j &= \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})
\end{aligned}$$

Letztendlich erhalten wir folgende Gleichung

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

für $j = 1, \ldots, n-1$.
Die zusätzlichen zwei Gleichung für c_0 und c_n werden durch die Randbedingungen definiert.
Im Fall von freie Randbedingungen erhalten wir

$$S''(x_0) = 2c_0 = 0 \text{ und } S''(x_n) = 2c_n = 0.$$

Daraus folgt

SATZ 6.7. *Sei $f(x_j)$ gegeben für $x_0 < x_1 < \ldots < x_n$. Die eindeutige Spline Interpolation (Approximation) von f mit freien Randbedingungen $S''(x_0) = S''(x_n) = 0$, ist gegeben durch die Lösung von $A\,\underline{c} = \underline{r}$ wobei $\underline{c} = (c_0, \ldots, c_n)$ und*

$$A = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
h_0 & 2(h_0 + h_1) & h_1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & h_{n-2} & 0 \\
0 & \cdots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\
0 & \ddots & 0 & 0 & 1
\end{bmatrix}$$

$$\underline{r} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix}$$

Im Fall von feste Randbedingungen erhalten wir

$$S'(x_0) = f'(x_0) = b_0 = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1).$$

Daraus folgt

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(x_0)$$

Weiters

$$\begin{aligned} S'(x_n) &= f'(x_n) =: b_n = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2 \\ &= \frac{1}{h_{n-1}}(a_n - a_{n-1}) - \frac{h_{n-1}}{3}(2c_{n-1} + c_n) + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2 \\ &= \frac{1}{h_{n-1}}(a_n - a_{n-1}) + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n) \end{aligned}$$

Daraus folgt

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = -\frac{3}{h_{n-1}}(a_n - a_{n-1}) + 3f'(x_n)$$

Als Zusammenfassung haben wir

SATZ 6.8. *Sei $f(x_j)$ gegeben für $x_0 < x_1 < \ldots < x_n$. Die eindeutige Spline Interpolation (Approximation) von f mit festen Randbedingungen $S'(x_0) = f'(x_0)$ und $S'(x_n) = f'(x_n)$, ist gegeben durch die Lösung von $A\,\underline{c} = \underline{r}$ wobei $\underline{c} = (c_0, \ldots, c_n)$ und*

$$A = \begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \cdots & 0 \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}$$

$$\underline{r} = \begin{bmatrix} \frac{3}{h_0}(a_1 - a_0) - 3f'(x_0) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3f'(x_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix}$$

Durch lösen von $A \cdot \underline{c} = \underline{r}$ erhalten wir $\underline{c} = (c_0, \ldots, c_n)$. Mit

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \text{ und } d_j = \frac{1}{3h_j}(c_{j+1} - c_j), \quad j = 0, \ldots, n-1.$$

sind dann b_j und d_j und daher S_j bestimmt.

Algorithmus-Splines:

Input: n; $x_0, \ldots, x_n$; $a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n)$
Case: $S''(x_0) = S''(x_n) = 0$ or $S'(x_0) = f'(x_0)$ $S'(x_n) = f'(x_n)$

For $j = 0, \ldots, n-1$
  $h_j = x_{j+1} - x_j$
end
Assemble A $(n+1) \times (n+1)$ Matrix
(only row 0 and row n depend on case)
Set RHS For $i = 1, \ldots, n-1$
  $r_i = \frac{3[a_{i+1}h_{i-1} - a_i(x_{i+1} - x_{i-1}) + a_{i-1}h_i]}{h_{i-1}h_i}$; r_0, r_n depend on R.B. case
  Solve $A\underline{c} = \underline{r}$
  For $j = 0, \ldots, n-1$
    $b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$
    $d_j = \frac{1}{3h_j}(c_{j+1} - c_j)$
end
Output: $\underline{a}, \underline{b}, \underline{c}, \underline{d}$

SATZ 6.9. *Sei $f \in C^4[a,b]$ mit $M \geq max_{x\in[a,b]} \left|f^{(4)}(x)\right|$ und sei S die eindeutige Spline mit fester Randbedingung für f auf $a = x_0 < \ldots < x_n = b$. Dann*

$$\max_{x\in[a,b]} |f(x) - S(x)| \leq \frac{5M}{384} \max_{0\leq j\leq n-1} (x_{j+1} - x_j)^4 .$$

Kapitel 7

Bestimmte Integrale

7.1 Quadraturformeln

Wir diskutieren die Berechnung des Riemann-Integrals

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

Dabei ist f eine stückweise stetige Funktion auf dem Intervall $[a, b]$. Unser Ziel ist die Konstruktion von *positiven Linearformen*

$$\hat{I} : C[a, b] \to \mathbb{R}, \quad f \mapsto \hat{I}(f),$$

die das Integral I möglichst gut approximiert, d.h.

$$\hat{I}(f) - I(f) \text{ „klein“}.$$

Definition 7.0. Unter einer *Quadraturformel* $\hat{I}$ zur Berechnung des bestimmten Integrals verstehen wir eine Summe

$$\hat{I}(f) = (b-a)\sum_{i=0}^{n} w_i f(x_i)$$

mit den Knoten $x_0, \ldots, x_n$ und den *Gewichten* $w_0, \ldots, w_n$, so dass

$$\sum_{i=0}^{n} w_i = 1. \tag{7.1}$$

Die Bedingung (7.1) an die Gewichte garantiert uns, dass die Quadraturformel konstante Funktionen exakt integriert, d.h. $\hat{I}(1) = I(1) = b-a$. Ferner liegt auf der Hand, dass eine Quadraturformel genau dann positiv ist, wenn es alle ihre Gewichte sind, d.h.

$$\hat{I} \text{ postiv } \Leftrightarrow w_i \geq 0 \quad \forall i = 0, \ldots, n.$$

7.2 Newton-Cotes-Formeln

Die Grundidee besteht darin, die Funktion f durch eine Approximation(Interpolation) $\hat{f}$ zu ersetzen, für die sich die Quadratur einfach ausführen lässt, und $I(\hat{f})$ als Approximation von $I(f)$ anzusehen, also

$$\hat{I}(f) := I(\hat{f}).$$

Insbesondere ist für gegebene Knoten $x_0, \ldots, x_n$

$$\hat{f}(x) = \sum_{i=0}^{n} f(x_i)L_i(x),$$

das Interpolationspolynom von f, wobei $L_i \in P_n$ das i-te Lagrange-Polynom zu den Knoten x_i ist, d.h. $L_i(x_j) = \delta_{ij}$. Dieser Ansatz liefert die Quadraturformel

$$\hat{I}(f) = (b-a)\sum_{i=0}^{n} w_i^L f(x_i)$$

wobei die Gewichte

$$w_i^L := \frac{1}{b-a}\int_a^b L_i(x)dx$$

nur von der Wahl der Knoten $x_0, \ldots, x_n$ abhängen.
Per Konstruktion ist klar, dass die so definierten Quadraturformeln für Polynome $P \in P_n$ vom Grad kleiner oder gleich n exakt sind,

$$\hat{I}(P) = I(P_n(P)) = I(P) \text{ für } P \in P_n.$$

Für vorgegebene Knoten x_i ist die Quadraturformel durch diese Eigenschaft sogar bereits eindeutig bestimmt.

Für den Spezialfall *äquidisanter* Knoten

$$h_i = h = \frac{b-a}{n}, \quad x_i = a + ih, \quad i = 1, \ldots, n$$

heißen die so konstruierten Quadraturformeln *Newton-Cotes-Formeln*. Durch Substitution $s := (x-a)/h$ erhalten wir

$$w_i^L = \frac{1}{b-a}\int_a^b \prod_{j=0, j\neq i}^{n} \frac{x-x_j}{x_i-x_j}dx = \frac{1}{n}\int_0^n \prod_{j=0, j\neq i}^{n} \frac{s-j}{i-j}ds.$$

Die von den Intervallgrenzen unabhängigen Gewichten w_i^L müssen nur einmal berechnet bzw. eingegeben werden. Wir haben sie in der folgenden Tabelle bis zur Ordnung $n = 4$ zusammengestellt. Für die Ordnungen $n = 1, \ldots, 7$ sind die Gewichte und damit auch die Quadraturformeln stets positiv. Höhere Ordnungen sind weniger attraktiv, da ab $n = 8$ auch negative Gewichte auftreten können.

n	$w_i^L \quad i = 0, \ldots, n$	Fehler	Name
1	$\frac{1}{2} \quad \frac{1}{2}$	$\frac{h^3}{12}f''(\xi)$	Trapezregel
2	$\frac{1}{6} \quad \frac{4}{6} \quad \frac{1}{6}$	$\frac{h^5}{90}f^{(4)}(\xi)$	Simpson-Regel, Keplersche Faßregel
3	$\frac{1}{8} \quad \frac{3}{8} \quad \frac{3}{8} \quad \frac{1}{8}$	$\frac{3h^5}{80}f^{(4)}(\xi)$	Newton 3/8-Regel
4	$\frac{7}{90} \quad \frac{32}{90} \quad \frac{12}{90} \quad \frac{32}{90} \quad \frac{7}{90}$	$\frac{8h^7}{945}f^{(6)}(\xi)$	Milne-Regel

Wie man aus dieser Tabelle ablesen kann, ist der Approximationsfehler abhängig davon ob n gerade oder ungerade ist, d.h.

$$n \text{ gerade} \quad : \quad f \in C^{n+2}[a,b] \Rightarrow \int_a^b f dx - \hat{I}(f) = \frac{h^{n+3}f^{(n+2)}(\xi)}{(n+2)!}\int_0^n x^2(x-1)\ldots(x-n)dx$$

$$n \text{ ungerade} \quad : \quad f \in C^{n+1}[a,b] \Rightarrow \int_a^b f dx - \hat{I}(f) = \frac{h^{n+2}f^{(n+1)}(\xi)}{(n+1)!}\int_0^n x^2(x-1)\ldots(x-n)dx$$

Dies bedeutet, dass falls n gerade ist keine Verbesserung der Approximation erreicht wird mit $n \to n+1$. Um die Genauigkeit zu erhöhen muss man also $n \to n+2$ verwenden.

Die oben genannten Quadraturformeln werden normalerweise auf Teilintervalle angewendet. Wir zerlegen das Intervall $[a,b]$ in n Teilintervalle $[x_{i-1}, x_i]$ mit $i = 1, \ldots, n$,

$$a = x_0 < x_1 < \ldots < x_n = b,$$

so dass aufgrund der Additivität des Integrals

$$\int_a^b f dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f dx.$$

Demnach ist

$$\hat{I}(f) := \sum_{i=1}^{n} \hat{I}_{[x_{i-1},x_i]}(f)$$

eine (eventuell bessere) Approximation des Integrals, wobei $\hat{I}_{[x_i,x_{i+1}]}$ eine beliebige Quadraturformel auf dem Intervall $[x_i, x_{i+1}]$ bezeichne.

Kapitel 8

Nichtlineare Gleichungssysteme

8.1 Fixpunktiteration

Betrachten wir die skalare nichtlineare Gleichung (NLG) $f(x) = 0$ mit $f : \mathbb{R} \to \mathbb{R}$. Die Idee der Fixpunktiteration besteht darin, diese Gleichung äquivalent in eine Fixpunktgleichung

$$\phi(x) = x$$

umzuformen und mit Hilfe der Iteration

$$x_{k+1} = \phi(x_k), \qquad k = 0, 1, ...$$

für einen gegebenen Startwert x_0 eine gegen einen Fixpunkt x^*, mit $x^* = \phi(x^*)$, konvergierende Folge (x_k) zu konstruieren, so dass $f(x^*) = 0$ gilt.

Definition 8.0. Sei $I \subset \mathbb{R}$ ein abgeschlossenes Intervall und $\phi : I \to \mathbb{R}$. ϕ ist *kontrahierend* auf I, falls es ein $0 \leq L < 1$ gibt, so dass

$$|\phi(x) - \phi(y)| \leq L|x - y| \qquad \forall x, y \in I$$

L heißt *Lipschitz-Konstante*.

SATZ 8.1. *Ist $\phi : I \to \mathbb{R}$, $\phi \in \mathcal{C}^1(I)$, also stetig differenzierbar, so gilt*

$$\sup_{x,y \in I} \frac{|\phi(x) - \phi(y)|}{|x - y|} = \sup_{z \in I} |\phi'(z)| < \infty.$$

SATZ 8.2 (Banach)**.** *Sei $I \subset \mathbb{R}$ ein abgeschlossenes Intervall und $\phi : I \to I$ eine kontrahierende Abbildung mit Lipschitz-Konstante $L < 1$. Dann folgt:*

1. *Es existiert genau ein Fixpunkt x^*, das heißt: $x^* = \phi(x^*)$.*
2. *Für jeden Startwert $x_0 \in I$ konvergiert die Fixpunktiteration $x_{k+1} = \phi(x_k)$ gegen x^* mit*

$$|x_{k+1} - x_k| \leq L|x_k - x_{k-1}| \text{ und } |x_k - x^*| \leq \frac{L^k}{1 - L}|x_1 - x_0|$$

Für Systeme von NLG gilt:

SATZ 8.3. *Sei $D = \{(x_1, .., x_n) \in \mathbb{R}^n \mid a_i \leq x_i \leq b_i, \quad i = 1, .., n\}$ mit a_i, b_i konstant. Weiters sei $\phi : D \subset \mathbb{R}^n \to \mathbb{R}^n$, $\phi = (\phi_1, \phi_2, .., \phi_n)$, stetig und $\phi(x) \in D$ für $x \in D$. Dann hat ϕ einen Fixpunkt in D. Falls alle partielle Ableitungen von ϕ stetig sind und es eine Konstante $L < 1$ mit*

$$\left|\frac{\partial \phi_i}{\partial x_j}\right| \leq \frac{L}{n}, \qquad x \in D, \qquad i, j = 1,..., n$$

gibt, so gilt:
Die Fixpunktiteration $x_{k+1} = \phi(x_k)$ konvergiert für jeden Startwert $x_0 \in D$ gegen x^* mit $x^* = \phi(x^*)$. Weiters gilt:

$$\|x_k - x^*\|_\infty \leq \frac{L^k}{1 - L}\|x_1 - x_0\|_\infty$$

Beispiel 8.4.

$$F(x) = \begin{pmatrix} x_1^2 - 10x_1 + x_2^2 + 8 \\ x_1 x_2^2 + x_1 - 10x_2 + 8 \end{pmatrix}, \quad \textit{gesucht ist die Lösung von } F(x) = 0$$

Das ist äquivalent zur Fixpunktgleichung $x = \phi(x)$, wobei

$$x_1 = \phi_1(x_1, x_2) = \frac{x_1^2 + x_2^2 + 8}{10} \qquad x_2 = \phi_2(x_1, x_2) = \frac{x_1 x_2^2 + x_1 + 8}{10}$$

$$\frac{\partial \phi_1}{\partial x_1} = \frac{x_1}{5} \qquad \frac{\partial \phi_1}{\partial x_2} = \frac{x_2}{5} \qquad \frac{\partial \phi_2}{\partial x_1} = \frac{1 + x_2^2}{10} \qquad \frac{\partial \phi_2}{\partial x_2} = \frac{x_1 x_2}{5}$$

Wir setzen $D = \{(x_1, x_2) \in \mathbb{R}^2 \quad | \quad -\frac{3}{2} \leq x_i \leq \frac{3}{2} \quad i = 1, 2\}$. Dann gilt:

$$\left| \frac{\partial \phi_i}{\partial x_j} \right| \leq \frac{0.9}{2}$$

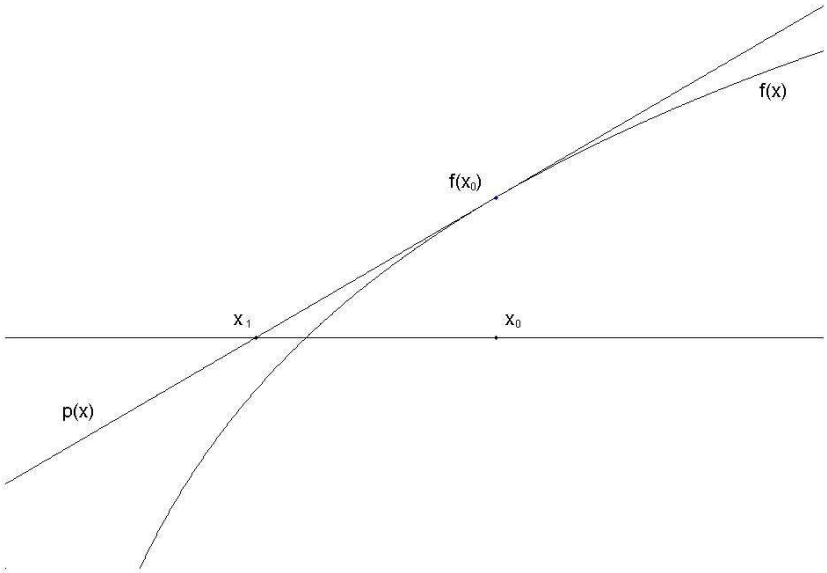
Wir setzen nun $x_0 = (0, 0)$ und berechnen die ersten Schritte der Fixpunktiteration $x_{k+1} = \phi(x_k)$:

k	0	1	2	3	4	5	...	∞
x_1	0	0.8	0.928	0.972	0.989	0.995	...	1
x_2	0	0.8	0.931	0.973	0.989	0.995	...	1
$\|x_k - x^*\|_\infty$	1	0.2	0.072	0.027	0.010	0.004	...	0

Die Lösung des Problems ist also: $x_1 = 1$ und $x_2 = 1$.

8.2 Newton-Verf

Man sieht dass bei der
liefert den Newton-Ans
$f(x_0) + f'(x_0)(x - x_0)$
x_0 und setzen $p(x) = 0$,



Daraus folgt unter der Voraussetzung $f'(x) \neq 0$ die Iteration

$$x_{k+1} = x_k - [f'(x_k)]^{-1} f(x_k)$$

Das ist eine spezielle Fixpunktiteration, mit $\phi(x) = x - \frac{f(x)}{f'(x)}$ gilt. Das Newton-Verfahren läßt sich also auf Systeme von NLG $F(x) = 0$ anwenden, das heißt:

$$x_{k+1} = x_k - J_F(x_k)^{-1} \cdot F(x_k)$$

oder äquivalent:

$$J_F(x_k) \cdot \Delta x_k = -F(x_k)$$
$$x_{k+1} = x_k + \Delta x_k$$

wobei $J_F(x_k)$ die Jakobimatrix von F an der Stelle x_k ist und vorausgesetzt wird, dass $\det J_F(x_k) \neq 0$.

SATZ 8.5. *Sei $D \subset \mathbb{R}^n$ offen und konvex und $F : D \to \mathbb{R}$ eine stetig differenzierbare Funktion mit invertierbarer Jacobimatrix $J_F(x)$ für alle $x \in D$. Es gelte für ein $\omega \geq 0$ die folgende Lipschitz-Bedingung:*

$$\|J_F(x)^{-1} \cdot (J_F(x + sv) - J_F(x)) \cdot v\| \leq s\omega\|v\|$$

für alle $s \in [0, 1]$, $x \in D$, $v \in \mathbb{R}^n$, so dass $x + v \in D$.
Ferner existiere eine Lösung $x^* \in D$ und ein Startwert $x_0 \in D$ derart, dass

$$\rho = \|x^* - x_0\| < \frac{2}{\omega} \quad \textit{und} \quad B_\rho(x^*) = \{x \in \mathbb{R}^n \quad | \quad \|x - x^*\| < \rho\} \subseteq D \quad (B_\rho(x^*) \ \textit{...„Kugel um } x^*\textit{“})$$

Dann bleibt die durch das Newton-Verfahren definierte Folge (x_k), $k > 0$, in der offenen Kugel $B_\rho(x^*)$ und konvergiert gegen x^*, d.h.

$$\|x_k - x^*\| < \rho, \quad \lim_{k\to\infty} x_k = x^*$$

Bemerkung 8.6. Die Konvergenzgeschwindigkeit ist:

$$\|x_{k+1} - x^*\| \leq \frac{\omega}{2}\|x_k - x^*\|^2$$

d.h. das Newton-Verfahren konvergiert **lokal quadratisch**.

Beispiel 8.7.

$$F(x) = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} x_1^2 - 10x_1 + x_2^2 + 8 \\ x_1x_2^2 + x_1 - 10x_2 + 8 \end{pmatrix}$$

Gesucht ist die Lösung von $F(x) = 0$.

$$\textit{Betrachten wir } J_F(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 - 10 & 2x_2 \\ x_2^2 + 1 & 2x_1x_2 - 10 \end{pmatrix}$$

Wir wählen den Startwert $x_0 = (0, 0)$ und führen die Fixpunktiteration $x_{k+1} = x_k + \Delta x_k$ durch, wobei wir Δx_k durch Lösen des Systems $J_F(x_k) \cdot \Delta x_k = -F(x_k)$ erhalten. Durch Nachrechnen kann folgendes Ergebnis verifiziert werden:

k	*0*	*1*	*2*	*3*	*4*
x_1	*0*	*0.8*	*0.991*	*0.999*	*1*
x_2	*0*	*0.88*	*0.991*	*0.999*	*1*
$\|x_k - x^*\|$	*1*	*0.2*	*0.008*	$3 \cdot 10^{-5}$	10^{-10}

(Vergleiche mit vorigem Beispiel!)

Die Fixpunktiteration und das Newtonverfahren benötigen einen guten Startwert. Dazu kann man das Gradienten-Verfahren verwenden: Sei

$$g(x) = \sum_{i=1}^{n} f_i(x)^2.$$

g hat ein Minimum (eine Nullstelle) bei x^* mit $F(x^*) = 0$. Der Gradient von g ist gegeben durch:

$$\nabla g(x) = \left(\frac{\partial g(x)}{\partial x_i}\right) \qquad i = 1, .., \text{n}$$

und die Iterationsschritte durch:

$$x_{k+1} = x_k - \alpha \nabla g(x_k).$$

Literaturverzeichnis

[1] Numerische Mathematik I (Eine algorithmisch orientierte Einführung) / Peter Deuflhard, Andreas Hohmann; De-Gruyter-Lehrbuch, Berlin 2002, ISBN 3-11-017182-1

[2] An Introduction to Numerical Analysis / Endre Süli, David F. Mayers; Cambride University Press, Cambridge 2003, ISBN 0-521-00794-1