

Einführung in die Numerik WS2018/19

Dozent: Prof. Dr. ANDREAS FISCHER

12. Dezember 2018

Inhaltsverzeichnis

I	Interpolation	2
1	Grundlagen	2
2	Interpolation durch Polynome	4
2.1	Existenz und Eindeutigkeit	4
2.2	NEWTON-Form des Interpolationspolynoms	5
2.3	Interpolationsfehler	6
3	Interpolation durch Polynomsplines	9
3.1	Polynomsplines	9
3.2	Interpolation durch kubische Polynomsplines	9
3.3	Interpolation mit kubischen C^2 -Splines	10
3.4	Eine Minimaleigenschaft kubischer C^2 -Interpolationssplines	13
3.5	Interpolationsfehler bei kubischer C^2 -Interpolation	14
II	numerische Integration (Quadratur)	15
1	Integration von Interpolationspolynomen	15
2	NEWTON-COTES-Formeln	16
3	spezielle NEWTON-COTES-Formeln	17
4	Zusammengesetzte NEWTON-COTES-Formeln	20
5	GAUSS'sche Quadraturformeln	21
III	direkte Verfahren für lineare Gleichungssysteme	23
1	GAUSS'scher Algorithmus für quadratische Systeme	23
1.1	Grundform des GAUSS'schen Algorithmus	23
1.2	Pivotisierung	26
1.3	LU-Faktorisierung	26
1.4	GAUSS'scher Algorithmus für trigonale Systeme	29
2	CHOLESKY-Faktorisierung für symmetrische positiv definite Matrizen	31
2.1	Existenz der CHOLESKY-Faktorisierung	31
2.2	Berechnung des CHOLESKY-Faktors	32
3	Lineare Quadratmittelprobleme	34
3.1	Die GAUSS'schen Normalgleichungen	35
3.2	Orthonormalisierungsverfahren nach HOUSEHOLDER	36
3.3	Anwendung in der Ausgleichsrechnung	40
4	Kondition linearer Gleichungssysteme	41
4.1	Normen	41
4.2	Störungslemma	42
4.3	Kondition	42
IV	Kondition von Aufgaben und Stabilität von Algorithmen	45

1	Maschinenzahlen und Rundungsfehler	45
2	Fehleranalyse	47
2.1	Die Kondition einer Aufgabe	47
2.2	Stabilität von Algorithmen	48
V	Newton-Verfahren zur Lösung nichtlinearer Gleichungssysteme	51
1	Das NEWTON-Verfahren	52
2	Gedämpftes NEWTON-Verfahren	55
VI	lineare Optimierung	59
1	Ecken und ihre Charakterisierung	59
2	Simplex-Verfahren	60
3	Die Tableauform des Simplex-Verfahrens	61
4	Revidiertes Simplex-Verfahren	62
5	Bestimmung einer ersten zulässigen Basislösung	63
	Anhang	65
	Index	65

Vorwort

Kapitel I

Interpolation

1. Grundlagen

Aufgabe:

Gegeben sind $n + 1$ Datenpaare $(x_0, f_0), \dots, (x_n, f_n)$, alles reelle Zahlen und paarweise verschieden.

Gesucht ist eine Funktion $F : \mathbb{R} \rightarrow \mathbb{R}$, die die Interpolationsbedingungen

$$F(x_0) = f_0, \dots, F(x_n) = f_n \quad (1)$$

genügt.

Definition (Stützstellen, Stützwerte)

Die x_0 bis x_n werden Stützstellen genannt.

Die f_0 bis f_n werden Stützwerte genannt.

Die oben gestellte Aufgabe wird zum Beispiel durch

$$F(x) = \begin{cases} 0 & x \notin \{x_0, \dots, x_n\} \\ f_i & x = x_i \end{cases}$$

gelöst. Weitere Möglichkeiten sind: Polygonzug, Treppenfunktion, Polynom, ...

- In welcher Menge von Funktionen soll F liegen?
- Gibt es im gewählten Funktionenraum für beliebige Datenpaare eine Funktion F , die den Interpolationsbedingungen genügt (eine solche Funktion heißt Interpolierende)?
- Ist die Interpolierende in diesem Raum eindeutig bestimmt?
- Welche weiteren Eigenschaften besitzt die Interpolierende, zum Beispiel hinsichtlich ihrer Krümmung oder der Approximation einer Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ mit $f_k = f(x_k)$ für $k = 0, \dots, n$?
- Wie sollte man die Stützstellen wählen, falls nicht vorgegeben?
- Wie lässt sich die Interpolierende effizient bestimmen, gegebenenfalls auch unter der Berücksichtigung, dass neue Datenpaare hinzukommen oder dass sich nur die Stützwerte ändern?

■ Beispiel 1.1

k	0	1	2	3	4	5
x_k in s	0	1	2	3	4	5
f_k in °C	80	85,8	86,4	93,6	98,3	99,1

Interpolation im

- Raum der stetigen stückweise affinen Funktionen
- Raum der Polynome höchstens 5. Grades
- Raum der Polynome höchstens 4. Grades (Interpolation im Allgemeinen nicht lösbar, Regression nötig)

2. Interpolation durch Polynome

Π_n bezeichne den Vektorraum der Polynome von Höchstgrad n mit der üblichen Addition und Skalarmultiplikation. Für jedes $p \in \Pi_n$ gibt es $a_0, \dots, a_n \in \mathbb{R}$, sodass

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (1)$$

und umgekehrt.

2.1. Existenz und Eindeutigkeit

Satz 2.1

Zu $n+1$ Datenpaaren $(x_0, f_0), \dots, (x_n, f_n)$ mit paarweise verschiedenen Stützstellen existiert genau ein Polynom $p \in \Pi_n$, dass die Interpolationsbedingung Gleichung (1) erfüllt.

Beweis. • Existenz: Sei $j \in \{0, \dots, n\}$ und $L_j : \mathbb{R} \rightarrow \mathbb{R}$ mit

$$L_j(x) := \prod_{\substack{i=0 \\ i \neq j}}^n \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)}$$

das LAGRANGE-Basispolynom vom Grad n . Offenbar gilt $L_j \in \Pi_n$ und

$$L_j(x_k) = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases} = \delta_{jk} \quad (2)$$

Definiert man $p : \mathbb{R} \rightarrow \mathbb{R}$ durch

$$p(x) := \sum_{j=0}^n f_j \cdot L_j(x) \quad (3)$$

so ist $p \in \Pi_n$ und außerdem erfüllt p wegen Gleichung (2) die Interpolationsbedingung Gleichung (1)

- Eindeutigkeit: Angenommen es gibt Interpolierende $p, \tilde{p} \in \Pi_n$ mit $p \neq \tilde{p}$. Dann folgt $p - \tilde{p} \in \Pi_n$ und $(p - \tilde{p})(x_k) = p(x_k) - \tilde{p}(x_k) = 0$ für $k = 0, \dots, n$. Also hat $(p - \tilde{p})$ mindestens $n + 1$ Nullstellen, hat aber Grad n . Das heißt, dass $(p - \tilde{p})$ das Nullpolynom sein muss. \square

Definition (Interpolationspolynom)

Das Polynom, dass die Interpolationsbedingung erfüllt, heißt Interpolationspolynom zu $(x_0, f_0), \dots, (x_n, f_n)$.

► Bemerkung 2.2

- Die Darstellung Gleichung (3) heißt LAGRANGE-Form des Interpolationspolynoms.
- Um mittels Gleichung (3) einen Funktionswert $p(x)$ zu berechnen, werden $\mathcal{O}(n^2)$ Operationen benötigt; bei gleichabständigen Stützstellen kann man diesen Aufwand auf $\mathcal{O}(n)$ verringern. Ändern sich die Stützwerte, kann man durch Wiederverwendung von den $L_j(x)$ das $p(x)$ in $\mathcal{O}(n)$ Operationen berechnen.
- Man kann zeigen, dass L_0 bis L_n eine Basis von Π_n bilden.

2.2. Newton-Form des Interpolationspolynoms

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0) \cdots (x - x_{n-1}) \quad (4)$$

mit Koeffizienten $c_0, \dots, c_n \in \mathbb{R}$. Die Berechnung des Koeffizienten c_j kann rekursiv durch Ausnutzen der Interpolationsbedingung Gleichung (1) erfolgen. Für c_0 erhält man

$$f_0 \stackrel{!}{=} p(x_0) = c_0$$

Seien c_0 bis c_{j-1} bereits ermittelt. Dann folgt:

$$f_j \stackrel{!}{=} p(x_j) = c_0 + \underbrace{\sum_{k=1}^{j-1} c_k (x_j - x_0) \cdots (x_j - x_{k-1})}_{\text{bekannt}} + c_j \underbrace{(x_j - x_0) \cdots (x_j - x_{j-1})}_{\text{bekannt}}$$

► Bemerkung 2.3

- Der Aufwand um die Koeffizienten c_0, \dots, c_n zu ermitteln ist $\mathcal{O}(n^2)$. Kommt ein Datenpaar hinzu, kann man Gleichung (4) um einen Summanden erweitern und mit $\mathcal{O}(n)$ Operationen c_{n+1} bestimmen.
- Sind die Koeffizienten c_0, \dots, c_n in Gleichung (4) bekannt, dann benötigt man zur Berechnung von $p(x)$ $\mathcal{O}(n)$ Operationen.
- Die Polynome $N_0, \dots, N_n : \mathbb{R} \rightarrow \mathbb{R}$ mit

$$N_0 = 1 \quad \text{und} \quad N_j = (x - x_0) \cdots (x - x_{j-1})$$

heißen NEWTON-Basispolynome und bilden eine Basis von Π_n .

Die Koeffizienten c_0, \dots, c_n ergeben sich wegen Gleichung (1) auch als Lösung des folgenden linearen Gleichungssystems:

$$\begin{pmatrix} 1 & & & & \\ 1 & (x_1 - x_0) & & & \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & (x_n - x_0) & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{i=0}^{n-1} (x_n - x_i) \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

Die Systemmatrix dieses linearen Gleichungssystems ist eine reguläre untere Dreiecksmatrix.

Zu effizienten Berechnung eines Funktionswertes $p(x)$ nach Gleichung (4) mit gegebenen Koeffizienten

c_0, \dots, c_n kann man das HORNER-Schema anwenden. Überlegung für $n = 3$.

$$\begin{aligned} p(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2) \\ &= c_0 + (x - x_0) \left[c_1 + (x - x_1) [c_2 + (x - x_2)c_3] \right] \end{aligned}$$

Für beliebiges n liefert das den folgenden Algorithmus:

■ **Algorithmus 2.4 (Horner-Schema für Newton-Form)**

Input: $n, x, c_0, \dots, c_n, x_0, \dots, x_n$

```

1  p = c_n
2  do j = n-1, 0, -1
3    p = c_j + (x - x_j) p
4  end do
```

2.3. Interpolationsfehler

Definition (Maximum-Norm)

Die Norm

$$\|g\|_\infty := \max_{x \in [a, b]} |g(x)| \quad \text{für } g \in C[a, b]$$

definiert die Maximum-Norm in $C[a, b]$.

Satz 2.5

Sei $f \in C[a, b]$. Dann existiert zu jedem $\varepsilon > 0$ ein Polynom p_ε mit $\|f - p_\varepsilon\| \leq \varepsilon$.

Also liegt die Menge aller Polynome (beliebig hohen Grades) direkt in $C[a, b]$.

Definition 2.6 (Stützstellensystem)

Stützstellensystem : $a \leq x_0^{(n)} < \dots < x_n^{(n)} \leq b$. Weiterhin bezeichne $p_n \in \Pi_n$ das zu den Datenpaaren $(x_k^{(n)}, f(x_k^{(n)}))$ gehörende eindeutig bestimmte Interpolationspolynom.

Satz 2.7 (Satz von Faber 1914)

Zu jedem Stützstellensystem gibt es $f \in C[a, b]$, sodass (p_n) nicht gleichmäßig gegen f konvergiert. $\|p_n - f\|_\infty \rightarrow 0$ bedeutet, dass (p_n) gleichmäßig gegen f konvergiert.

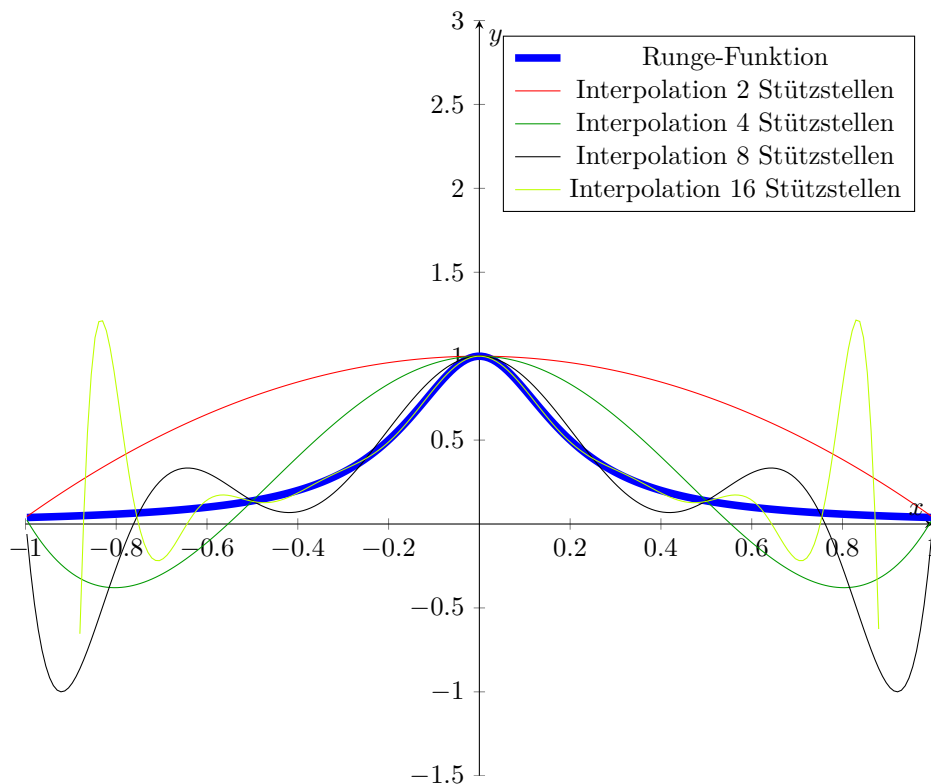
Nach einem Resultat von ERDÖS/VERTESI (1980) gilt sogar, dass $(p_n(x))$ fast überall divergiert.

■ **Beispiel 2.8 (Runge)**

$$f : \mathbb{R} \rightarrow \mathbb{R}, f(x) = \frac{1}{1+25x^2}$$

äquidistante Stützstellen $x_0, \dots, x_n, p \in \Pi_n$ als Interpolationspolynom

Stützstellen	interpoliertes Polynom
2	$1 - \frac{25x^2}{26}$
4	$3,31565x^4 - 4,27719x^2 + 1$
8	$53,6893x^8 - 102,815x^6 + 61,3672x^4 - 13,203x^2 + 1$
16	$15403,1x^{16} - 49713,5x^{14} + 63743,8x^{12} - 41870x^{10} + 15206x^8 - 3100,35x^6 + 351,984x^4 - 22,7759x^2 + 1$

**Anmerkung**

Wer mit Mathematica selber diese Polynome berechnen will, muss folgende Befehle benutzen:

- Funktion definieren: `f[x_]:=1/(1+25x^2)`
- Interpolationspolynome ausrechnen: `Expand[InterpolatingPolynomial[Table[{i,f[i]}],{i,-1,-1,Schrittweite}],{x}]`
- plotten: `Plot[f[x],InterpolatingPolynomial[Table[{i,f[i]}],{i,-1,-1,Schrittweite}],{x},{x,-1,1}]`

Satz 2.9

Sei $f \in C^{n+1}[a, b]$ und gelte $a \leq x_0 < \dots < x_n \leq b$. Mit $p_n \in \Pi_n$ werde das zu den Datenpaaren $(x_0, f(x_0)), \dots, (x_n, f(x_n))$ gehörende Interpolationspolynom bezeichnet. Dann existiert zu jedem $x \in [a, b]$ eine Zahl $\xi \in (a, b)$, so dass

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} w(x) \quad \text{für alle } x \in [a, b]$$

wobei $w(x) = (x - x_0) \cdot \dots \cdot (x - x_n)$

Beweis. Für $x = x_k$ mit $k = 0, \dots, n$ ist nicht zu zeigen, da p_n die Interpolationsbedingung erfüllt. Sei nun $x \in [a, b]$ fest gewählt mit $x \notin \{x_0, \dots, x_n\}$. Weiter seien

$$K = \frac{f(x) - p_n(x)}{w(x)} \quad \text{und} \quad F: \begin{cases} [a, b] \rightarrow \mathbb{R} \\ t \mapsto f(t) - p_n(t) - Kw(t) \end{cases}$$

Man stellt unter Beachtung der Interpolationsbedingung fest, dass $F(x_0) = F(x_1) = \dots = F(x_n) = 0$ und $F(x) = 0$. Also besitzt F mindestens $n+2$ paarweise verschiedene Nullstellen in $[a, b]$. Da $F \in C^{n+1}[a, b]$ erhält man durch $n+1$ -fache Anwendung des Satzes von Rolle, dass $F^{(n+1)}$ mindestens eine Nullstelle $\xi(x)$ in (a, b) besitzt. Also folgt

$$0 = F^{(n+1)}(\xi(x)) = f^{(n+1)}(\xi(x)) - \underbrace{p_n^{(n+1)}(\xi(x))}_{=0} - \underbrace{K w^{(n+1)}(\xi(x))}_{\text{Konstante}}$$

Da $w^{(n+1)} = (n+1)!$, erhält man

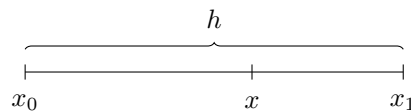
$$K = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}$$

Da $x \in [a, b]$ beliebig gewählt war, ist die Behauptung bewiesen. □

■ Beispiel 2.10

Sei $f \in C^2[a, b]$ mit $\|f\|_\infty \leq M$. Weiter sei $a = x_0 < x_1 = x_0 + h = b$. Mit Satz 2.9 folgt:

$$\begin{aligned} |f(x) - p_2(x)| &= \left| \frac{f''(\xi(x))}{2} (x - x_0)(x - x_1) \right| \\ &\leq \frac{1}{2} M \cdot \lambda(x) h \cdot (1 - \lambda(x)) h \\ &\leq \frac{1}{2} M \cdot h^2 \underbrace{\lambda(x)(1 - \lambda(x))}_{\leq 1/4} \\ &\leq \frac{1}{8} M \cdot h^2 \end{aligned}$$



$$\Rightarrow x = x_0 + \lambda \cdot (x_1 - x_0) = \lambda x_1 + (1 - \lambda)x_0$$

3. Interpolation durch Polynomsplines

3.1. Polynomsplines

Zur Abkürzung bezeichne Δ eine Zerlegung des Intervall $[a, b]$ durch die Stützstellen $a =: x_0 < \dots < x_n := b$.

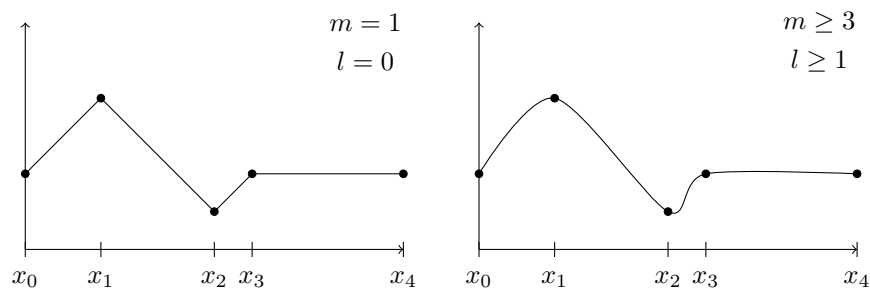
Definition 3.1 (Polynomspline)

Ein Polynomspline vom Grad $m \in \mathbb{N}$ und Glattheit $l \in \mathbb{N}$ zur Zerlegung Δ ist eine Funktion $s \in C^l[a, b]$ mit

$$s_k := s|_{[x_k, x_{k+1}]} \in \Pi_m \quad \text{für } k = 0, \dots, n-1$$

Dabei bezeichnet $s|_{[x_k, x_{k+1}]}$ die Einschränkung von s auf das Intervall $[x_k, x_{k+1}]$. Die Menge aller Splines wird mit $\mathcal{S}_m^l(\Delta)$ bezeichnet.

Folglich ist ein Polynomspline $s \in \mathcal{S}_m^l(\Delta)$ auf jedem der Teilintervall $[x_k, x_{k+1}]$ ein Polynom vom Höchstgrad m . Außerdem ist $s \in \mathcal{S}_m^l(\Delta)$ in allen Punkten $x \in [a, b]$ (also auch in den Stützstellen) l -mal stetig differenzierbar. $\mathcal{S}_m^l(\Delta)$ ist mit der üblichen Addition und Multiplikation ein Vektorraum. Speziell ist $\mathcal{S}_1^0(\Delta)$ die Menge aller stetigen stückweise affin linearen Funktionen.



3.2. Interpolation durch kubische Polynomsplines

Gegeben sei eine Zerlegung Δ und die Stützwerte f_0, \dots, f_n . Gesucht ist eine Funktion $s \in \mathcal{S}_3^l(\Delta)$ mit $l = 1, 2$ derart, dass

$$s(x_k) = f_k \quad \text{für } k = 0, \dots, n \quad (1)$$

Jede derartige Funktion heißt kubischer Interpolationspline.

Konstruktion eines solchen Splines:

$$\begin{aligned} h_k &:= x_{k+1} - x_k \quad \text{für } k = 0, \dots, n-1 \\ m_k &:= s'(x_k) \quad \text{für } k = 0, \dots, n-1 \end{aligned}$$

Wegen $l \in \{1, 2\}$ ist s zunächst stetig differenzierbar. Wegen $s_k = s|_{[x_k, x_{k+1}]}$ für $k = 0, \dots, n-1$ und $m = 3$ kann man folgenden Ansatz für s_k benutzen:

$$s_k(x) = a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k \quad (2)$$

Aus den Interpolationsbedingungen Gleichung (1) und der stetigen Differenzierbarkeit aller Funktionen in $s \in \mathcal{S}_m^l(\Delta)$ für $l \geq 1$ ergeben sich folgende Forderungen an s_k , $k = 0, \dots, n-1$:

$$\begin{aligned} s_k(x_k) &= f_k \quad \text{und} \quad s_k(x_{k+1}) = f_{k+1} \\ s'_k(x_k) &= m_k \quad \text{und} \quad s'_k(x_{k+1}) = m_{k+1} \end{aligned} \quad (3)$$

Diese liefern:

$$\begin{aligned} d_k &= s_k(x_k) = f_k \\ c_k &= s'_k(x_k) = m_k \end{aligned} \quad (4)$$

und damit:

$$\begin{aligned} s_k(x_{k+1}) &= a_k h_k^3 + b_k h_k^2 + m_k h_k + f_k = f_{k+1} \\ s'_k(x_{k+1}) &= 3a_k h_k^2 + 2b_k h_k + m_k = m_{k+1} \end{aligned}$$

Damit ergeben sich a_k und b_k als eindeutige Lösung für das lineare Gleichungssystem

$$\begin{pmatrix} h_k^3 & h_k^2 \\ 3h_k^2 & 2h_k \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} f_{k+1} - f_k - m_k h_k \\ m_{k+1} - m_k \end{pmatrix} \quad (5)$$

Die Determinante ist $-h_k^4 \neq 0$.

Satz 3.2

Sei eine Zerlegung Δ des Intervalls $[a, b]$ gegeben. Dann gibt es für beliebig gewählte reelle Zahlen f_0, \dots, f_n und m_0, \dots, m_n einen Interpolationsspline $s \in \mathcal{S}_3^1(\Delta)$, der den Interpolationsbedingungen

$$s'(x_0) = m_0, \dots, s'(x_n) = m_n$$

genügt. Außerdem gilt: $s|_{[x_k, x_{k+1}]} = s_k$ für $k=0, \dots, n-1$ mit s_k entsprechend Gleichung (2), wobei sich a_k, b_k, c_k, d_k aus Gleichung (4) und Gleichung (5) ergeben.

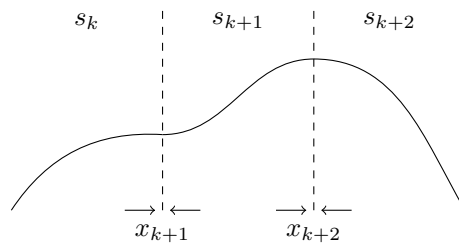
Für die Wahl der m_k gibt es verschiedene Möglichkeiten, zum Beispiel:

- Falls Ableitungswerte der zu interpolierenden Funktion f bekannt sind, kann man $m_k = f'(x_k)$ setzen.
- Man wählt m_0, \dots, m_n so, dass s zweimal stetig differenzierbar ist, das heißt $s \in \mathcal{S}_3^2(\Delta)$ statt $s \in \mathcal{S}_3^1(\Delta)$ gilt.

3.3. Interpolation mit kubischen C^2 -Splines

Damit ein kubischer Interpolationsspline s zu $\mathcal{S}_3^2(\Delta)$ gehört, muss neben den Forderungen in Gleichung (3) die Stetigkeit von s'' an den Stützstellen x_1, \dots, x_{n-1} gewährleistet sein. Also hat man zusätzliche Bedingungen

$$s''_k(x_{k+1}) = s''_{k+1}(x_{k+1}) \quad \text{für } k = 0, \dots, n-2$$



Mit Gleichung (2) ergibt sich $s''(x) = 6a_k(x - x_0) + 2b_k$ für $x \in [x_k, x_{k+1}]$ und damit $s''_k(x_{k+1}) = 6a_k h_k + 2b_k$ und $s''_{k+1}(x_{k+1}) = 2b_{k+1}$, also

$$3a_k h_k + b_k = b_{k+1} \quad \text{für } k = 0, \dots, n-2 \quad (6)$$

Aus Gleichung (5) folgt

$$\begin{aligned} a_k &= \frac{-2}{h_k^3}(f_{k+1} - f_k) + \frac{1}{h_k^2}(m_k + m_{k+1}) \\ b_k &= \frac{3}{h_k^2}(f_{k+1} - f_k) - \frac{1}{h_k}(2m_k + m_{k+1}) \end{aligned}$$

für $k = 0, \dots, n-1$. Wegen Gleichung (6) erhält man für $k = 0, \dots, n-2$

$$\begin{aligned} &\frac{-6}{h_k^2}(f_{k+1} - f_k) + \frac{3}{h_k}(m_k + m_{k+1}) + \frac{3}{h_k^2}(f_{k+1} - f_k) - \frac{1}{h_k}(2m_k + m_{k+1}) \\ &= \frac{-6}{h_{k+1}^2}(f_{k+2} - f_{k+1}) - \frac{1}{h_{k+1}}(2m_{k+1} + m_{k+2}) \end{aligned}$$

Damit folgt

$$\begin{aligned} \frac{1}{h_k}(m_k + 2m_{k+1}) + \frac{1}{h_{k+1}}(2m_{k+1} + m_{k+2}) &= \frac{3}{h_k^2}(f_{k+1} - f_k) + \frac{3}{h_{k+1}^2}(f_{k+2} - f_{k+1}) \\ \text{bzw. } h_{k+1}m_k + 2(h_{k+1} + h_k)m_{k+1} + h_k m_{k+2} &= \frac{3h_{k+1}}{h_k}(f_{k+1} - f_k) + \frac{3h_k}{h_{k+1}}(f_{k+2} - f_{k+1}) \end{aligned}$$

Also müssen die $n+1$ Zahlen m_0, \dots, m_n den $n-1$ Gleichungen des linearen Gleichungssystems

$$\begin{pmatrix} \lambda_0 & 2 & \mu_0 & & \\ & \lambda_1 & 2 & \mu_1 & \\ & & \ddots & \ddots & \ddots \\ & & & \lambda_{n-2} & 2 & \mu_{n-2} \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_n \end{pmatrix} = \begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n-2} \end{pmatrix}$$

genügen, wobei λ_k, μ_k, r_k durch

$$\begin{aligned} \lambda_k &= \frac{h_{k+1}}{h_k + h_{k+1}} \\ \mu_k &= \frac{h_k}{h_k + h_{k+1}} \\ r_k &= \frac{3h_{k+1}}{h_k(h_k + h_{k+1})}(f_{k+1} - f_k) + \frac{3h_k}{h_{k+1}(h_k + h_{k+1})}(f_{k+2} - f_{k+1}) \end{aligned}$$

für $k = 0, \dots, n-2$ gegeben sind. Die Systemmatrix und die erweiterte Systemmatrix haben den Rang $n-1$. Somit ist das Gleichungssystem lösbar, besitzt aber keine eindeutige Lösung. Um solche zu erhalten, kann man zusätzliche Bedingungen stellen, etwa

(a) **natürliche Randbedingungen:**

$$s''(x_0) = s''(x_n) = 0 \quad (7)$$

Diese sind gleichbedeutend mit

$$s_0''(x_0) = 6a_0(x - x_0) + 2b_0 = 0 \quad \text{und} \quad s_{n-1}''(x_n) = 6a_{n-1}(x_n - x_{n-1}) + 2b_{n-1} = 0$$

Also folgt

$$b_0 = 0 \quad \text{und} \quad 3a_{n-1}h_{n-1} + b_{n-1} = 0$$

Nutzt man noch die Darstellung für b_0 sowie für a_{n-1} und b_{n-1} , so folgt

$$2m_0 + m_1 = \frac{3}{h_0}(f_1 - f_0) \quad \text{und} \quad m_{n-1} + 2m_n = \frac{3}{h_{n-1}}(f_n - f_{n-1})$$

Fügt man beide Gleichungen geeignet zum obigen System hinzu, erhält man ein lineares Gleichungssystem mit einer regulären trigonalen Systemmatrix. Dieses kann in $\mathcal{O}(n)$ Operationen gelöst werden.

(b) **Vollständige Randbedingungen:** Sind $f'(a)$ und $f'(b)$ bekannt, dann können die zusätzlichen Bedingungen

$$s'(x_0) = f'(a) \quad \text{und} \quad s'(x_n) = f'(b) \quad (8)$$

mittels $m_0 = f'(a)$ und $m_n = f'(b)$ geeignet in das Gleichungssystem eingefügt werden, so dass man analog zu Fall (a) eine trigonale reguläre Systemmatrix erhält.

(c) **Periodische Spline-Interpolation:** Falls

$$f'(a) = f'(b) \quad (9)$$

und $f''(a) = f''(b)$ gilt, dann sind

$$s'(x_0) = s'(x_n) \quad \text{und} \quad s''(x_0) = s''(x_n) \quad (10)$$

sinnvolle Randbedingungen, woraus sich zwei zusätzliche lineare Gleichungen zur Ergänzung des Gleichungssystems ableiten lassen.

(d) (nicht in der Vorlesung) **Not-in-knot Bedingung:** Es soll zusätzlich

$$s_0'''(x_1) = s_1'''(x_1) \quad \text{und} \quad s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1})$$

gelten, das heißt s ist auf $[x_0, x_2]$ und auf $[x_{n-2}, x_n]$ ein Polynom dritten Grades. Man erhält daraus die Forderungen $a_0 = a_1$ und $a_{n-2} = a_{n-1}$, woraus sich zusätzliche Gleichungen in den

Variablen m_0, m_1, m_2 und m_{n-2}, m_{n-1}, m_n ergeben.

3.4. Eine Minimaleigenschaft kubischer C^2 -Interpolationssplines

Durch

$$\langle f, g \rangle := \int_a^b f(x)g(x)dx \quad \text{bzw.} \quad \|g\|_2 := \sqrt{\int_a^b g(x)^2 dx} \quad \text{für } f, g \in L^2[a, b]$$

ist ein Skalarprodukt bzw. eine Norm in $L^2[a, b]$ definiert.

Satz 3.3

Seien $f \in C^2[a, b]$, Δ eine Zerlegung von $[a, b]$ und $f_k := f(x_k)$ für $k = 0, \dots, n$. Für einen Interpolationsspline $s \in \mathcal{S}_3^2(\Delta)$, der die natürlichen, vollständigen oder periodischen Randbedingungen (bei letzteren gelte Gleichung (9)) erfüllt, gilt:

$$\|s''\|_2^2 = \|f''\|_2^2 - \|f'' - s''\|_2^2 \leq \|f''\|_2^2$$

Beweis. Durch Nachrechnen sieht man

$$\int_a^b (f''(x))^2 dx - \int_a^b (f''(x) - s''(x))^2 dx = \int_a^b (s''(x))^2 dx + 2 \int_a^b [f''(x) - s''(x)] s''(x) dx$$

Es wird nun $J := \int_a^b [f''(x) - s''(x)] s''(x) dx = 0$ gezeigt. Mit Hilfe partieller Integration folgt

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

wobei s''' auf jedem Teilintervall $[x_k, x_{k+1}]$ konstant ist. Dies ergibt wegen Gleichung (1)

$$\begin{aligned} \int_a^b [f'(x) - s'(x)] s'''(x) dx &= \sum_{k=0}^{n-1} s''' \left(x_k + \frac{h_k}{2} \right) \int_{x_k}^{x_{k+1}} f'(x) - s'(x) dx \\ &= \sum_{k=0}^{n-1} s''' \left(x_k + \frac{h_k}{2} \right) ([f(x_{k+1}) - s(x_{k+1})] - [f(x_k) - s(x_k)]) \\ &= 0 \end{aligned}$$

und damit

$$J = [f'(x) - s'(x)] s''(x) \Big|_a^b = [f'(b) - s'(b)] s''(b) - [f'(a) - s'(a)] s''(a)$$

Nutzt man nun noch Gleichung (7), Gleichung (8) bzw. Gleichung (9) mit Gleichung (10), so folgt $J = 0$. \square

Anmerkung

- Gleichung (7): natürliche Randbedingungen: $s''(a) = s''(b) = 0$
- Gleichung (8): vollständige Randbedingungen: $s(a') = f'(a)$, $s'(b) = f'(b)$
- Gleichung (9) und Gleichung (10): periodische Randbedingungen: $s'(a) = s'(b)$, $s''(a) = s''(b)$, $f'(a) = f'(b)$

3.5. Interpolationsfehler bei kubischer C^2 -Interpolation

Anmerkung

Die CAUCHY-SCHWARZ-Ungleichung hat folgende Form:

$$|\langle f, g \rangle| \leq \|f\|_2 \cdot \|g\|_2$$

Satz 3.4

Seien $f \in C^2[a, b]$, Δ eine Zerlegung von $[a, b]$ und $f_k := f(x_k)$ für $k = 0, \dots, n$. Für einen Interpolationsspline $s \in \mathcal{S}_3^2(\Delta)$, der die natürlichen, vollständigen oder periodischen Randbedingungen (bei letzteren gelte Gleichung (9)) erfüllt, gilt:

$$\|f - s\|_\infty \leq \frac{1}{2} h^{3/2} \|f''\|_2$$

wobei $h := \max\{h_0, \dots, h_{n-1}\}$.

Beweis. Die Funktion $r := f - s$ hat wegen Gleichung (1) die $n+1$ Nullstellen x_0, \dots, x_n . Der maximale Abstand benachbarter Nullstellen ist h . Nach dem Satz von Rolle besitzt r' mindestens n Nullstellen. Der Abstand zweier Nullstellen von r' ist durch $2h$ nach oben beschränkt. Sei $z \in [a, b]$ so gewählt, dass $|r'(z)| = \|r'\|_\infty$. Dann gilt $|z - z^0| \leq h$ für die z am nächsten liegende Nullstelle z^0 von r' . O.B.d.A. sei $z^0 \leq z$. Mit der CAUCHY-SCHWARZ-Ungleichung folgt:

$$\begin{aligned} \|r'\|_\infty^2 &= |r'(z) - \underbrace{r'(z^0)}_{z^0 \text{ NST}}|^2 \\ &\stackrel{*}{=} \left| \int_{z^0}^z r''(x) \cdot 1 dx \right|^2 \\ &\stackrel{\text{CS}}{\leq} \int_{z^0}^z r''(x)^2 dx \cdot \underbrace{\int_{z^0}^z 1^2 dx}_{= z - z^0 \leq h} \\ &\stackrel{\text{UG}}{\leq} h \|r''\|_2^2 \end{aligned} \tag{11}$$

*: Anwendung des Hauptsatzes der Integralrechnung

Sei nun $y \in [a, b]$ so gewählt, dass $|r(y)| = \|r\|_\infty$. Dann gilt $|y - y_0| \leq h/2$ für die y am nächsten liegende Nullstelle y_0 von r . O.B.d.A. sei $y_0 \leq y$. Mit Gleichung (11) ergibt sich

$$\|r\|_\infty = |r(y) - r(y_0)| = \left| \int_{y_0}^y r'(x) dx \right| \leq \max |r'(x)| \cdot \int_{y_0}^y dx \leq \frac{1}{2} \|r'\|_\infty \leq \frac{1}{2} h^{3/2} \|r''\|_2$$

Mit Satz 3.3 hat man $\|r''\|_2 \leq \|f\|_2$ und damit die Behauptung. \square

► Bemerkung 3.5

Besitzt f eine höhere Glattheit, so kann die obige Fehlerschranke bezüglich der h -Potenz verbessert werden. Es lassen sich ferner Abschätzungen für $\|f' - s'\|_\infty$ und $\|f'' - s''\|_\infty$ herleiten.

Kapitel II

numerische Integration (Quadratur)

1. Integration von Interpolationspolynomen

Für eine Funktion $f \in C[a, b]$ ist eine Näherung für den Wert des bestimmten Integrals

$$J(f) := \int_a^b f(x) dx$$

gesucht. Seien $a \leq x_0 < \dots < x_n \leq b$ Stützstellen und $f_k = f(x_k)$ für $k = 0, \dots, n$. Weiter bezeichne $p_n \in \Pi_n$ das zugehörige Interpolationspolynom. Dann kann man

$$Q_n(f) := J(p_n) = \int_a^b p_n(x) dx$$

als Näherung für $J(f)$ verwenden. Mit der LAGRANGE-Form des Interpolationspolynoms sieht man, dass

$$Q_n(f) = \int_a^b \sum_{k=0}^n f_k \cdot L_k(x) dx = \sum_{k=0}^n f_k \cdot \int_a^b L_k(x) dx$$

das heißt die Quadraturformel $Q_n(f)$ ist die gewichtete Summe von Funktionswerten der Funktion f mit den Gewichten $\int_a^b L_k(x) dx$.

2. Newton-Cotes-Formeln

Falls die Stützstellen gleichabständig sind mit $x_0 = a$ und $x_n = b$, das heißt

$$x_{k+1} = x_k + h \quad \text{für } k = 0, \dots, n-1 \quad (1)$$

mit der Schrittweite $h = \frac{b-a}{n}$ gilt, so nennt man $Q_n(f)$ geschlossene NEWTON-COTES-Formel. Ist $a < x_0$ und $x_n < b$ und gilt Gleichung (1) mit $h = \frac{b-a}{n+2}$, so bezeichnet man $Q_n(f)$ als offene NEWTON-COTES-Formel. Im Folgenden wollen wir uns auf den Fall geschlossener NEWTON-COTES-Formeln beschränken.

3. spezielle Newton-Cotes-Formeln

Für $n = 1$ erhält man die Trapezformel mit $x_0 = a$, $x_1 = b$ und $h = b - a$ wie folgt:

$$Q_1(f) = f_0 \int_a^b L_0(x) dx + f_1 \int_a^b L_1(x) dx$$

Mit

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{b - x}{h} \quad \text{und} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{x - a}{h}$$

$$\int_a^b L_0(x) dx = \frac{1}{h} \int_a^b (b - x) dx = \frac{1}{h} \left(bx - \frac{1}{2} x^2 \right) \Big|_a^b = \frac{1}{h} \left(\frac{b^2}{2} - ab + \frac{a^2}{2} \right) = \frac{h}{2}$$

$$\int_a^b L_1(x) dx = \frac{h}{2}$$

folgt

$$Q_1(f) = \frac{h}{2}(f_0 + f_1)$$

Für Polynomgrad $n = 2$ erhält man auf ähnliche Weise die SIMPSON-Formel (auch KEPLER'sche Fassregel genannt):

$$Q_2(f) = \frac{h}{3}(f_0 + 4f_1 + f_2)$$

Für Polynomgrade bis $n = 6$ findet man weitere Formeln in der Literatur. Formeln nur $n > 6$ werden aus numerischen Gründen nicht verwendet. Es können dann negative Gewichte auftreten.

Satz 3.1

(a) Sei $f \in C^2[a, b]$. Dann gilt:

$$|Q_1(f) - J(f)| \leq \frac{1}{12} h^3 \|f''\|_\infty$$

(b) Sei $f \in C^4[a, b]$. Dann gilt:

$$|Q_2(f) - J(f)| \leq \frac{1}{12} h^5 \|f^{(4)}\|_\infty$$

Beweis. (a) Für die Trapezformel erhält man mit Satz 2.9

$$|Q_1(f) - J(f)| = \left| \int_a^b f(x) - p_1(x) dx \right| \leq \frac{\|f''\|_\infty}{2!} \int_a^b |(x-a)(x-b)| dx$$

Mit $y := x - \frac{a+b}{2}$ ergibt sich:

$$x - a = y + \frac{b-a}{2} = y + \frac{h}{2} \quad \text{und} \quad x - b = y + \frac{a-b}{2} = y - \frac{h}{2}$$

$$\int_a^b |(x-a)(x-b)| dx = - \int_{-h/2}^{h/2} \left(y + \frac{h}{2}\right) \left(y - \frac{h}{2}\right) dy = - \left(\frac{1}{3}y^3 - \frac{1}{4}h^2y\right) \Big|_{-h/2}^{h/2} = \frac{1}{6}h^3$$

(b) Zur Analyse des Quadraturfehlers der SIMPSON-Formel untersuchen wir zunächst

$$W := \int_a^b (x-a)(x-x_1)(x-b) dx$$

$$\bar{W} := \int_a^b |(x-a)(x-x_1)(x-b)| dx$$

Mit der Substitution $y := x - x_1$ folgen $x - a = y + h$, $x - x_1 = y$ und $x - b = y - h$, also

$$W = \int_a^b (y+h)y(y-h) dy = 0 \quad (1)$$

da die zu integrierende Funktion $y \mapsto (y+h)y(y-h)$ ungerade ist. Weiter erhält man

$$\bar{W} = \int_{-h}^h |(y+h)y(y-h)| dy = -2 \int_0^h (y^2 - h^2)y dy = \frac{1}{2}h^4 \quad (2)$$

Wegen Satz 2.9 haben wir

$$f(x) - p_2(x) = \frac{f'''(\xi(x))}{3!} w(x) = \frac{1}{6} f'''(x_1) w(x) + \frac{1}{6} (f'''(\xi(x)) - f'''(x_1)) w(x)$$

für alle $x \in [a, b]$. Insbesondere ist daher $x \mapsto f'''(\xi(x))$ eine Funktion aus $C^4[a, b]$. Durch Integration folgt mit Gleichung (1):

$$\begin{aligned} \left| \int_a^b f(x) - p_2(x) dx \right| &= \frac{1}{6} \left| f'''(x_1) \int_a^b w(x) dx + \int_a^b (f'''(\xi(x)) - f'''(x_1)) w(x) dx \right| \\ &= \frac{1}{6} \left| \int_a^b (f'''(\xi(x)) - f'''(x_1)) w(x) dx \right| \\ &\leq \max_{x \in [a, b]} \left\{ |f'''(x) - f'''(x_1)| \cdot \int_a^b |w(x)| dx \right\} \end{aligned} \quad (3)$$

Da $f \in C^4[a, b]$ erhält man mit dem Mittelwertsatz

$$|f'''(x) - f'''(x_1)| = |f^{(4)}(\zeta(x))| \cdot |x - x_1| \leq h \|f^{(4)}\|_\infty$$

mit $\zeta \in (a, b)$. Deshalb und wegen Gleichung (2) folgt aus Gleichung (3) weiter

$$|Q_2(f) - J(f)| = \left| \int_a^b f(x) - p_2(x) dx \right| \leq \frac{1}{12} h^5 \|f^{(4)}\|_\infty \quad \square$$

► Bemerkung 3.2

Durch verfeinerte Abschätzungen kann man in Satz 3.1 b)

$$|Q_2(f) - J(f)| \leq \frac{1}{90} h^5 \|f^{(4)}\|_\infty \quad (4)$$

erreichen. Auch für Quadraturformeln $Q_n(f)$ mit $n > 2$ lassen sich entsprechende Ergebnisse für den Quadraturfehler herleiten, insbesondere hat der Quadraturfehler für $n = 3$ mit $f \in C^4[a, b]$ die Ordnung h^5 und für $n = 4$ mit $f \in C^6[a, b]$ die Ordnung h^7 .

4. Zusammengesetzte Newton-Cotes-Formeln

Um den Quadraturfehler weiter zu reduzieren, bietet es sich unter Berücksichtigung der Abschätzung des Quadraturfehlers in Satz 3.1 an, das Intervall $[a, b]$ in r Teilintervalle zu zerlegen und auf jedem der Teilintervalle dieselbe Quadraturformel (niedriger Ordnung) anzuwenden. Dazu wird das Intervall $[a, b]$ in $l = rn$ Elementarintervalle gleicher Länge zerlegt, wobei n die Ordnung des auf jedem Teilintervall zu verwendenden Interpolationspolynoms ist. Mit f_0, \dots, f_l werden die Funktionswerte an den Stellen $x_k = a + kh$ für $k = 0, \dots, l$ bezeichnet, wobei $h = \frac{b-a}{l}$ die Länge des Elementarintervalls ist. Die zusammengesetzte Trapezformel ist dann gegeben durch:

$$T_h(f) = \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \dots + 2f_{l-1} + f_l)$$

die zusammengesetzte SIMPSON-Formel durch

$$S_h(f) = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \dots + 2f_{l-2} + 4f_{l-1} + f_l)$$

Satz 4.1

(a) Für $f \in C^2[a, b]$ gilt

$$|T_h(f) - J(f)| \leq \frac{b-a}{12} h^2 \|f''\|_\infty$$

(b) Für $f \in C^4[a, b]$ gilt

$$|S_h(f) - J(f)| \leq \frac{b-a}{180} h^4 \|f^{(4)}\|_\infty$$

Beweis. Wendet man Satz 3.1 auf die SIMPSON-Formel (unter Beachtung von Gleichung (4)) für $[x_k, x_{k+2}]$ anstelle von $[a, b]$ an, so folgt

$$|S_h(f) - J(f)| = \left| S_h(f) - \sum_{k=0}^{r-1} \int_{x_{2k}}^{x_{2k+2}} f(x) dx \right| \leq \frac{1}{90} r h^5 \|f^{(4)}\|_\infty \leq \frac{b-a}{2 \cdot 90} h^4 \|f^{(4)}\|_\infty$$

und damit Behauptung b). Behauptung a) zeigt man auf ähnliche Weise. \square

5. Gauss'sche Quadraturformeln

Wir gehen zunächst vom Interpolationsfehler (vgl. Satz 2.9)

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} w_n(x) \quad \text{für } x \in [a, b]$$

mit $w_n(x) = (x - x_0) \dots (x - x_n)$ aus. Bezogen auf das ganze Intervall $[a, b]$, kann man etwa $\|f - p_n\|_\infty$ oder $\|f - p_n\|_2$ als Maß für diesen Fehler verwenden. Da man über $f^{(n+1)}$ nicht verfügt, wird anstelle dessen $\|w_n\|_\infty$ oder $\|w_n\|_2$ untersucht. Dieses Fehlermaß ist offenbar nur von der Lage der Stützstellen $x_0, \dots, x_n \in [a, b]$ abhängig. Zur Vereinfachung beschränkt man sich zunächst auf das Intervall $[-1, 1]$. Die Aufgabe, die Funktion

$$F_\infty : \begin{cases} \mathbb{R}^{n+1} & \rightarrow \mathbb{R} \\ F_\infty(x_0, \dots, x_n) & \mapsto \max_{x \in [-1, 1]} |(x - x_0) \dots (x - x_n)| \end{cases}$$

unter der Bedingung $(x_0, \dots, x_n) \in [-1, 1]^{n+1}$ zu minimieren, hat als Lösung die Nullstellen des sogenannten TSCHEBYSCHOW-Polynoms T_{n+1} der Ordnung $n + 1$. Die Funktion

$$F_2 : \begin{cases} \mathbb{R}^{n+1} & \rightarrow \mathbb{R} \\ F_2(x_0, \dots, x_n) & \mapsto \int_{-1}^1 (x - x_0)^2 \dots (x - x_n)^2 dx \end{cases}$$

wird unter der Bedingung $(x_0, \dots, x_n) \in [-1, 1]^{n+1}$ durch die Nullstellen des sogenannten LEGENDRE-Polynoms P_{n+1} minimiert. Die LEGENDRE-Polynome können rekursiv wie folgt definiert werden:

$$\begin{aligned} P_0(t) &= 1 \\ P_1(t) &= t \\ &\vdots \\ (k+1)P_{k+1}(t) &= (2k+1)tP_k(t) - kP_{k-1}(t) \end{aligned}$$

für alle $t \in \mathbb{R}$ und $k = 1, 2, \dots$. Für $n = 1$ erhält man zum Beispiel $P_2(t) = t^2 - \frac{1}{3}$ mit den Nullstellen $x_{1/2} = \pm \frac{\sqrt{3}}{3}$. Das Interpolationspolynom zu diesen Stützstellen und den Stützwerten $f_0 = f(x_0)$ sowie $f_1 = f(x_1)$ lautet dann (in der LAGRANGE-Form)

$$q_1(x) = f_0 \frac{x - x_1}{x_0 - x_1} + f_1 \frac{x - x_0}{x_1 - x_0}$$

Wegen

$$\begin{aligned} \int_{-1}^1 \frac{x - x_1}{x_0 - x_1} dx &= -\frac{\sqrt{3}}{2} \int_{-1}^1 \left(x - \frac{\sqrt{3}}{3} \right) dx = 1 \\ \int_{-1}^1 \frac{x - x_0}{x_1 - x_0} dx &= 1 \end{aligned}$$

hat man die GAUSS-LEGENDRE-Quadraturformel für das Integral $\int_{-1}^1 f(x)dx$ für $n = 1$:

$$f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

Man kann zeigen, dass die GAUSS-LEGENDRE-Quadraturformel mit $n+1$ Stützstellen (also den Nullstellen von P_{n+1}) Polynome bis zum Grad $2n+1$ exakt integriert. Bei den geschlossenen NEWTON-COTES-Formeln ist dies nur bis Grad n (falls n ungerade) bzw. bis zum Grad $n+1$ (falls n gerade) möglich (vgl. Übungsaufgabe). Spezielle Modifikationen der GAUSS-LEGENDRE-Quadratur beziehen einen Randpunkt (GAUSS-RADAU) oder beide Randpunkte (GAUSS-LOBATTO) des Integrals mit ein.

Falls über $[a, b]$ zu integrieren ist, führt eine Variablentransformation auf ein Integral über $[-1, 1]$ zum Ziel. Mit

$$y = \frac{2}{b-a} \left(x - \frac{a+b}{2} \right)$$

ergibt sich $x = \frac{b-a}{2}y + \frac{a+b}{2}$, $dx = \frac{b-a}{2}dy$ und

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}y + \frac{a+b}{2}\right) dy$$

Kapitel III

direkte Verfahren für lineare Gleichungssysteme

1. Gauss'scher Algorithmus für quadratische Systeme

1.1. Grundform des Gauss'schen Algorithmus

■ Beispiel 1.1

$$\begin{array}{rcl} 2x_1 - 2x_2 + 4x_3 & = & 10 \quad E_1 \\ x_1 + 3x_2 + 6x_3 & = & 25 \quad E_2 \\ -x_1 + 2x_2 + x_3 & = & 6 \quad E_3 \end{array}$$

$$E_1 \text{ behalten} \rightarrow E'_1, E_2 - \frac{1}{2}E_1 \rightarrow E'_2, E_3 + \frac{1}{2}E_1 \rightarrow E'_3$$

$$\begin{array}{rcl} 2x_1 - 2x_2 + 4x_3 & = & 10 \quad E'_1 \\ 4x_2 + 4x_3 & = & 20 \quad E'_2 \\ x_2 + 3x_3 & = & 11 \quad E'_3 \end{array}$$

$$E'_1 \text{ behalten} \rightarrow E''_1, E'_2 \text{ behalten} \rightarrow E''_2, E'_3 - \frac{1}{4}E'_2 \rightarrow E''_3$$

$$\begin{array}{rcl} 2x_1 - 2x_2 + 4x_3 & = & 10 \quad E''_1 \\ 4x_2 + 4x_3 & = & 20 \quad E''_2 \\ 2x_3 & = & 6 \quad E''_3 \end{array}$$

$$\Rightarrow x_3 = 3, x_2 = 2, x_1 = 1$$

Alle drei Systeme sind äquivalent, das heißt ihre Lösungsmengen sind gleich. Das letzte System wird Dreieckssystem oder System in Zeilenstufenform oder gestaffeltes System genannt.

Gegeben seien $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ und $b = (b_i) \in \mathbb{R}^n$. Gesucht ist, falls vorhanden, eine Lösung des linearen Gleichungssystems

$$\begin{array}{ccccccc} a_{11}x_1 & + & \dots & + & a_{1n}x_n & = & b_1 \\ \vdots & & & & \vdots & \vdots & \vdots \\ a_{n1}x_1 & + & \dots & + & a_{nn}x_n & = & b_n \end{array}$$

bzw. in Matrix-Schreibweise: $Ax = b$.

Prinzipielles Vorgehen

1. Vorwärtselimination (unter Voraussetzung der Durchführbarkeit): Schrittweise Transformation der erweiterten Koeffizientenmatrix

$$(A, b) = (A^{(1)}, b^{(1)}) \rightarrow (A^{(2)}, b^{(2)}) \rightarrow \dots \rightarrow (A^{(n)}, b^{(n)}) = (U, z)$$

wobei U eine obere Dreiecksmatrix ist. Der Eliminationsschritt $(A^{(k)}, b^{(k)}) \rightarrow (A^{(k+1)}, b^{(k+1)})$ für $k = 1, \dots, n-1$ verwendet die Eliminationsfaktoren

$$l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

um die i -te Zeile der neuen Matrix aus der alten Matrix zu bestimmen

$$\begin{aligned} \text{neue Zeile } i &= \text{alte Zeile } i & \text{für } i = 1, \dots, k \\ \text{neue Zeile } i &= \text{alte Zeile } i - l_{ik} \cdot \text{neue Zeile } k & \text{für } i = k+1, \dots, n \end{aligned}$$

2. Rücksubstitution (unter Voraussetzung der Durchführbarkeit): Lösung des Gleichungssystems $Ux = z$ nach x für gegebenes U, z

■ Algorithmus 1.2 (Vorwärtselimination)

Input: n, A, b

```

1  do k = 1, n-1
2  do i = k+1, n
3    lik = aik / akk
4    bi = bi - likbk
5  do j = k+1, n
6    aij = aij - likakj
7  end do
8  end do
9  end do

```

Output: (U, z) und l_{ik} für $i > k$. U steht in der oberen Hälfte von A mit Hauptdiagonale, b enthält z , die Zahlen l_{ik} lassen sich in der unteren Hälfte von A abspeichern.

■ Algorithmus 1.3 (Rücksubstitution)

Input: n, U, z

```

1  do i = n, 1, -1
2    s = 0
3    do j = i+1, n
4      s = s + uijxj

```

```

5   end do
6   end do

```

Output: x

Der Aufwand bei uneingeschränkter Durchführbarkeit von Algorithmus 1.2 ist $\sim \frac{2}{3}n^3$ und Algorithmus 1.3 ist $\sim n^2$

Durchführbarkeit

Der Algorithmus 1.2 ist genau dann durchführbar, wenn $a_{kk}^{(k)} \neq 0$ für alle $k = 1, \dots, n-1$ gilt. Gilt auch $a_{nn}^{(n)} \neq 0$, so folgt $u_{ii} \neq 0$ für $i = 1, \dots, n$ und damit die Durchführbarkeit von Algorithmus 1.3.

Definition 1.4 (streng diagonal dominant)

Eine Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ heißt streng diagonal dominant, wenn

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \quad \text{für alle } i = 0, \dots, n$$

Lemma 1.5

Ist die Matrix $A \in \mathbb{R}^{n \times n}$ streng diagonal dominant, so sind Algorithmus 1.2 und Algorithmus 1.3 durchführbar

Beweis (nicht in der Vorlesung). Die Matrix $A^{(1)}$ sei streng diagonal dominant. Weiter seien die Matrizen $A^{(k)}$ für ein $k \in \{1, \dots, n-1\}$ durch Vorwärtselimination erzeugt und streng diagonal dominant. Dies zieht $|a_{kk}^{(k)}| > 0$ nach sich, so dass die Erzeugung von $A^{(k+1)}$ durch Vorwärtselimination wohldefiniert ist. Es wird nun gezeigt, dass $A^{(k+1)}$ wieder streng diagonal dominant ist. Da $A^{(1)} = A$ als streng diagonal dominant vorausgesetzt wurde, folgt dann die Durchführbarkeit der gesamten Vorwärtselimination durch vollständige Induktion. Sei $i > k$ eine Zeile der Matrix $A^{(k+1)}$. Dann hat man

$$\begin{aligned}
 \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}^{(k+1)}| &= \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{ij}^{(k+1)}| = \sum_{\substack{j=k+1 \\ j \neq i}}^n \left| a_{ij}^{(k)} - \frac{a_{kj}^{(k)} a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \\
 &\leq \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{ij}^{(k)}| + \left| \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{kj}^{(k)}| \\
 &< |a_{ii}^{(k)}| - |a_{ik}^{(k)}| + \left| \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| (|a_{kk}^{(k)}| - |a_{ki}^{(k)}|) \\
 &= |a_{ii}^{(k)}| - \left| \frac{a_{ik}^{(k)} a_{ki}^{(k)}}{a_{kk}^{(k)}} \right| \\
 &\leq \left| a_{ii}^{(k)} - \frac{a_{ik}^{(k)} a_{ki}^{(k)}}{a_{kk}^{(k)}} \right| \\
 &= |a_{ii}^{(k+1)}|
 \end{aligned}$$

Falls $i \leq k$, so ändert sich $A^{(k+1)}$ gegenüber $A^{(k)}$ bezüglich der Zeile i nicht. Also ist $A^{(k+1)}$ streng diagonal dominant und man schließt auf die Durchführbarkeit der Vorwärtselimination für $k = 1, \dots, n-1$ und insbesondere auf $|a_{ii}^{(n)}| > 0$ für $i = 1, \dots, n$. Die Matrix $A^{(n)}$ enthält die Matrix U im oberen Dreieck, deren Diagonalelemente

sind gerade $a_{11}^{(n)}, \dots, a_{nn}^{(n)}$, also ist auch die Rücksubstitution wohldefiniert. \square

1.2. Pivotisierung

Die Regularität der Matrix $A \in \mathbb{R}^{n \times n}$ ist zwar äquivalent zur Lösbarkeit des linearen Gleichungssystems $Ax = b$, für jeden beliebigen Vektor $b \in \mathbb{R}^n$, jedoch sichert die Regularität nicht die Durchführbarkeit der Grundform des GAUSS'schen Algorithmus. Um die Durchführbarkeit bei regulärem A zu erzwingen, kann man eine Spaltenpivotisierung der Matrix durchführen. Dabei werden in jedem Durchlauf der Vorwärtselimination auf bestimmte Weise Zeilen der Matrix (A, b) vertauscht:

- Bestimme $p = p(k) \in \{k, \dots, n\}$, sodass $|a_{pk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|$.
 k -te Spalte von $A^{(k)}$ heißt Pivotspalte, $a_{pk}^{(k)}$ heißt Pivotelement, die Regularität von A sichert dann $a_{pk}^{(k)} \neq 0$.
- Vertausche die Zeilen p und k in der Matrix $(A^{(k)}, b^{(k)})$.
praktisch: Zeilentausch nicht ausführen, sondern einen Permutationsvektor mitführen.
formal: Beschreibung der Zeilen- und Spaltenvertauschungen durch Permutationsmatrizen. Dazu sei $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ eine Permutation und e_i bezeichne den i -ten kanonischen Einheitsvektor. Dann heißt $P_\pi = (e_{\pi(1)}, \dots, e_{\pi(n)})$ Permutationsmatrix.

Satz 1.6

Ist die Matrix A regulär, so ist der GAUSS'sche Algorithmus mit Spaltenpivotisierung (bei exakter Arithmetik) durchführbar.

Weitere Pivotisierungstechniken sind insbesondere die Zeilenpivotisierung (in Analogie zur Spaltenpivotisierung) und die vollständige Pivotisierung.

1.3. LU-Faktorisierung

Der k -te Schritt von Algorithmus 1.2 (ohne Pivotisierung) lässt sich schreiben als

$$\begin{aligned} A^{(k+1)} &= L_k A^{(k)} \\ b^{k+1} &= L_k b^{(k)} \end{aligned} \tag{1}$$

mit der GAUSS'schen Eliminationsmatrix

$$L_k = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & -l_{k+1,k} & 1 & \\ & & & \vdots & & \ddots \\ & & & -l_{n,k} & & 1 \end{pmatrix} \quad \begin{matrix} k \\ \\ \\ k\text{-te Zeile} \end{matrix}$$

Satz 1.7

Es gelten

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

$$L_1^{-1} \cdot \dots \cdot L_{n-1}^{-1} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{pmatrix} = L$$

Beweis. (a) Zunächst erkennt man, dass $L_k = \mathbb{1} - l_k e_k^T$, wobei $\mathbb{1} \in \mathbb{R}^{n \times n}$ die Einheitsmatrix und $e_k \in \mathbb{R}^n$ den k -ten kanonischen Einheitsvektor bezeichnen. Damit erhält man

$$\begin{aligned} L_k(\mathbb{1} + l_k e_k^T) &= (\mathbb{1} - l_k e_k^T)(\mathbb{1} + l_k e_k^T) \\ &= \mathbb{1} - l_k e_k^T + l_k e_k^T - l_k e_k^T l_k e_k^T \\ &\stackrel{(*)}{=} \mathbb{1} \end{aligned}$$

(*) $l_k e_k^T = 0$, da l_k erst an der $k+1$ -ten Stelle einen Wert hat, aber e_k nur an der k -ten Stelle eine 1 hat

(b) Es wird durch vollständige Induktion gezeigt, dass

$$L_1^{-1} \cdot \dots \cdot L_{n-1}^{-1} = \mathbb{1} + \sum_{i=1}^k l_i e_i^T \quad (2)$$

für $k = 1, \dots, n-1$ gilt. Daraus ergibt sich unmittelbar die zweite Aussage des Satzes. Für $k = 1$ folgt Gleichung (2) direkt aus Teil (a). Sei nun Gleichung (2) für ein $k < n-1$ erfüllt. Dann folgt mit $e_i^T l_{k+1} = 0$ für $i < k$, dass

$$\begin{aligned} L_1^{-1} \cdot \dots \cdot L_{k+1}^{-1} &= \left(\mathbb{1} + \sum_{i=1}^k l_i e_i^T \right) L_{k+1}^{-1} \\ &= \left(\mathbb{1} + \sum_{i=1}^k l_i e_i^T \right) (\mathbb{1} + l_{k+1} e_{k+1}^T) \\ &= \mathbb{1} + \sum_{i=1}^k l_i e_i^T + l_{k+1} e_{k+1}^T + \sum_{i=1}^k l_i e_i^T l_{k+1} e_{k+1}^T \\ &= \mathbb{1} + \sum_{i=1}^{k+1} l_i e_i^T \end{aligned} \quad \square$$

Aus $A^{(k+1)} = L_k A^{(k)}$ nach Gleichung (1) folgt

$$A = A^{(1)} = L_1^{-1} A^{(2)} = \dots = L_1^{-1} \cdot \dots \cdot L_{n-1}^{-1} A^{(n)} = L \cdot U$$

und analog $b = Lz$.

Satz 1.8

Seien $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ gegeben. Falls Algorithmus 1.2 ohne Pivotisierung durchführbar ist, dann gilt $A = LU$ mit der oberen Dreiecksmatrix $U = A^{(n)}$ und der unteren Dreiecksmatrix $L = (l_{ik})$ mit

$$l_{ik} = \begin{cases} 0 & i < k \\ 1 & i = k \\ l_{ik} & i > k \end{cases}$$

■ Algorithmus 1.9 (LU-Version der Grundform des Gauss'schen Algorithmus)

Input: A, b

```

1  compute L,U
2  solve Lz=b
3  solve Ux=z
```

Output: x, L, U

Der Aufwand an Rechenoperationen bei uneingeschränkter Durchführbarkeit: $\frac{2}{3}n^3 + 2n^2$.

Ein Vorteil der LU-Version besteht darin, dass man mit einer ermittelten LU-Faktorisierung von A das Gleichungssystem $Ax = b$ für mehrere rechte Seiten b mit dem Aufwand von je $2n^2$ lösen kann.

Satz 1.10

Sei $A \in \mathbb{R}^{n \times n}$ regulär. Dann gibt es eine durch Zeilenvertauschungen aus A hervorgegangene Matrix \tilde{A} , für die Algorithmus 1.2 ohne Pivotisierung durchführbar ist und $\tilde{A} = \tilde{L}\tilde{U}$.

Beweis. Nach Satz 1.6 ist der GAUSS'sche Algorithmus mit Spaltenpivotisierung durchführbar. Wendet man die dabei vorkommenden Zeilenvertauschungen vor Beginn von Algorithmus 1.2 auf A an, entsteht die Matrix \tilde{A} . Für $A = \tilde{A}$ liefert Satz 1.8 die Darstellung $\tilde{A} = \tilde{L}\tilde{U}$. Die Regularität von U folgt aus der Regularität von A mit dem Determinantenmultiplikationssatz. \square

1.4. Gauss'scher Algorithmus für trigonale Systeme

Sei $A \in \mathbb{R}^{n \times n}$ trigonal, das heißt

$$A = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_2 & \alpha_2 & \beta_2 & & \\ & \gamma_3 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \gamma_{n-1} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \gamma_n & \alpha_n \end{pmatrix} \quad (3)$$

Die Speicherung kann mittels geeigneter Vektoren, etwa

$$\alpha = (\alpha_1, \dots, \alpha_n)^T \quad \beta = (\beta_1, \dots, \beta_{n-1}, 0)^T \quad \gamma = (0, \gamma_2, \dots, \gamma_n)^T \quad (4)$$

erfolgen. Angenommen Algorithmus 1.2 ist ohne Pivotisierung durchführbar. Dann gibt es eine LU-Faktorisierung von A . Aus der Trigonalität von A und aus $A = LU$ ergibt sich die folgende Gestalt für L und U

$$L = \begin{pmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{pmatrix} \quad \text{und} \quad U = \begin{pmatrix} d_1 & \beta_1 & & & \\ & d_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & d_n \end{pmatrix} \quad (5)$$

mit $d_1 = \alpha_1$ und $l_k = \frac{\gamma_k}{d_{k-1}}$, $d_k = \alpha_k - l_k \beta_k$ für $k = 2, \dots, n$.

■ Algorithmus 1.11 (LU-Faktorisierung einer trigonalen Matrix ohne Pivotisierung)

Input: α, β, γ entsprechend Gleichung (3) und Gleichung (4)

```

1  d1 = α1
2  do k = 2, n
```



```

3    $l_k = \gamma_k / d_{k-1}$ 
4    $d_k = \alpha_k - l_k \beta_k$ 
5   end do

```

Output: $l = (0, l_2, \dots, l_n)^T$, $d = (d_1, \dots, d_n)^T$ für Gleichung (5)

► **Bemerkung 1.12**

Der Aufwand für Algorithmus 1.11 beträgt etwa $3n$ Operationen. Auch das Lösen der Dreieckssysteme $Lz = b$ und $Ux = z$ ist billig (je etwa $2n^2$). Bei Spaltenpivotisierung kommt in U im Allgemeinen eine zweite Nebendiagonale hinzu. Die dargestellte Verfahrensweise lässt sich auf Systeme mit Bandmatrizen erweitern, die mehr als je eine untere bzw. obere Nebendiagonale besitzen.

2. Cholesky-Faktorisierung für symmetrische positiv definite Matrizen

Definition 2.1 (positiv definit)

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt positiv definit, wenn

$$x^T A x > 0 \quad \forall x \in \mathbb{R}^n \setminus \{0\}$$

Bekannt sind folgende Zusammenhänge:

- Falls A symmetrisch ist, besitzt A nur reelle Eigenwerte.
- Sei A symmetrisch. Dann ist A genau dann positiv definit, wenn alle Eigenwerte von A positiv sind.
- Eine Matrix A ist genau dann positiv definit, wenn ihr symmetrischer Anteil $\frac{1}{2}(A + A^T)$ positiv definit ist.

2.1. Existenz der Cholesky-Faktorisierung

Satz 2.2

Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Dann existiert genau eine untere Dreiecksmatrix $L = (l_{ik})$ mit $l_{kk} > 0$, sodass

$$A = LL^T$$

Beweis. Mittels vollständiger Induktion. Für $n = 1$ gilt $A = (a_{11})$ mit $a_{11} > 0$ (wegen positiver Definitheit). Also gilt die Behauptung genau für $L = (l_{11})$ mit $l_{11} = \sqrt{a_{11}}$. Sei nun die Behauptung für $n - 1$ erfüllt. Die Matrix A kann man wegen ihrer Symmetrie mit geeigneten $A_{n-1} \in \mathbb{R}^{(n-1) \times (n-1)}$, $a \in \mathbb{R}^{n-1}$ und $a_{nn} \in \mathbb{R}$ schreiben als

$$A = \begin{pmatrix} A_{n-1} & a \\ a^T & a_{nn} \end{pmatrix} \quad (1)$$

Aus der positiven Definitheit von A folgt

$$0 < \begin{pmatrix} x \\ 0 \end{pmatrix}^T A \begin{pmatrix} x \\ 0 \end{pmatrix} = x^T A_{n-1} x \quad (2)$$

das heißt A_{n-1} ist positiv definit. Nach Induktionsvoraussetzung gibt es genau eine untere Dreiecksmatrix $L_{n-1} = (l_{ij}^{(n-1)})$ mit $l_{kk}^{(n-1)} > 0$ für $k = 1, \dots, n-1$. Um A als Produkt der Form $L_n L_n^T$ mit einer unteren Dreiecksmatrix $L_n \in \mathbb{R}^{n \times n}$ mit ausschließlich positiven Hauptdiagonalelementen darzustellen, ist der Ansatz

$$L_n = \begin{pmatrix} L_{n-1} & 0 \\ c^T & l_{nn} \end{pmatrix}$$

mit freien Parametern $c \in \mathbb{R}^{n-1}$ und $l_{nn} > 0$ die einzige Möglichkeit. Der Ansatz liefert

$$L_n L_n^T = \begin{pmatrix} L_{n-1} & 0 \\ c^T & l_{nn} \end{pmatrix} \begin{pmatrix} L_{n-1}^T & c \\ 0 & l_{nn} \end{pmatrix} = \begin{pmatrix} L_{n-1} L_{n-1}^T & L_{n-1} c \\ c^T L_{n-1}^T & c^T c + l_{nn}^2 \end{pmatrix} = \begin{pmatrix} A_{n-1} & a \\ a^T & a_{nn} \end{pmatrix}$$

Also müssen c und l_{nn} den Bedingungen $L_{n-1} c = a$ und $c^T c + l_{nn}^2 = a_{nn}$ genügen. Da L_{n-1} regulär ist, folgt

$$c = L_{n-1}^{-1} a \quad (3)$$

so dass c eindeutig bestimmt ist. Es wird nun gezeigt, dass $a_{nn} - c^T c > 0$ ist und damit $l_{nn}^2 > 0$ folgt. Somit ist dann l_{nn} in L_n durch die Bedingung $l_{nn} > 0$ eindeutig festgelegt auf $l_{nn} = \sqrt{a_{nn} - c^T c}$. Da A positiv definit ist und die Darstellung Gleichung (1) mit $A_{n-1} = L_{n-1} L_{n-1}^T$ nach Induktionsvoraussetzung angenommen werden kann, folgt

$$0 < (x^T, y) A \begin{pmatrix} x \\ y \end{pmatrix} = x^T L_{n-1} L_{n-1}^T x + 2y a^T x + y^2 a_{nn}$$

für alle $(x, y) \in \mathbb{R}^{n-1} \times \mathbb{R}$ mit $(x^T, y) \neq 0$. Setzt man speziell $(x, y) = (\hat{x}, \hat{y})$ mit

$$\hat{x} = (L_{n-1}^T)^{-1} L_{n-1}^{-1} a \quad \text{und} \quad \hat{y} = -1$$

so folgt mit Gleichung (3) das Gewünschte.

$$0 < a^T (L_{n-1}^T)^{-1} L_{n-1}^{-1} a - 2a^T (L_{n-1}^T)^{-1} L_{n-1}^{-1} a + a_{nn} = -c^T c + a_{nn} \quad \square$$

2.2. Berechnung des Cholesky-Faktors

Sei A wieder symmetrisch und positiv definit. Aus $A = LL^T$ folgt durch elementweises Vergleichen beider Matrizen

$$a_{ik} = (LL^T)_{ik} = \sum_{j=1}^l l_{ij} (L^T)_{kj} = \sum_{j=1}^k l_{ij} l_{kj}$$

Dies sind $\frac{n(n+1)}{2}$ Gleichungen für $\frac{n(n+1)}{2}$ Unbekannte l_{ik} , da A symmetrisch ist. Diese Gleichungen lassen sich durch spaltenweise Bestimmung der l_{ik} in L wie folgt lösen.

■ Algorithmus 2.3 (Cholesky-Faktorisierung)

Input: n , A symmetrisch und positiv definit

```

1  do k = 1, n
2    lkk =  $\sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$ 
3    do i = k+1, n
4      lik =  $\left( a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj} \right) / l_{kk}$ 
5    end do
6  end do
```

Output: l_{ik} für $1 \leq k \leq i \leq n$

► **Bemerkung 2.4**

Der Aufwand von Algorithmus 2.3 beträgt etwa $\frac{n^3}{3}$ Operationen. Der Algorithmus ist genau dann durchführbar, wenn A symmetrisch und positiv definit ist. Andernfalls entsteht im Verlauf des Verfahrens ein nicht-positiver Wert unter der Wurzel. Um das Ziehen der Quadratwurzel zu vermeiden, kann man anstelle von $A = LL^T$ die Umformung

$$A = LL^T = LD^{-1}D^2D^{-1}L^T$$

mit $D = \text{diag}(l_{11}, \dots, l_{nn})$ verwenden. Dann ist $\tilde{L} = LD^{-1}$ genau eine untere Dreiecksmatrix mit $l_{kk} = 1$ für $k = 1, \dots, n$ und $\tilde{D} = D^2$ eine Diagonalmatrix mit $\tilde{D} = \text{diag}(l_{11}^2, \dots, l_{nn}^2)$, sodass die sogenannte LDL-Faktorisierung $A = LL^T = \tilde{L}\tilde{D}\tilde{L}^T$ folgt.

■ **Algorithmus 2.5 (Wurzelfreie Cholesky-Faktorisierung)**

Input: n , A symmetrisch und positiv definit

```

1  do k = 1, n
2     $\widetilde{d}_{kk} = a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2 \widetilde{d}_{jj}$ 
3    do i = k+1, n
4       $\widetilde{l}_{ik} = \left( a_{ik} - \sum_{j=1}^{k-1} \widetilde{d}_{jj} l_{ij} l_{kj} \right) / \widetilde{d}_{kk}$ 
5    end do
6  end do
```

Output: $\widetilde{l}_{ik}, \widetilde{d}_{kk}$ für $1 \leq k \leq i \leq n$

Der Aufwand beträgt etwa $\frac{n^3}{3}$. Hat man die Faktorisierung $A = \tilde{L}\tilde{D}\tilde{L}^T$ bestimmt, so ist auch die LU-Faktorisierung von A bekannt: $A = \tilde{L}U$ mit $U = \tilde{D}\tilde{L}^T$.

3. Lineare Quadratmittelprobleme

Das lineare Gleichungssystem $Ax = b$ mit $A \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$ besitzt genau dann eine Lösung, wenn $\text{rang}(A) = \text{rang}((A, b))$. Falls $m > n$, so heißt das Gleichungssystem überbestimmt. Im Allgemeinen gilt dann $\text{rang}(A) \leq n < \text{rang}((A, b))$, so dass das Gleichungssystem keine Lösung besitzt. Der Fall der Nichtlösbarkeit kann auch für $m \leq n$ eintreten, falls $\text{rang}(A) < m$. Anstelle des Systems $Ax = b$ betrachtet man folgende Ersatzaufgabe:

$$\|Ax - b\|_2 \rightarrow \min \quad (1)$$

die als lineares Quadraturmittelproblem bezeichnet wird. Kurz schreibt man dafür auch $Ax \cong b$.

Satz 3.1

Seien $A \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$ gegeben. Da ist das lineare Quadraturmittelproblem Gleichung (1) lösbar.

Beweis. Die restringierte Optimierungsaufgabe

$$f(y) = \|y - b\|_2 \rightarrow \min \quad \text{mit } y \in L = \{Ax \mid x \in \mathbb{R}^n\} \quad (2)$$

ist offenbar genau dann lösbar, wenn Gleichung (1) eine Lösung besitzt. Wegen $f(0) = \|b\|_2$ und $0 \in L$ hat

$$f(y) \rightarrow \min \quad \text{mit } y \in L, \|y - b\|_2 \leq \|b\|_2 \quad (3)$$

dieselbe Lösungsmenge wie Gleichung (2). Der zulässige Bereich $\{x \in L \mid \|y - b\|_2 \leq \|b\|_2\}$ dieser Optimierungsaufgabe ist nicht-leer, abgeschlossen und beschränkt. Da zudem $f : \mathbb{R}^m \rightarrow \mathbb{R}$ stetig ist, besitzt Gleichung (3) nach dem Satz von WEIERSTRASS eine Lösung. Also sind auch Gleichung (2) und Gleichung (1) lösbar. \square

■ Beispiel 3.2

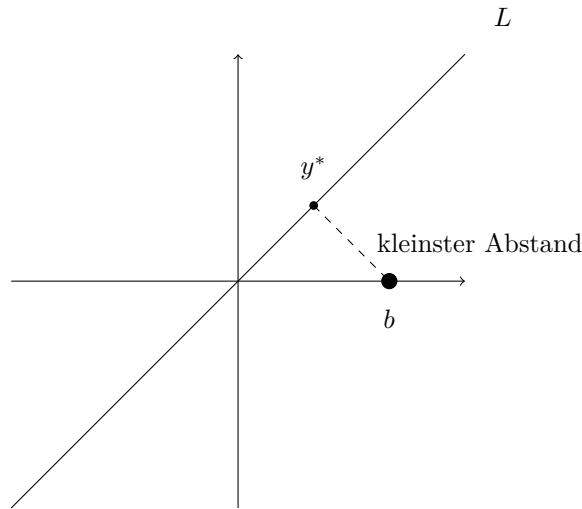
Seien

$$A = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Dann ist der lineare Teilraum L gegeben durch $L = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix} x \mid x \in \mathbb{R} \right\}$. Anschaulich ergibt sich, dass eine Lösung $y^* \in L$ von Gleichung (2) der Bedingung $(y^* - b) \perp L$ genügen muss. Daraus folgt

$$\left(\begin{pmatrix} y_1^* \\ y_2^* \end{pmatrix} - \begin{pmatrix} 2 \\ 0 \end{pmatrix} \right)^T \begin{pmatrix} 1 \\ 1 \end{pmatrix} x = 0 \quad \text{und} \quad \begin{pmatrix} y_1^* \\ y_2^* \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} x$$

für alle $x \in \mathbb{R}$. Einzige Lösung von Gleichung (2) ist damit $y^* = (1, 1)^T$. Somit ist $x^* = 1$ die einzige Lösung von Gleichung (1).



3.1. Die Gauss'schen Normalgleichungen

Satz 3.3

Seien $A \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$ gegeben. Dann gilt:

- (a) Jede Lösung des linearen Quadratmittelproblems Gleichung (1) löst die GAUSS'schen Normalgleichungen

$$A^T A x = A^T b \quad (4)$$

und umgekehrt.

- (b) Falls $\text{rang}(A) = n$ (dies impliziert $m \geq n$), so ist $A^T A$ positiv definit und Gleichung (1) besitzt genau eine Lösung, nämlich $x^* = (A^T A)^{-1} A^T b$.
- (c) Falls $\text{rang}(A) < n$, so ist $A^T A$ positiv semidefinit und singulär und Gleichung (1) besitzt unendlich viele Lösungen.

Beweis. (a) Die Zielfunktion $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ der zu Gleichung (1) äquivalenten Aufgabe

$$\varphi(x) = \frac{1}{2} \|Ax - b\|_2^2 \rightarrow \min \quad (5)$$

lässt sich schreiben als

$$\begin{aligned} \varphi(x) &= \frac{1}{2} (Ax - b)^T (Ax - b) \\ &= \frac{1}{2} (x^T A^T A x - 2b^T A x + b^T b) \end{aligned}$$

Die notwendige Optimalitätsbedingung für Gleichung (5) lautet $\nabla \varphi(x) = 0$, das heißt

$$A^T A x = A^T b$$

Also ist jede Lösung von Gleichung (1) auch eine Lösung der GAUSS'schen Normalgleichungen Gleichung (4). Da φ eine konvexe Funktion ist (wegen $\nabla^2 \varphi(x) = A^T A$ positiv semidefinit), ist Gleichung (4) zugleich eine hinreichende Optimalitätsbedingung, das heißt jede Lösung von Gleichung (4) löst Gleichung (1).

chung (1).

- (b) Sei $\text{rang}(A) = n$. Dann hat A vollen Spaltenrang und $Ax \neq 0$ für alle $x \neq 0$. Folglich gilt $x^T A^T Ax = (x^T A^T)(Ax) = \|Ax\|_2^2 > 0$ für alle $x \neq 0$. Also ist A positiv definit und damit regulär. Somit sind die GAUSS'schen Normalgleichungen Gleichung (4) eindeutig lösbar, ihre Lösung ist $x^* = (A^T A)^{-1} A^T b$. Wegen Teil (a) ist dies auch die einzige Lösung von Gleichung (1).
- (c) Sei $\text{rang}(A) < n$. Dann gibt es $\hat{x} \neq 0$ mit $A\hat{x} = 0$. Folglich ist einerseits A positiv semidefinit (denn $x^T A^T Ax = \|Ax\|_2^2 \geq 0$) aber andererseits $A^T A\hat{x} = 0$ und $A^T A$ daher singulär. Da nach Satz 3.1 das lineare Quadratmittelproblem Gleichung (1) eine Lösung besitzt, muss nach Teil (a) auch Gleichung (4) lösbar sein. Aufgrund der Singularität von $A^T A$ hat Gleichung (4) unendlich viele Lösungen. \square

Sei x^* eine Lösung von Gleichung (1). Dann gilt wegen Satz 3.3

$$0 = A^T Ax^* - A^T b = A^T (Ax^* - b)$$

Dies ist äquivalent zu folgenden Aussagen

- $0 = x^T A^T (Ax^* - b)$
- $(Ax^* - b) \perp Ax$
- $(Ax^* - b) \perp L$

■ Algorithmus 3.4 (Prinzip des Normalgleichungsverfahrens)

Input: $A \in \mathbb{R}^{m \times n}$ mit $\text{rang}(A) = n$, $b \in \mathbb{R}^m$

```

1  G = transpose(A) * A
2  c = transpose(A) * b
3  compute L ! als Cholesky-Faktor von G
4  solve Lz=c
5  solve transpose(L)x = z
```

Output: x, L

► Bemerkung 3.5

Der Aufwand beträgt etwa mn^2 Operationen zur Berechnung der unteren Hälfte von G , $\frac{n^3}{3}$ für die CHOLESKY-Faktorisierung sowie je n^2 für die Lösung der Dreieckssysteme. Offenbar ist der Aufwand für kleine n günstig. Nachteilig bezüglich numerischer Fehler kann sich beim Normalgleichungsverfahren die schlechte Kondition (siehe später) der Matrix $A^T A$ auswirken. Abhilfe schaffen geeignete Nachiterationen oder andere Verfahren (HOUSEHOLDER, SVD) zur Lösung des linearen Quadratmittelproblems.

3.2. Orthonormalisierungsverfahren nach Householder

Ziel ist zunächst die Beschreibung eines Verfahrens zur sogenannten QR -Faktorisierung einer Matrix $A \in \mathbb{R}^{m \times n}$, das heißt es sollen Matrizen $Q \in \mathbb{R}^{m \times m}$ und $R \in \mathbb{R}^{m \times n}$ bestimmt werden, so dass

$$A = QR$$

gilt, wobei Q eine orthogonale Matrix ($Q^{-1} = Q^T$) und R eine verallgemeinerte obere Dreiecksmatrix der Form

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \quad (\text{falls } m \geq n)$$

$$R = (R_1, R_2) \quad (\text{falls } m < n)$$

mit einer oberen Dreiecksmatrix $R_1 \in \mathbb{R}^{n \times n}$ bzw. einer oberen Dreiecksmatrix $R_1 \in \mathbb{R}^{m \times m}$ und einer Matrix $R_2 \in \mathbb{R}^{m \times (n-m)}$ ist. Später wird die QR -Zerlegung zur Lösung von Quadraturmittelproblemen (für den Fall $\text{rang}(A) = n$) eingesetzt.

Satz 3.6

Sei $w \in \mathbb{R}^m$ gegeben mit $w^T w = 1$. Dann ist die HOUSEHOLDER-Matrix

$$H = \mathbb{1} - 2ww^T$$

symmetrisch und orthogonal, das heißt es gilt $H = H^T = H^{-1}$.

Beweis. Offenbar gilt $H^T = \mathbb{1} - 2ww^T = H$. Weiter erhält man

$$H^T H = (\mathbb{1} - 2ww^T)(\mathbb{1} - 2ww^T) = \mathbb{1} - 4ww^T + 4w(w^T w)w^T = \mathbb{1} \quad \square$$

Die Wirkung einer HOUSEHOLDER-Matrix H auf einen Vektor $a \in \mathbb{R}^m$ (bei Multiplikation mit diesem Vektor) lässt sich wie folgt veranschaulichen. Zunächst hat man

$$Ha = (\mathbb{1} - 2ww^T)a = a - 2ww^T a = a - (2w^T a)w$$

Wegen $(a - w^T a w)^T w = a^T w - w^T a = 0$ (beachte $w^T w = 1$) liegt $a - w^T a w$ auf der Ebene $\mathcal{E} = \{y \in \mathbb{R}^m \mid y^T w = 0\}$ und Ha liegt bezüglich dieser Ebene (als Spiegelebene) spiegelbildlich zu a .

Satz 3.7

Es seien $a \in \mathbb{R}^m$ mit $a \notin \text{span}(e_1)$ und

$$w = \frac{a + pe_1}{\|a + pe_1\|_2} \quad \text{mit } p \in \{\|a\|_2, -\|a\|_2\}$$

gegeben. Dann gilt

$$Ha = -pe$$

Beweis. Wegen $a \notin \text{span}(e_1)$ folgt $a + pe_1 \neq 0$. Also ist w wohldefiniert mit $w^T w = 1$ und man erhält

$$Ha = (\mathbb{1} - 2ww^T)a = a - (2w^T a)w = a - 2 \frac{a^T(a + pe_1)}{\|a + pe_1\|_2} \frac{a + pe_1}{\|a + pe_1\|_2} \quad (6)$$

Da $p \in \{\|a\|_2, -\|a\|_2\}$, gilt

$$\|a + pe_1\|_2^2 = a^T a + 2pa^T e_1 + p^2 = 2a^T(a + pe_1)$$

Deshalb liefert Gleichung (6) $Ha = a - (a + pe_1) = -pe_1$. \square

Satz 3.7 wird für die schrittweise HOUSEHOLDER-Transformation einer Matrix $A \in \mathbb{R}^{n \times m}$ in eine verallgemeinerte obere Dreiecksmatrix ausgenutzt. Dazu sei $A^{(1)} = A$ eine Matrix von Rang n . Ohne Beschränkung der Allgemeinheit gelte $m > n$. Weiter sei $A^{(k)}$ für ein $k \in \{1, \dots, n+1\}$ in der Form

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & \dots & a_{1k}^{(k)} & \dots & a_{1n}^{(k)} \\ & \ddots & \vdots & & \vdots \\ & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & \vdots & & \vdots \\ & & a_{mk}^{(k)} & \dots & a_{mn}^{(k)} \end{pmatrix}$$

gegeben. Der Vektor $a^k = (0, \dots, 0a_{kk}^{(k)}, \dots, a_{mk}^{(k)}) \in \mathbb{R}^m$ übernimmt die Rolle von a . Er soll durch Multiplikation mit $H_k \in \mathbb{R}^{m \times m}$ auf $p_k e_k$ transformiert werden, wobei

$$\begin{aligned} H_k &= \mathbb{1} - 2w_k w_k^T \\ w_k &= \frac{a^k + p_k e_k}{\|a^k + p_k e_k\|_2} \\ p_k &\in \{\|a^k\|_2, -\|a^k\|_2\} \end{aligned}$$

Zur Vermeidung von Stellenauslöschung in $a^k + p_k e_k$ wird man

$$p_k = \begin{cases} \|a_k\|_2 & \text{falls } a_{kk}^{(k)} \geq 0 \\ -\|a_k\|_2 & \text{falls } a_{kk}^{(k)} < 0 \end{cases}$$

wählen. Die Operation $A^{(k)} \mapsto H_k A^{(k)}$ lässt die ersten $k-1$ Zeilen und Spalten der Matrix $A^{(k)}$ unverändert und es gilt:

$$H_k A^{(k)} = A^{(k+1)} = \begin{pmatrix} a_{11}^{(k)} & \dots & a_{1,k-1}^{(k)} & a_{1k}^{(k+1)} & a_{1,k+1}^{(k+1)} & \dots & a_{1n}^{(k+1)} \\ & \ddots & \vdots & \vdots & \vdots & & \vdots \\ & & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k+1)} & a_{k-1,k+1}^{(k+1)} & \dots & a_{k-1,n}^{(k+1)} \\ & & & a_{kk}^{(k+1)} & a_{k,k+1}^{(k+1)} & \dots & a_{kn}^{(k+1)} \\ & & & & a_{k+1,k+1}^{(k+1)} & \dots & a_{k+1,n}^{(k+1)} \\ & & & & a_{m,k+1}^{(k+1)} & \dots & a_{mn}^{(k+1)} \end{pmatrix}$$

speziell mit $a_{kk}^{(k+1)} = -p_k$. Dabei garantiert die Bedingung $\text{rang}(A^{(k)}) = n$ dasselbe für den Rang von $A^{(k+1)}$. Die Hintereinanderausführung von HOUSEHOLDER-TRANSFORMATIONEN liefert

$$R = A^{(n+1)} = H_n H_{n-1} \dots H_1 A \quad (7)$$

Dabei ist R eine verallgemeinerte obere Dreiecksmatrix. Wegen Satz 3.6 existiert

$$Q = (H_n \dots H_1)^{-1} \quad (8)$$

und es gilt

$$\begin{aligned} Q &= H_1^{-1} \dots H_n^{-1} = H_1 \dots H_n \\ Q^T Q &= (H_n^T \dots H_1^T)(H_1 \dots H_n) = \mathbb{1} \end{aligned}$$

das heißt Q ist orthogonal. Wegen Gleichung (7) und Gleichung (8) folgt schließlich noch $A = QR$.

Satz 3.8

Sei $A \in \mathbb{R}^{m \times n}$ mit $m \geq n = \text{rang}(A)$ gegeben. Dann gibt es eine orthogonale Matrix $Q \in \mathbb{R}^{m \times m}$ und eine verallgemeinerte Dreiecksmatrix

$$R = \begin{pmatrix} R_1 \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

mit einer regulären oberen Dreiecksmatrix $R_1 \in \mathbb{R}^{n \times n}$, so dass $A = QR$.

Satz 3.9

Seien $A \in \mathbb{R}^{m \times n}$ mit $m \geq n = \text{rang}(A)$ und $b \in \mathbb{R}^m$ gegeben. Weiter seien Matrizen Q und R bzw. R_1 aus der QR -Zerlegung von A bekannt und Vektoren $y_1 \in \mathbb{R}^n$ und $y_2 \in \mathbb{R}^{m-n}$ so gegeben, dass

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = Q^T b$$

gilt. Dann ist das lineare Quadraturmittelproblem Gleichung (1) äquivalent zum linearen Gleichungssystem

$$R_1 x = y_1$$

Beweis. Wegen $A = QR$ und $QQ^T = \mathbb{1}$ gilt

$$\|Ax - b\|_2^2 = \|QRx - QQ^T b\|_2^2 = \|Q(Rx - Q^T b)\|_2^2$$

Da $\|Qz\|_2^2 = z^T Q^T Q z = z^T z = \|z\|_2^2$ für beliebige $z \in \mathbb{R}^m$ ist, folgt

$$\|Ax - b\|_2^2 = \left\| \begin{pmatrix} R_1 x \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\|_2^2 = \|R_1 x - y_1\|_2^2 + \|y_2\|_2^2$$

Also nimmt $\|Ax - b\|_2^2$ sein Minimum genau dann an, wenn x das lineare Gleichungssystem $R_1 x = y_1$ löst. \square

3.3. Anwendung in der Ausgleichsrechnung

Gegeben seien Messpunkte $(t_i, y_i) \in \mathbb{R} \times \mathbb{R}$ für $i = 1, \dots, m$ mit $t_i \neq t_j$ für $i \neq j$. Weiter seien sogenannte Basisfunktionen $\varphi_j : \mathbb{R} \rightarrow \mathbb{R}$ und die Funktion $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ durch

$$f(t, x) = \sum_{j=1}^n x_j \varphi_j(t) \quad \text{für } (t, x) \in \mathbb{R} \times \mathbb{R}^n$$

gegeben. Gesucht ist ein Parametervektor $x^* = (x_1, \dots, x_n)^T$, so dass

$$f(t_i, x^*) \approx y_i \quad \text{für } i = 1, \dots, m$$

Eine Möglichkeit ein solches x^* zu bestimmen ist die Lösung des Optimierungsproblems (Ausgleichsproblems)

$$\sum_{i=1}^m (y_i - f(t_i, x))^2 \rightarrow \min \quad (9)$$

Mit

$$A = \begin{pmatrix} \varphi_1(t_1) & \dots & \varphi_n(t_1) \\ \vdots & & \vdots \\ \varphi_1(t_m) & \dots & \varphi_n(t_m) \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

gilt $r(x) = \|Ax - b\|_2^2$, man beachte $y_i - f(t_i, x) = y_i - \sum_{j=1}^n x_j \varphi_j(t_i) = y_i - A_i x$, wobei A_i die i -te Zeile von A bezeichnet. Also ist Gleichung (9) ein lineares Quadratmittelpunktproblem.

■ **Beispiel 3.10 (Ausgleichsgerade)**

Seien $m = 3$ und $n = 2$. Es seien $(t_1, y_1) = (0, 1)$, $(t_2, y_2) = (3, 8)$, $(t_3, y_3) = (4, 10)$ und $\varphi_1(t) = 1$, $\varphi_2(t) = t$ für $t \in \mathbb{R}$. Dann ist $f(t, x) = x_1 + tx_2$,

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 1 \\ 8 \\ 10 \end{pmatrix}$$

und das Ausgleichsproblem Gleichung (9) hat die Lösung $x^* = (1.0385\dots, 2.2692\dots)^T$.

4. Kondition linearer Gleichungssysteme

Seien $A \in \mathbb{R}^{n \times n}$ und $b \in \mathbb{R}^n$ mit $b \neq 0$ gegeben. Es stellt sich die Frage, wie sich die Fehler in A bzw. b auf die Lösung $x = A^{-1}b$ des linearen Gleichungssystems $Ax = b$ auswirken. Dazu seien $\Delta A \in \mathbb{R}^{n \times n}$ bzw. $\Delta b \in \mathbb{R}^n$ Störungen kleiner Norm, insbesondere soll $A + \Delta A$ noch regulär sein. Weiter sei

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

der absolute Fehler zwischen den Lösungen des gestörten und des ungestörten Gleichungssystems in Abhängigkeit von den Fehlern ΔA und Δb der Eingangsdaten A und b . Es wird nun eine obere Schranke für den relativen Fehler

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

in Abhängigkeit von den relativen Fehlern der Eingangsdaten $\frac{\|\Delta A\|}{\|A\|}$ und $\frac{\|\Delta b\|}{\|b\|}$ gesucht.

4.1. Normen

Satz 4.1

Sei $\|\cdot\| : \mathbb{R}^n \rightarrow [0, \infty)$ eine Vektornorm. Dann ist durch

$$\|A\|_* = \sup_{\substack{x \in \mathbb{R}^n \\ x \neq 0}} \frac{\|Ax\|}{\|x\|} \quad \forall A \in \mathbb{R}^{n \times n}$$

eine Matrixnorm $\|\cdot\|_* : \mathbb{R}^{n \times n} \rightarrow [0, \infty)$ definiert. Diese der Vektornorm zugeordnete Matrixnorm ist mit der Vektornorm verträglich, das heißt

$$\|Ax\| \leq \|A\|_* \|x\| \quad \forall A \in \mathbb{R}^{n \times n} \text{ und } b \in \mathbb{R}^n$$

submultiplikativ, das heißt

$$\|A \cdot B\|_* \leq \|A\|_* \cdot \|B\|_* \quad \forall A, B \in \mathbb{R}^{n \times n}$$

und es gilt $\|\mathbb{1}\|_* = 1$.

Beispiele für eine Vektornorm und eine zugeordnete Matrixnorm sind:

- Der Maximum-Norm $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$ ist die Zeilensummen-Norm $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{k=1}^n |a_{ik}|$ zugeordnet.
- Der Summen-Norm $\|x\|_1 = \sum_{i=1}^n |x_i|$ ist die Spaltensummen-Norm $\|A\|_1 = \max_{1 \leq k \leq n} \sum_{i=1}^n |a_{ik}|$ zugeordnet.
- Der euklidischen Norm $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$ ist die Spektralnorm $\|A\|_2 = \sqrt{\rho(A^T A)}$ zugeordnet, wobei $\rho(B) = \max\{|\lambda| \mid \lambda \text{ ist Eigenwert von } B\}$. Also ist $\|A\|_2^2$ gleich dem betragsgrößten Eigenwert von $A^T A$.

4.2. Störungslemma

Lemma 4.2 (von Neumann'sches Störungslemma)

Seien $\|\cdot\|$ eine Vektornorm im \mathbb{R}^n bzw. die zugeordnete Matrixnorm und $B \in \mathbb{R}^{n \times n}$ mit $\|B\| < 1$. Dann ist $\mathbb{1} + B$ regulär und es gilt

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

Beweis. Mit der Dreiecksungleichung folgt für jedes $x \in \mathbb{R}^n$

$$\begin{aligned} \|(\mathbb{1} + B)x\| &= \|x + Bx\| \\ &\geq \|x\| - \|Bx\| \\ &\geq \|x\| - \|B\|\|x\| \\ &= \|x\|(1 - \|B\|) \\ &> 0 \end{aligned} \tag{1}$$

Also gilt $(\mathbb{1} + B)x = 0$ genau dann, wenn $x = 0$. Somit ist $\mathbb{1} + B$ regulär. Aus Gleichung (1) hat man

$$\|y\| = \|(\mathbb{1} + B)(\mathbb{1} - B)^{-1}y\| \geq (1 - \|B\|)\|(\mathbb{1} + B)^{-1}y\|$$

und damit

$$\frac{\|(\mathbb{1} + B)^{-1}y\|}{\|y\|} \leq \frac{1}{1 - \|B\|}$$

für alle $y \in \mathbb{R}^n \setminus \{0\}$. Dies zieht unter Beachtung der Definition der zugeordneten Matrixnorm die zweite Behauptung des Lemmas nach sich. \square

4.3. Kondition

Satz 4.3

Es seien $A \in \mathbb{R}^{n \times n}$ regulär, $\Delta A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n \setminus \{0\}$, $\Delta b \in \mathbb{R}^n$ und $\|\cdot\|$ eine Vektornorm in \mathbb{R}^n bzw. die zugeordnete Matrixnorm. Falls $\|A^{-1}\| \cdot \|\Delta A\| < 1$, dann ist $A + \Delta A$ regulär und es gibt eindeutig bestimmte Vektoren $x, \Delta x \in \mathbb{R}^n$, so dass

$$Ax = b \quad \text{und} \quad (A + \Delta A)(x + \Delta x) = b + \Delta b \tag{2}$$

und

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A\| \cdot \|A^{-1}\|}{1 - \|\Delta A\| \|A^{-1}\|} \left\{ \frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right\} \tag{3}$$

Beweis. Mit $B = A^{-1}\Delta A$ ergibt sich

$$\begin{aligned} A + \Delta A &= A(\mathbb{1} + B) \\ \|B\| &= \|A^{-1}\Delta A\| \leq \|A^{-1}\| \|\Delta A\| < 1 \end{aligned}$$

Wegen Lemma 4.2 ist $\mathbb{1} + B = \mathbb{1} + A^{-1}\Delta A$ regulär und

$$\|(\mathbb{1} + B)^{-1}\| \leq \frac{1}{1 - \|B\|} \leq \frac{1}{1 - \|A^{-1}\|\|\Delta A\|}$$

Damit ist auch $A(\mathbb{1} + B) = A + \Delta A$ regulär, es gilt $x = A^{-1}b$ sowie $\Delta x = -x + (A + \Delta A)^{-1}(b + \Delta b)$ und

$$\|(A + \Delta A)^{-1}\| = \|(A(\mathbb{1} + B))^{-1}\| = \|(\mathbb{1} + B)^{-1}A^{-1}\| \leq \frac{1}{1 - \|A^{-1}\|\|\Delta A\|} \|A^{-1}\| \quad (4)$$

Aus Gleichung (2) erhält man durch Substitution und weiter wegen der schon gezeigten Regularität von $A + \Delta A$

$$\begin{aligned} (A + \Delta A)\Delta x &= \Delta b - \Delta Ax \\ \Delta x &= (A + \Delta A)^{-1}(b + \Delta b) \end{aligned}$$

Mit Gleichung (4) folgt

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

Wegen $\|Ax\| = \|b\|$ liefert dies

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\|}{1 - \|\Delta A\|\|A^{-1}\|} \left\{ \frac{\|\Delta b\|}{\|x\|} \frac{\|Ax\|}{\|b\|} + \|\Delta A\| \right\}$$

Da $\|Ax\| \leq \|A\|\|x\|$ hat man schließlich Gleichung (3). □

Definition 4.4 (Konditionszahl)

Es sei $A \in \mathbb{R}^{n \times n}$ regulär und $\|\cdot\|$ eine Matrixnorm. Dann heißt

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

die Konditionszahl der Matrix A in der gewählten Norm.

Satz 4.5

Es sei $A \in \mathbb{R}^{n \times n}$ regulär und $\|\cdot\|$ eine Vektornorm in \mathbb{R}^n . Für die zugeordnete Matrixnorm und die damit gebildete Konditionszahl $\text{cond}(A)$ gilt:

$$\text{cond}(A) = \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|}$$

und eine untere Schranke für den relativen Abstand der zu A nächsten singulären Matrix

$$\min \left\{ \frac{\|A - B\|}{\|A\|} \mid B \in \mathbb{R}^{n \times n} \text{ singulär} \right\} \geq \text{cond}^{-1}(A)$$

► Bemerkung 4.6

Falls $\text{cond}(A)$ groß ist, heißt die Matrix A bzw. das lineare Gleichungssystem $Ax = b$ schlecht konditioniert, andernfalls gut konditioniert.

Sei \tilde{x} so gegeben, dass der Defekt $\|A\tilde{x} - b\|$ klein ist. Selbst dann muss man bei großer von A damit

rechnen, dass

$$\frac{\|x - \tilde{x}\|}{\|x\|}$$

groß ausfällt.

Kapitel IV

Kondition von Aufgaben und Stabilität von Algorithmen

1. Maschinenzahlen und Rundungsfehler

Ein Computer kann nur endlich viele Maschinenzahlen in normalisierter Gleitpunktdarstellung

$$z = \sigma \cdot d_0 d_1 \dots d_{t-1} \cdot b^e$$

exakt speichern, wobei

- $b \in \mathbb{N}$ mit $b \geq 2$ die Basis
- $d_0 d_1 \dots d_{t-1}$ mit $d_0, d_1, \dots, d_{t-1} \in \{0, \dots, b-1\}$ die Mantisse mit den Ziffern d_i
- $t \in \mathbb{N}$ mit $t \geq 1$ die Mantissenlänge
- $e \in \mathbb{N}$ mit $-m \leq e \leq M$ der Exponent
- $\sigma \in \{+1, -1\}$ das Vorzeichen

bedeuten. Zusätzlich ist 0 eine Maschinenzahl. Mit $\mathbb{M} = \mathbb{M}(b, t, m, M)$ wird die Menge aller Maschinenzahlen bezeichnet. Jede andere Zahl $x \in \mathbb{R}$, die im Computer gespeichert werden soll (auch Zwischenergebnisse), wird vorher auf eine Zahl $\text{rd}(x) \in \mathbb{M}$ so gerundet, dass der durch die Rundung entstehende relative Fehler durch

$$\frac{|\text{rd}(x) - x|}{|x|} = \min_{z \in \mathbb{M}} \frac{|z - x|}{|x|} \quad \text{für } x \in \mathbb{R} \setminus \mathbb{M}$$

gegeben ist.

Lemma 1.1

Für jedes $x \in \mathbb{R} \setminus \{0\}$ mit $b^{-m} \leq |x| \leq b^M$ gilt

$$\frac{|\text{rd}(x) - x|}{|x|} \leq \text{eps} = \frac{1}{2} b^{1-t}$$

Beweis. Ohne Beschränkung der Allgemeinheit sei $x > 0$. Dann gibt es Zahlen $e \in \mathbb{Z}$ mit $m \leq e \leq M$ und eine (gegebenenfalls unendliche) Ziffernfolge $(x_k) \subset \{0, \dots, b-1\}$, so dass

$$x = (x_0 x_1 \dots x_{t-1} x_t x_{t+1} \dots) \cdot b^e$$

Damit folgt

$$\begin{aligned} |\text{rd}(x) - x| &\leq \frac{b}{2} b^{e-t} \\ \frac{|\text{rd}(x) - x|}{|x|} &\leq \frac{b^{e-t+1}}{2b^e} \leq \frac{1}{2} b^{1-t} \end{aligned} \quad \square$$

Die Zahl \textit{eps} wird als Maschinengenauigkeit bezeichnet und gibt den maximalen relativen Rundungsfehler für $x \in [-b^m, b^M]$ an.

2. Fehleranalyse

2.1. Die Kondition einer Aufgabe

Unter Aufgabe wird hier die Auswertung einer zumindest stetig differenzierbaren Abbildung

$$\Phi : D \rightarrow \mathbb{R}^n$$

verstanden. Die Lösung der Aufgabe für ein Argument $a \in D \subset \mathbb{R}^n$ besteht also darin, $\Phi(a)$ zu ermitteln. Wir interessieren uns nun für die Frage, welchen Einfluss ein Fehler in a (also die Verwendung der Maschinenzahl $\text{rd}(a)$ statt a) auf das Ergebnis $\Phi(a)$ bei ansonsten exakter Rechnung hat. Dazu bezeichne $\tilde{a} \in D$ das fehlerbehaftete Argument. Für \tilde{a} nahe bei a erhält man aus der TAYLOR-Formel

$$\Phi(\tilde{a}) - \Phi(a) \approx \nabla \Phi(a)^T (\tilde{a} - a) = \left(\frac{\partial \Phi_i(a)}{\partial a_j} \right)_{\substack{i=1,\dots,m \\ j=1,\dots,n}} (\tilde{a} - a)$$

und damit (unter der Bedingung $a_j \neq 0$ und $\Phi(a)_i \neq 0$)

$$\frac{\Phi_i(\tilde{a}) - \Phi_i(a)}{\Phi_i(a)} \approx \frac{1}{\Phi_i(a)} \sum_{j=1}^n \frac{\partial \Phi_i(a)}{\partial a_j} (\tilde{a}_j - a_j) = \sum_{j=1}^n \frac{a_j}{\Phi_i(a)} \frac{\partial \Phi_i(a)}{\partial a_j} \frac{\tilde{a}_j - a_j}{a_j} \quad (1)$$

Definition 2.1 (relative Konditionszahlen, gut/schlecht konditioniert)

Es seien $D \subset \mathbb{R}^n$, $\Phi : D \rightarrow \mathbb{R}^m$ stetig differenzierbar und $a \in D$ mit $\Phi(a) \neq 0$. Dann heißen

$$K_{ij} = \left| \frac{a_j}{\Phi_i(a)} \frac{\partial \Phi_i(a)}{\partial a_j} \right|$$

relative Konditionszahlen der Aufgabe $\Phi(a)$. Die Aufgabe heißt gut konditioniert, wenn alle diese Konditionszahlen klein sind, sonst schlecht konditioniert.

Mit Gleichung (1) folgt

$$\frac{\Phi_i(\tilde{a}) - \Phi_i(a)}{\Phi_i(a)} \approx \sum_{j=1}^n K_{ij} \frac{\tilde{a}_j - a_j}{a_j}$$

Bei einer gut konditionierten Aufgabe verhält sich also der relative Fehler des Ergebnisses etwa wie die relativen Fehler der Eingangsdaten.

■ **Beispiel 2.2**

(a) Mit $\Phi(a) = \sqrt{a}$ für $a > 0$ erhält man $m = n = 1$ und

$$K_{11} = \left| \frac{a}{\sqrt{a}} \frac{1}{2\sqrt{a}} \right| = \frac{1}{2}$$

Die Aufgabe ist für alle $a > 0$ gut konditioniert.

(b) Mit $\Phi(a) = a_1 + a_2$ für $a_1 + a_2 \neq 0$ und $a_2 \neq 0$ erhält man $m = 1$, $n = 2$ und

$$K_{1j} = \left| \frac{a_j}{a_1 + a_2} \cdot 1 \right| \quad \text{für } j = 1, 2$$

Die Aufgabe ist schlecht konditioniert, falls $a_1 + a_2 \approx 0$. Selbstausslöschung bei Subtraktion etwa gleich großer Zahlen, zum Beispiel:

$$\begin{aligned} a_1 &= 1.23456789 & \tilde{a}_1 &= 1.234567885 \\ a_2 &= -1.23456788 & \tilde{a}_2 &= -1.23456788 \\ \Rightarrow a_1 + a_2 &= 10^{-8} \\ \Rightarrow \tilde{a}_1 + \tilde{a}_2 &= -0.5 \cdot 10^{-8} \end{aligned}$$

Der relative Fehler des Ergebnisses beträgt $\frac{\tilde{a}_1 + \tilde{a}_2 - (a_1 + a_2)}{a_1 + a_2} = 0.5$, also 50%!

2.2. Stabilität von Algorithmen

Nun wird untersucht, wie sich die Rundungsfehler in einzelnen Rechenschritten eines Algorithmus auswirken. Dazu wird die Aufgabe $\Phi : D \subset \mathbb{R} \rightarrow \mathbb{R}^m$ zu Grunde gelegt. ($m = 1$ ist keine Einschränkung, da die folgende Untersuchung auch für weitere Komponenten analog möglich ist) und ein allgemeiner Algorithmus mit stetig differenzierbaren Abbildungen $r_i : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ (mit $i = 0, \dots, N$) betrachtet.

■ Algorithmus 2.3

Input: a

$$\begin{aligned} 1 \quad y_0 &= r_0(a) \\ 2 \quad y_1 &= r_1(a, y_0) \\ 3 \quad &\dots \end{aligned}$$

Output: y_N ($= \Phi(a)$ bei exakter Rechnung)

Es stellt sich damit die Frage, wie sich die Rundungsfehler in y_0, \dots, y_{N-1} auf das Ergebnis des Algorithmus auswirken. Dazu betrachten wir die rekursiv definierten Restabbildungen $R_k : \mathbb{R}^{n+k+1} \rightarrow \mathbb{R}$ ($k = 0, \dots, N-1$) mit

$$\begin{aligned} R_{N-1}(a, y_0, \dots, y_{N-1}) &= r_N(a, y_0, \dots, y_{N-1}) \\ R_{N-2}(a, y_0, \dots, y_{N-2}) &= r_N(a, y_0, \dots, y_{N-2}, r_N(a, y_0, \dots, y_{N-2})) \\ &\vdots \\ R_0(a, y_0) &= \dots \end{aligned}$$

Für $k = 0, \dots, N-1$ wird nun die Kondition der durch $y_k \mapsto R_k(a, y_0, \dots, y_k)$ definierten Abbildung betrachtet, um festzustellen, wie sich Rundungsfehler in y_k auf das Ergebnis y_N auswirken.

Definition 2.4 (relative Konditionszahlen, stabil, instabil)

Die Zahlen

$$AK_k = \left| \frac{y_k}{R_k(a, y_0, \dots, y_k)} \cdot \frac{\partial R_k(a, y_0, \dots, y_k)}{\partial y_k} \right| \quad (2)$$

heißen relative Konditionszahlen des Algorithmus 2.3 bezüglich der Zwischenergebnisse y_k . Der Algorithmus 2.3 wird stabil (oder gutartig) genannt, wenn alle diese Konditionszahlen nicht wesentlich größer sind als die Konditionszahlen der Aufgabe $\Phi(a)$. Andernfalls heißt der Algorithmus instabil.

■ Beispiel 2.5

Um die größere Nullstelle des Polynoms $x^2 + 2px + q$ zu bestimmen, kann man

(a) $\Phi(p, q) = -p + \sqrt{p^2 - q}$ oder

(b) $\Theta(p, q) = \frac{q}{-p - \sqrt{p^2 - q}}$ (entsprechend dem Satz von VIETA, sofern $p^2 \neq q$)

verwenden. Wir setzen dabei

$$p \gg 1 > q > 0 \quad (3)$$

voraus. Die relativen Konditionszahlen zur Aufgabe $\Phi(p, q)$ ergeben sich zu

$$\begin{aligned} K_{11} &= \left| \frac{p}{-p + \sqrt{p^2 - q}} \left(-1 + \frac{p}{\sqrt{p^2 - q}} \right) \right| = \left| \frac{p}{-p + \sqrt{p^2 - q}} \left(\frac{-\sqrt{p^2 - q} + p}{\sqrt{p^2 - q}} \right) \right| \\ &= \left| \frac{p}{\sqrt{p^2 - q}} \right| \approx 1 \\ K_{12} &= \left| \frac{q}{-p + \sqrt{p^2 - q}} \frac{1}{2\sqrt{p^2 - q}} \right| = \left| \frac{q(p + \sqrt{p^2 - q})}{-q} \frac{1}{2\sqrt{p^2 - q}} \right| \\ &= \left| \frac{p + \sqrt{p^2 - q}}{2\sqrt{p^2 - q}} \right| \approx 1 \end{aligned}$$

Also sind die Aufgaben $\Phi(p, q)$ und $\Theta(p, q)$ für Gleichung (3) gut konditioniert. Die Umsetzung der Aufgabe $\Phi(p, q)$ erfolge mit dem Algorithmus

$$y_0 = p \cdot p \quad y_1 = y_0 - q \quad y_2 = \sqrt{y_1} \quad y_3 = -p + y_2 \quad (4)$$

Es ergibt sich $N = 3$, $n = 2$, $R_2(p, q, y_0, y_1, y_2) = -p + y_2$ und

$$AK_2 = \left| \frac{y_2}{R_2(p, q, y_0, y_1, y_2)} \frac{\partial R_2(p, q, y_0, y_1, y_2)}{\partial y_2} \right| = \left| \frac{y_2}{-p + y_2} \cdot 1 \right| = \left| \frac{\sqrt{p^2 - q}}{-p + \sqrt{p^2 - q}} \right| \gg 1$$

für Gleichung (3). Der Algorithmus Gleichung (4) ist daher dann numerisch instabil. Betrachten wir nun den Algorithmus

$$y_0 = p \cdot p \quad y_1 = y_0 - q \quad y_2 = \sqrt{y_1} \quad y_3 = -p - y_2 \quad y_4 = \frac{q}{y_3} \quad (5)$$

Es ist $N = 4$. Die Restabbildungen lauten dann

$$\begin{aligned}R_3(p, q, y_0, y_1, y_2, y_3) &= \frac{q}{y_3} \\R_2(p, q, y_0, y_1, y_2) &= \frac{q}{-p - y_2} \\R_1(p, q, y_0, y_1) &= \frac{q}{-p - \sqrt{y_1}} \\R_0(p, q, y_0) &= \frac{q}{-p - \sqrt{y_0 - q}}\end{aligned}$$

Unter der Bedingung Gleichung (3) sind die Konditionszahlen AK_0, \dots, AK_3 kleiner gleich 1, so dass der Algorithmus Gleichung (5) dann numerisch stabil ist. Unter anderen Bedingungen kann das Stabilitätsverhalten der Algorithmen Gleichung (4) und Gleichung (5) anders sein, so dass bei der Implementierung eine Fallunterscheidung notwendig ist.

Kapitel V

Newton-Verfahren zur Lösung nichtlinearer Gleichungssysteme

Seien $D \subseteq \mathbb{R}^n$ und $F : D \rightarrow \mathbb{R}^n$. Dann heißt $x^* \in D$ Nullstelle der Funktion F , wenn $F(x^*) = 0$ gilt. Wir interessieren uns für Methoden zur Bestimmung einer solchen Nullstelle.

Satz 0.1 (Taylor-Formel mit Integralrestglied)

es seien $D \subseteq \mathbb{R}^n$ offen und konvex sowie $F : D \rightarrow \mathbb{R}^m$ stetig differenzierbar. Dann gilt

$$F(x) = F(y) + F'(y)(x - y) + \int_0^1 \left(F'(y + t(y - x)) - F'(y) \right) (x - y) dt$$

für beliebige $x, y \in \mathbb{R}^n$.

Beweis. Sei $I \subset \mathbb{R}$ ein offenes Intervall. Nach dem Hauptsatz der Differential- und Integralrechnung gilt für eine stetig differenzierbare Funktion $\varphi : I \rightarrow \mathbb{R}$ die Beziehung

$$\varphi(t) = \varphi(0) + \int_0^t \varphi'(\tau) d\tau \quad (1)$$

für alle $t \in I$. Seien $f : D \rightarrow \mathbb{R}$ stetig differenzierbar und $x, y \in D$. Dann gibt es ein offenes Intervall $I \supset [0, 1]$, so dass $\varphi : I \rightarrow \mathbb{R}$ durch $\varphi(t) = f(y + t(x - y))$ wohldefiniert und stetig differenzierbar ist mit $\varphi'(t) = f'(y + t(x - y))(x - y)$. Mit Gleichung (1) folgt damit für $t = 1$

$$f(x) = f(y) + f'(y)(x - y) + \int_0^1 \left(f'(y + \tau(x - y)) - f'(y) \right) (x - y) d\tau$$

Setzt man noch $f = F_i$ für $i = 1, \dots, n$, so folgt die Behauptung komponentenweise. \square

1. Das Newton-Verfahren

Das NEWTON-Verfahren beruht auf der Idee, die Gleichung $F(x) = 0$ für eine gegebene Näherung $y \in \mathbb{R}^n$ einer Nullstelle durch die linearisierte Gleichung

$$F(y) + F'(y)(x - y) = 0$$

zu ersetzen, eine Lösung x dieser Gleichung (sofern möglich) zu berechnen und als neue Näherung für die Nullstelle zu verwenden.

■ Algorithmus 1.1 (Newton-Verfahren)

Input: $x^0 \in \mathbb{R}^n$, $\varepsilon > 0$ und F in geeigneter Form

```
1  k = 0
2  do while ||F(xk)|| > ε
3    compute xk+1
4    ! als Lösung von F'(xk)(x - xk) + F(xk) = 0
5    k = k + 1
6  end do
```

Output: x^k

Vorausgesetzt F ist differenzierbar, dann ist dieses Verfahren durchführbar, wenn alle auftretenden Matrizen $F'(x^k) \in \mathbb{R}^{n \times n}$ invertierbar sind.

■ Beispiel 1.2

Sei $n = 1$. Aus $F'(x^k)(x - x^k) + F(x^k) = 0$ erhält man eine eindeutige Lösung x^{k+1} , falls $F'(x^k) \neq 0$. Es folgt dann durch Umstellen

$$x^{k+1} = x^k - \frac{F(x^k)}{F'(x^k)}$$

Speziell sei $F: \mathbb{R} \rightarrow \mathbb{R}$ gegeben durch $F(x) = e^{-x} - x$ und $x^0 = 0.5$. Damit erhält man

$$x^1 = 0.5663\dots$$

$$x^2 = 0.56714316\dots$$

$$x^3 = 0.567143290409781\dots$$

$$x^4 = 0.567143290409784\dots$$

also lokal sehr schnelle Konvergenz. Es werden nun Voraussetzungen dafür angegeben.

Satz 1.3

Es sei $D \subseteq \mathbb{R}^n$ offen, $F : D \rightarrow \mathbb{R}^m$ differenzierbar und $x^* \in D$ eine reguläre Nullstelle von F , das heißt $F(x^*) = 0$ und $F'(*)$ ist regulär. Weiter sei $F' : D \rightarrow \mathbb{R}^{n \times n}$ LIPSCHITZ-stetig in D , das heißt es gibt ein $L > 0$, so dass

$$\|F'(x) - F'(y)\| \leq L\|x - y\| \quad \forall x, y \in D \quad (1)$$

Dann gibt es eine Kugel $B(x^*, \delta) = \{x \in \mathbb{R}^n \mid \|x - x^*\| \leq \delta\} \subset D$ um x^* , so dass Algorithmus 1.1 für jeden Startvektor $x^0 \in B(x^*, \delta)$ wohldefiniert ist. Falls der Algorithmus eine unendlich Folge $\{x^k\}$ erzeugt und $\delta > 0$ hinreichend klein ist, dann konvergiert $\{x^k\}$ gegen x^* und es gibt ein $C > 0$, so dass

$$\|x^{k+1} - x^k\| \leq C\|x^k - x^*\|^2 \quad (2)$$

für alle $k \in \mathbb{N}$ gilt.

Beweis. Da F' stetig in der offenen Menge D ist, gibt es $\delta_1 > 0$ und $M \geq 1$, so dass $B(x^*, \delta_1) \subset D$ sowie

$$F'(x) \text{ regulär und } \|F''(x)^{-1}\| \leq M \quad \forall x \in B(x^*, \delta_1) \quad (3)$$

Weiter folgt wegen Satz 0.1 und Gleichung (1)

$$\begin{aligned} \|F(x) - F(y) - F'(y)(x - y)\| &= \left\| \int_0^1 \left(F'(y + \tau(y - x)) - F'(y) \right) (x - y) d\tau \right\| \\ &\leq \int_0^1 \left\| \left(F'(y + \tau(y - x)) - F'(y) \right) (x - y) \right\| d\tau \\ &\leq \frac{1}{2} L \|x - y\|^2 \end{aligned}$$

für alle $x, y \in B(x^*, \delta_1)$. Daher, mit Gleichung (3) und Algorithmus 1.1, erhält man für $x^k \in B(x^*, \delta_1)$

$$\begin{aligned} \|x^{k+1} - x^*\| &= \|x^k - F'(x^k)^{-1}F(x^k) - x^*\| \\ &= \left\| -F'(x^k)^{-1} \left(F(x^k) + F'(x^k)(x^* - x^k) \right) \right\| \\ &\leq M \|F(x^k) + F'(x^k)(x^* - x^k)\| \\ &\leq \frac{1}{2} ML \|x^k - x^*\|^2 \end{aligned}$$

Sei nun $\delta \in [0, \delta_1)$ so gewählt, dass $ML\delta \leq 1$. Dann folgt mit $x^k \in B(x^*, \delta) \subseteq B(x^*, \delta_1)$

$$\|x^{k+1} - x^*\| \leq \frac{1}{2} ML \|x^k - x^*\|^2 \leq \frac{1}{2} \|x^k - x^*\| \quad (4)$$

Für $x^0 \in B(x^*, \delta)$ erhält man damit und mit Gleichung (3) induktiv, dass der Algorithmus 1.1 durchführbar ist und $\{x^k\} \subset B(x^*, \delta)$. Aus Gleichung (4) folgt daher die Konvergenz der Folge $\{x^k\}$ und Gleichung (2) für alle $k \in \mathbb{N}$. \square

Die Konvergenz der Folge $\{x^k\}$ gegen x^* zusammen mit der Eigenschaft Gleichung (2) wird als Q-quadratische Konvergenz der Folge $\{x^k\}$ gegen x^* bezeichnet.

► **Bemerkung 1.4**

Es gibt zahlreiche Modifikationen des NEWTON-Verfahrens, zum Beispiel

- **Sekantenverfahren** für $n = 1$: Dabei wird der Anstieg $F'(x^k)$ der Tangente $y = F'(x^k)(x - x^k) + F(x^k)$ an den Graphen von F im Punkt $(x^k, F(x^k))$ durch den Anstieg

$$\frac{F(x^k) - F(x^{k-1})}{x^k - x^{k-1}}$$

der Sekante durch die Punkte $(x^k, F(x^k))$ und $(x^{k-1}, F(x^{k-1}))$ des Graphen von F ersetzt. Unter Voraussetzungen wie in Satz 1.3 lässt sich in einer Umgebung einer regulären Nullstelle x^* die Konvergenz des Sekantenverfahrens gegen x^* zeigen, genauer gibt es $c, \delta > 0$ und $\kappa \in (0, 1)$, so dass

$$\|x^k - x^*\| \leq c\kappa^{\tau^k}$$

für alle Startwerte $x^0, x^1 \in B(x^*, \delta)$ mit $x^0 \neq x^1$ gilt, wobei $\tau = \frac{1+\sqrt{5}}{2}$. Die Folge $\{x^k\}$ ist mit der R-Ordnung τ gegen x^* konvergent.

- **Quasi-Newton-Verfahren**: Hier wird $F'(x^k)$ durch eine Matrix $B_k \in \mathbb{R}^{n \times n}$ ersetzt, so dass die Quasi-NEWTON Gleichung (oder Sekantengleichung)

$$B_k(x^k - x^{k-1}) = F(x^k) - F(x^{k-1})$$

erfüllt ist, Speziell leistet dies etwa das BROYDEN-Verfahren mit

$$B_k = B_{k-1} + \frac{\left(F(x^k) - F(x^{k-1}) - B_{k-1}(x^k - x^{k-1})\right)(x^k - x^{k-1})^T}{(x^k - x^{k-1})^T(x^k - x^{k-1})}$$

- **Inexakte Newton-Verfahren**: Dabei ist x^{k+1} so zu wählen, dass

$$\|F(x^k) + F'(x^k)(x^{k+1} - x^k)\| \leq q_k \|F(x^k)\|$$

mit einer Nullfolge $\{q_k\} \subset [0, 1)$. Speziell lassen sich in diesem Rahmen Verfahren analysieren, bei denen etwa $F'(x^k)$ durch eine Differenzapproximation ersetzt wird oder das lineare Gleichungssystem des NEWTON-Verfahrens nur näherungsweise gelöst wird.

2. Gedämpftes Newton-Verfahren

Es wird eine Möglichkeit zur Erreichung globaler Konvergenzeigenschaften des NEWTON-Verfahrens vorgestellt. Dazu verwenden wird die Funktion

$$\phi(x) : \begin{cases} \mathbb{R}^n & \rightarrow \mathbb{R} \\ x & \mapsto \frac{1}{2} \|F(x)\|_2^2 \end{cases}$$

zur Beurteilung der Güter der Näherung x . Falls $F'(x)$ regulär und $d(x) \in \mathbb{R}^n$ die NEWTON-Richtung im Punkt x ist, das heißt die Gleichung $F(x) + F'(x)d = 0$ löst, dann folgt

$$\phi'(x)d(x) = F(x)^T F'(x)d(x) = F(x)^T F'(x) (-F'(x)^{-1} F(x)) = -2\phi(x) < 0 \quad (1)$$

und

$$\begin{aligned} \phi(x + td(x)) &= \phi(x) + t\phi'(x)d(x) + o(t) \\ &= \phi(x) - 2t\phi(x) + o(t) \\ &= (1 - 2t)\phi(x) + o(t) \end{aligned} \quad (2)$$

das heißt $\phi(x + td(x)) < \phi(x)$ für alle $t > 0$ hinreichend klein. Die NEWTON-Richtung $d(x)$ ist also eine Abstiegsrichtung von ϕ im Punkt x . Die Idee besteht nun darin, eine Iteration der Form

$$x^{k+1} = x^k + t_k d^k$$

durchzuführen, wobei $d^k = d(x^k)$ die NEWTON-Richtung im Punkt x^k und $t_k > 0$ eine Schrittweite ist, die so gewählt wird, dass zumindest $\phi(x^{k+1}) < \phi(x^k)$ gilt. Zur Bestimmung einer geeigneten Schrittweite wird im folgenden Algorithmus die ARMIJO-Schrittweitenstrategie verwendet. Dazu sei

$$S = \{2^{-i} \mid i \in \mathbb{N}\} = \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\right\}$$

■ Algorithmus 2.1

Input: $x^0 \in \mathbb{R}^n$, $\varepsilon \geq 0$, $q \in (0, 1)$ und F in geeigneter Form

```

1  k = 0
2  do while  $\|F(x^k)\| > \varepsilon$ 
3    compute  $d^k$ 
4    ! als Lösung von  $F'(x^k)d^k + F(x^k) = 0$ 
5    compute  $t_k = \max\{t \in S \mid \phi(x^k + td^k) \leq (1-qt)\phi(x^k)\}$ 
6     $x^{k+1} = x^k + t_k d^k$ 
7    k = k + 1
8  end do
```

Output: x^k

Zur Formulierung des folgenden Satzes wird die Niveaumenge

$$W(x_0) = \{x \in \mathbb{R}^n \mid \phi(x) \leq \phi(x^0)\}$$

benötigt.

Satz 2.2

Es sei $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ differenzierbar. Weiter sei $F' : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ lokal LIPSCHITZ-stetig und F' regulär für alle $x \in W(x^0)$. Dann ist der Algorithmus 2.1 wohldefiniert. Falls die vom Algorithmus erzeugte Folge $\{x^k\}$ eine gegen x^* konvergente Teilfolge besitzt, so gilt $F(x^*) = 0$ und es gibt $k_0 \in \mathbb{N}$, so dass $t_k = 1$ für alle $k \geq k_0$ (Übergang ins ungedämpfte NEWTON-Verfahren). Außerdem konvergiert $\{x^k\}$ dann Q-quadratisch gegen x^* .

Beweis. Sei $x^k \in W(x^0)$. Dann folgt nach Voraussetzung die Regularität von $F'(x^k)$. Somit ist die NEWTON-Richtung $d^k = d(x^k)$ und (unter Beachtung von Gleichung (2)) die Schrittweite $t_k \in S$ wohldefiniert. Also gilt $\phi(x^{k+1}) < \phi(x^k)$ und somit $x^{k+1} \in W(x^0)$. Die Wohldefiniertheit des Algorithmus folgt damit induktiv, da $x^0 \in W(x^0)$.

Wir zeigen nun, dass $F(x^*) = 0$. Sei $\delta > 0$ zunächst beliebig gewählt. Mit Satz 0.1 und Gleichung (1) folgt

$$\begin{aligned} \phi(x + td(x)) &= \phi(x) + t\phi'(x)d(x) + \int_0^1 \left(\phi'(x + std(x)) - \phi'(x) \right) td(x) ds \\ &= (1 - 2t)\phi(x) + \int_0^1 \left(\phi'(x + std(x)) - \phi'(x) \right) td(x) ds \end{aligned} \quad (3)$$

für alle $x \in B(x^*, \delta)$. Nach Voraussetzung sind F und F' stetig in \mathbb{R}^n . Wegen $x^* \in W(x^0)$ ist nach Voraussetzung $F'(x^*)$ regulär. Wir können damit $\delta > 0$ klein genug voraussetzen, so dass die Abbildung $x \mapsto d(x) = -F'(x)^{-1}F(x)$ stetig in $B(x^*, \delta)$ ist. Daher gibt es $c > 0$ mit

$$\|d(x)\| \leq c \quad \forall x \in B(x^*, \delta) \quad (4)$$

Die lokale LIPSCHITZ-Stetigkeit von F' zieht die lokale LIPSCHITZ-Stetigkeit von $\phi' = F^T F'$ nach sich, das heißt es gibt ein $L > 0$, so dass

$$\|\phi'(x) - \phi'(y)\| \leq L\|x - y\| \quad \forall x, y \in B(x^*, \delta) \quad (5)$$

Da zumindest eine Teilfolge $\{x^k\}_N$ von $\{x^k\}$ gegen x^* konvergiert, gibt es wegen Gleichung (4) ein $\bar{t} > 0$ und ein $k_0 \in \mathbb{N}$, so dass

$$x^k, x^k + td(x^k) \in B(x^*, \delta) \quad \forall k \in N \text{ mit } k \geq k_0 \text{ und alle } t \in [0, \bar{t}] \quad (6)$$

Somit liefert Gleichung (3) unter Beachtung von Gleichung (4) und Gleichung (5)

$$\begin{aligned} \phi(x^k + td(x^k)) &= (1 - 2t) + \max_{s \in [0, 1]} \left\{ \|\phi'(x^k + std(x^k)) - \phi'(x^k)\| \right\} t \|d(x^k)\| \\ &= (1 - 2t)\phi(x^k) + t^2 L c^2 \end{aligned}$$

für alle $t \in [0, \bar{t}]$ und alle $k \in N$ mit $k \geq k_0$. Um die Schrittweitenbedingung vom Algorithmus 2.1 zu realisieren, muss t_k als größtes Element aus S bestimmt werden, so dass

$$\phi(x^k + t_k d(x^k)) \leq (1 - qt_k)\phi(x^k)$$

gilt. Daraus folgt dann für $k \in N$ und $k \geq k_0$

$$t_k \geq \min \left\{ \bar{t}, \frac{(2-q)\phi'(x^k)}{2Lc^2} \right\}$$

Angenommen $\phi(x^*) > 0$. Dann zieht dies $\phi(x^k) \geq \frac{1}{2}\phi(x^*)$ für unendlich viele $k \in N$ nach sich. Also gilt

$$t_k \geq \hat{t} = \min \left\{ \bar{t}, \frac{(2-q)\phi'(x^*)}{4Lc^2} \right\} > 0$$

für unendlich viele $k \in N$. Da $\{\phi(x^k)\}$ monoton fällt und unendlich oft

$$\phi(x^{k+1}) = \phi(x^k + t_k d(x^k)) \leq (1 - qt_k)\phi(x^k) \leq (1 - q\hat{t})\phi(x^k)$$

erfüllt ist, muss $\{\phi(x^k)\}$ gegen 0 konvergieren. Die Stetigkeit von ϕ zieht dann $\phi(x^*) = 0$ (und damit einen Widerspruch zur Annahme) nach sich. Also ist x^* eine Nullstelle von F .

Nun wird der Übergang ins ungedämpfte ($t_k = 1$) NEWTON-Verfahren gezeigt. Mit Satz 0.1 erhält man

$$F(x^k + d^k) = F(x^k) + F'(x^k)d^k + \int_0^1 \left(F'(x^k + sd^k) - F'(x^k) \right) d^k ds \quad (7)$$

Mit den getroffenen Glattheits- und Regularitätsvoraussetzungen ist $F'(x)$ regulär für alle $x \in B(x^*, \delta)$ (für δ hinreichend klein) und es gibt Zahlen $L_0, M > 0$, so dass

$$\begin{aligned} \|F'(x) - F'(y)\| &\leq L_0 \|x - y\| \\ \|F'(x)^{-1}\| &\leq M \end{aligned}$$

für alle $x, y \in B(x^*, \delta)$ gilt. Für $k \in N$ hinreichend groß folgt (unter Beachtung von $F(x^k) + F'(x^k)d^k = 0$)

$$\|d^k\| \leq \|F'(x^k)^{-1}\| \|F(x^k)\| \leq M \|F(x^k)\| \quad (8)$$

und (unter Beachtung von $F(x^*) = 0$ und $\lim_{k \in N} x^k = x^*$)

$$\|F'(x^k + sd^k) - F'(x^k)\| \leq L_0 \|d^k\| \leq L_0 M \|F(x^k)\| \quad \forall s \in [0, 1]$$

Aus Gleichung (7) erhält man damit für $k \in N$ hinreichend groß

$$\|F(x^k + d^k)\| \leq \max_{s \in [0, 1]} \left\{ \|F'(x^k + sd^k) - F'(x^k)\| \right\} \|d^k\| \leq L_0 M^2 \|F(x^k)\|^2$$

Wählt man $\delta > 0$ auch so klein, dass $\|F(x)\| \leq L_0^{-1} M^{-2} \sqrt{1-q}$ für alle $x \in B(x^*, \delta)$, dann folgt weiter

$$\|F(x^k + d^k)\| \leq \sqrt{1-q} \|F(x^k)\|$$

und damit

$$\phi(x^k + d^k) \leq (1-q)\phi(x^k) \quad (9)$$

Für $\delta > 0$ hinreichend klein zieht also $x^k \in B(x^*, \delta)$ nach sich, dass $t_k = 1$ und wegen Gleichung (8) auch dass $x^{k+1} = x^k + d^k \in B(x^*, \delta)$.

Da unendlich viele Iterierte der Folge $\{x^k\}$ in der Kugel $B(x^*, \delta)$ liegen, gilt Gleichung (9) für unendlich viele $k \in \mathbb{N}$. Wegen $\phi(x^{k+1}) < \phi(x^k)$ für alle $k \in \mathbb{N}$, folgt $\lim_{k \rightarrow \infty} \phi(x^k) = 0$. Da ϕ stetig ist, muss x^* eine Nullstelle von ϕ und damit von F sein. Satz 1.3 liefert damit schließlich die Q-quadratische Konvergenz der Folge $\{x^k\}$ gegen x^* . \square

Kapitel VI

lineare Optimierung

1. Ecken und ihre Charakterisierung

2. Simplex-Verfahren

3. Die Tableauform des Simplex-Verfahrens

4. Revidiertes Simplex-Verfahren

5. Bestimmung einer ersten zulässigen Basislösung

Anhang

Index

- GAUSS'schen Normalgleichungen, 35
- HORNER-Schema, 6
- HOUSEHOLDER-Matrix, 37
- KEPLER'sche Fassregel, 17
- LAGRANGE-Form, 4
- LEGENDRE-Polynoms, 21
- SIMPSON-Formel, 17
- TSCHEBYSCHOW-Polynoms, 21

- Algorithmus
 - instabil, 49
 - stabil, 49

- Basisfunktionen, 40
- Basispolynom
 - LAGRANGE-Basispolynom, 4
 - NEWTON-Basispolynome, 5

- Dreieckssystem, 23

- Ersatzaufgabe, 34
- euklidischen Norm, 41

- Funktionenraum, 2

- gestaffeltes System, 23

- Interpolationsbedingungen, 2
- Interpolationspolynom, 4
- Interpolierende, 2

- Konditionszahl, 43
 - gut konditioniert, 43, 47
 - relative Konditionszahlen, 47, 49
 - schlecht konditioniert, 43, 47
- kubischer Interpolationspline, 9

- LDL-Faktorisierung, 33
- lineares Gleichungssystem
 - überbestimmt, 34
- lineares Quadraturmittelpunktproblem, 34

- Maschinengenauigkeit, 46
- Matrixnorm, 41
 - submultiplikativ, 41
 - verträglich, 41
 - zugeordnete Matrixnorm, 41
- Maximum-Norm, 6, 41

- Newton-Cotes-Formel
 - geschlossene NEWTON-COTES-Formel, 16
 - offene NEWTON-COTES-Formel, 16
- Nullstelle, 51

- Permutationsmatrix, 26
- Pivotelement, 26
- Pivotisierung
 - Spaltenpivotisierung, 26
 - vollständige Pivotisierung, 26
 - Zeilenpivotisierung, 26
- Pivotspalte, 26
- Polynomspline, 9
- positiv definit, 31

- Q-quadratische Konvergenz, 53

- Restabbildungen, 48

- Spaltensummen-Norm, 41
- Spektralnorm, 41
- Stützstellen, 2
- Stützstellensystem, 6
- Stützwerte, 2
- streng diagonaldominant, 25
- Summen-Norm, 41

- Trapezformel, 17

- Zeilenstufenform, 23
- Zeilensummen-Norm, 41
- zusammengesetzte SIMPSON-Formel, 20
- zusammengesetzte Trapezformel, 20