Vorlesungsskript

Numerische Mathematik

Wintersemester 2011/2012

Goethe-Universität Frankfurt am Main

Prof. Dr. Thomas Gerstner

Stand: 7. März 2012

Inhaltsverzeichnis

Einleitung

Die *numerische Mathematik* (kurz: Numerik) beschäftigt sich mit der Konstruktion und Analyse von Algorithmen zur Berechnung mathematischer Probleme auf dem Computer, wie z.B.

- die Lösung linearer Gleichungssysteme
- die Berechnung von (bestimmten) Integralen
- die Ermittlung der Nullstelle(n) einer stetigen Funktion

Dabei unterscheidet man zwischen

- *direkten Verfahren*, die bei unendlicher Rechengenauigkeit die exakte Lösung des Problems liefern (Beispiel: Gauss-Elimination) und
- *approximativen Verfahren*, die nur eine Näherung (Approximation) des Problems berechen (Beispiel: Quadraturformeln)

1 Fehleranalyse

1.1 Zahldarstellung

Ein (digitaler) Computer ist *endlich*:

- endlicher (Haupt-)Speicher, z.B. 2 GByte
- endlicher Sekundär-Speicher, z.B. Festplatte 600 GByte
- endlicher interne Rechengenauigkeit, z.B 32 Bit, 64 Bit

Zahlen werden durch *endliche Folgen* von 0 und 1 dargestellt.

1.1.1 Darstellung ganzer Zahlen

Natürliche Zahlen werden unter Verwendung von N Bits als *Dualzahl* dargestellt

$$\boxed{d_{N-1}d_{N-2}\dots d_0} \mathrel{\widehat{=}} \sum_{i=0}^{N-1} d_i 2^i,\ d_i \in \{0,1\} \tag{1}$$

Für festes N umfasst der *darstellbare Bereich* alle natürlichen Zahlen z mit

$$z_{min} = 0 \leq z \leq 2^N - 1 = z_{max} \tag{2}$$

Beispiel ($N = 4$): $0000 \mathrel{\widehat{=}} 0$
$0001 \mathrel{\widehat{=}} 1$
$0010 \mathrel{\widehat{=}} 2$
$\vdots$
$1111 \mathrel{\widehat{=}} 15 = 2^4 - 1$

Negative Zahlen könnte man im sogenannten *Einerkomplement* durch Invertieren aller Bits darstellen als

$$\boxed{1 \mid d_{N-2}d_{N-3}\dots d_0} \mathrel{\widehat{=}} \sum_{i=0}^{N-2}(1-d_i)2^i \tag{3}$$

die führende Eins zeigt dabei an, dass es sich um eine negative Zahl handelt.

Beispiel ($N = 4$): $5 = 0101, -5 = 1010$

Der darstellbare Bereich ist

$$z_{min} = -(2^{N-1}-1) \leq z \leq 2^{N-1}-1 = z_{max} \tag{4}$$

Nachteile dieser Darstellung sind:

- die Darstellung der 0 ist nicht eindeutig (0 0 ... 0, 1 0 ... 0)
- die Addition zweier ganzer Zahlen mit unterschiedlichem Vorzeichen ist nicht direkt durchführbar

Beispiel ($N = 4$):

$z_1 = 2$	0010	$z_1 = -2$	1101
$z_2 = 3$	0011	$z_2 = +3$	0011
$z_1 + z_2 = 5$	0101	$z_1 + z_2 = 0$(!)	0000

Deswegen werden negative ganze Zahlen in der Regel im *Zweierkomplement* dargestellt

$$\boxed{1 \mid d_{N-2}d_{N-3}\dots d_0} \mathrel{\widehat{=}} -\left(1 + \sum_{i=0}^{N-2}(1-d_i)2^i\right) \tag{5}$$

Das Vorgehen ist dabei ausgehend von der Dualdarstellung von $-z$ alle Bits umzuklappen und 1 zu addieren, z.B.

-3	$\widehat{=}$	-0011	$=$	$1100 + 1$	$=$	1101
Dezimalzahl		$-$Dualzahl		Umklappen + 1		Bitmuster von z

Beispiel ($N = 4$):

$$1111 \mathrel{\widehat{=}} -(1 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0) = -1$$
$$1110 \mathrel{\widehat{=}} -(1 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0) = -2$$
$$\vdots$$
$$1000 \mathrel{\widehat{=}} -(1 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0) = -8$$

Der darstellbare Bereich von Zahlen im Zweierkomplement ist

$$z_{min} = -2^{N-1} \leq z \leq 2^{N-1}-1 = z_{max} \tag{6}$$

Die Addition ist nun direkt durchführbar und die Darstellung der Null ist eindeutig.

Beispiel ($N = 4$):

$z_1 = 2$	0010
$z_2 = -3$	1101
$z_1 + z_2 = -1$	1111

Im sogenannten *Einerkomplement* wird die Addition der 1 weggelassen, dadurch bleibt jedoch die Nicht-Eindeutigkeit der Null bestehen. Alle ganzen Zahlen im darstellbaren Bereich können im Einer- bzw. Zweierkomplement exakt dargestellt werden.

Der Versuch ganze Zahlen mit $z < z_{min}$ oder $z > z_{max}$ (z.B. als Ergebnis einer Rechnung) darzustellen, führt zu *Überlauf*. Mögliche Reaktionen sind:

- Abbrechen mit Fehlermeldung
- Weiterrechnen mit $\tilde{z} = z \bmod z_{max}$ bzw. z_{min}
- Weiterrechnen mit $\tilde{z} = z_{max}$ bzw. z_{min}
- Weiterrechnen mit $\tilde{z} = +\infty$ bzw. $-\infty$ als spezielle Zahl

In modernen Programmierumgebungen wird hier eine Ausnahme (Exception) geworfen und der Benutzer kann selbst entscheiden, ob er mit dem Ergebnis weiterrechnen will.

1.1.2 Darstellung reeller Zahlen

Reelle Zahlen sind bekanntermassen überabzählbar, lückenlos und unbegrenzt, d.h.

- zu jeder reellen Zahl gibt es noch größere und kleinere reelle Zahlen
- zu jedem Paar reeller Zahlen gibt es unendlich viele weitere dazwischen liegende

Die Zahldarstellung eines Computers ist immer endlich, diskret und beschränkt. Demnach kann die Darstellung reeller Zahlen nur *näherungsweise* erfolgen.

Ziele:

1. mache einen möglichst geringen Fehler bei der Darstellung
2. decke einen möglichst großen Zahlenbereich ab
3. Elementaroperationen (Addition, Subtraktion, ...) sollen „stabil“ sein, d.h. die vorhandenen Fehler sollen sich nicht verstärken

Jeder Zahl $x \in \mathbb{R}$ im darstellbaren Bereich wird dabei eine Maschinenzahl $rd(x)$ so zugeordnet, dass entweder

$$|x - rd(x)| \leq \varepsilon \quad (\textit{absoluter Fehler}) \tag{7}$$

oder

$$\frac{|x - rd(x)|}{|x|} \leq \varepsilon \quad (\textit{relativer Fehler}) \tag{8}$$

für eine vorgegebene Fehlerschranke ε gilt.

Festkommaformat

Im *Festkommaformat* wird versucht, den absoluten Fehler bei vorgegebenen Zahlenbereich zu minimieren.

Eine Maschinenzahl im Festkommaformat mit N Bits und K Nachkommastellen ist definiert als

$$\boxed{s \mid d_{N-2}d_{N-3}\dots d_K \mid d_{K-1}d_{K-2}\dots d_0} \mathrel{\widehat{=}} (-1)^s \sum_{i=0}^{N-2} d_i 2^{i-K} \tag{9}$$

Beispiel ($N = 6, K = 2$):

$$7.25 = 0 \mid 111 \mid 01 = 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 + 0 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4}$$

Der darstellbare Bereich ist

$$x_{min} = -(2^{N-1} - 1)2^{-K} \leq x \leq (2^{N-1} - 1)2^{-K} = x_{max} \tag{10}$$

Die betragsmässig kleinste darstellbare Zahl ungleich Null ist

$$x_{|min|} = 2^{-K} \tag{11}$$

Der maximale absolute Fehler bei der Darstellung einer reellen Zahl x im darstellbaren Bereich ist im Festkommaformat

$$|x - rd(x)| < 2^{-K} \tag{12}$$

Beispiel ($N = 6, K = 2$):

Abbildung 1: Alle Festkommazahlen für $N = 7, K = 2$.

Nachteile:

- der darstellbare Bereich ist eng, betragsmäßig kleine und große Zahlen können nicht gut dargestellt werden
- es wird Speicherplatz verschwendet (siehe unten)
- der relative Fehler ist im allgemeinen das sinnvollere Fehlermass

Gleitkommaformat

Im Gleitkommaformat wird versucht, den relativen Fehler bei vorgegebenem Zahlenbereich zu minimieren.

Eine Maschinenzahl im *Gleitkommaformat* mit Mantissenlänge M und Exponentenlänge E ist definiert als

$$\boxed{s \mid e_{E-1}e_{E-2}\dots e_0 \mid d_{M-1}d_{M-2}\dots d_0} \mathrel{\widehat{=}} (-1)^s d \cdot 2^e \text{ mit } d = \sum_{i=0}^{M-1} d_i 2^{i-M} \text{ und } e = \sum_{j=0}^{E-1} e_j 2^j \tag{13}$$

Hierbei bezeichnet d die *Mantisse* und e den *Exponenten* der Gleitkommazahl.

Beispiel ($M = 4, E = 3$):

$$0 \mid 010 \mid 0101 = 5 \cdot 2^{-4} \cdot 2^2 = 1.25$$

Der Nachteil ist hier jedoch, dass die Zahldarstellung nicht eindeutig ist, z.B. ist ebenfalls

$$0 \mid 001 \mid 1010 = 10 \cdot 2^{-4} \cdot 2^1 = 1.25 \tag{14}$$

Um Eindeutigkeit bei der Zahldarstellung zu erreichen, wird die Mantisse auf ein vorgegebenes Binärintervall, z.B. $[1, 2)$, beschränkt. Gleichzeitig wird der Exponentenbereich durch Addition eines *Bias* in einen brauchbaren Bereich verschoben.

Eine Maschinenzahl im *normalisierten Gleitkommaformat* mit Mantissenlänge M, Exponentlänge E und Bias B ist definiert als

$$\boxed{s \mid e_{E-1}e_{E-2}\dots e_0 \mid d_{M-1}d_{M-2}\dots d_0} \mathrel{\widehat{=}} (-1)^s d \cdot 2^e \text{ mit } d = 1 + \sum_{i=0}^{M-1} d_i 2^{i-M} \text{ und } e = \left(\sum_{j=0}^{E-1} e_j 2^j\right) - B \tag{15}$$

Beispiel ($M = 3, E = 3, B = 3$):

$$0 \mid 011 \mid 010 = (1 + 2 \cdot 2^{-3}) \cdot (2^{3-3}) = 1.25$$

Dabei muss die führende Eins in der Mantisse nicht abgespeichert werden.

Der Wertebereich für den Exponenten ist

$$e_{min} = -B \leq e \leq (2^E - 1) - B = e_{max} \tag{16}$$

Der darstellbare Bereich für normalisierte Gleitkommazahlen ist damit

$$x_{min} = -(2 - 2^{-M})2^{e_{max}} \leq x \leq (2 - 2^{-M})2^{e_{max}} = x_{max} \tag{17}$$

und die betragsmäßig kleinste darstellbare Zahl ist

$$x_{|min|} = 2^{e_{min}} \tag{18}$$

Beispiel ($M = 3, E = 3, B = 3$):

$$e_{min} = -3, e_{max} = (2^3 - 1) - 3 = 4, x_{max} = (2 - 2^{-3})2^4 = 30, x_{|min|} = 2^{-3} = \frac{1}{8}$$

Abbildung 2: Alle (positiven) Gleitkommazahlen für $M = 3, E = 3, B = 3$.

Für den relativen Abstand zweier aufeinander folgender Gleitkommazahlen x_1, x_2 gilt

$$2^{-M-1} \leq \frac{|x_1 - x_2|}{|x_1|} \leq 2^{-M} \tag{19}$$

Die untere Schranke wird für Mantissen $0 \dots 00$ und $1 \dots 11$, die obere Schranke für Mantissen $0 \dots 00$ und $0 \dots 01$ angenomen.

Beispiel ($M = 3, E = 3, B = 3$):

$$0 \mid 110 \mid 111 = 15; 0 \mid 111 \mid 000 = 16; 0 \mid 111 \mid 001 = 18$$
$$(16 - 15)/16 = 2^{-4}, (18 - 16)/16 = 2^{-3}$$

Der maximale relative Fehler bei der Darstellung reeller Zahlen im darstellbaren Bereich ist für normalisierte Gleitkommazahlen bei korrekter Rundung

$$\varepsilon = \frac{1}{2}2^{-(M+1)+1} = 2^{-(M+1)} = eps \tag{20}$$

Die Zahl *eps* wird *Maschinengenauigkeit* genannt.

1.1.3 IEEE 754 Standard für Gleitkommazahlen

IEEE (sprich „I-Triple-E“) = Institute of Electrical and Electronics Engineers

Im IEEE 754 Standard (1985) werden (u.a.) definiert:

Zahl der Bits	Genauigkeit	Typ	Vorzeichen	Mantisse	Exponent	Bias
32 Bit	einfach	float	1 Bit	23 Bits	8 Bits	127
64 Bit	doppelt	double	1 Bit	52 Bits	11 Bits	1023

Im IEEE 754 Standard ist der Bias immer $B = 2^{E-1} - 1$. Der maximale ($e = 1 \dots 1$) und minimale ($e = 0 \dots 0$) Exponent sind reserviert, sodass für den Wertebereich des Exponenten gilt

$$e_{min} = -2^{E-1} + 2 \leq e \leq 2^{E-1} - 1 = e_{max} \tag{21}$$

Typ	Exponentenbereich
float	$-126 \leq e \leq 127$
double	$-1022 \leq e \leq 1023$

Insgesamt werden folgende Fälle unterschieden:

Exponent	Mantisse	Bedeutung
$0 \dots 0$	$0 \dots 0$	Null
$0 \dots 0$	$0 \dots 01$ bis $1 \dots 1$	denormalisierte Zahl
$0 \dots 01$ bis $1 \dots 10$	$0 \dots 0$ bis $1 \dots 1$	normalisierte Zahl
$1 \dots 1$	$0 \dots 0$	Unendlich
$1 \dots 1$	$0 \dots 01$ bis $1 \dots 1$	keine Zahl (NaN)

Um Zahlen, die betragsmässig kleiner als $x_{|min|}$ darzustellen, werden, wenn der Exponent Null, die Mantisse aber ungleich Null ist, *denormalisierte Gleitkommazahlen* (d.h. ohne die führende Eins) verwendet:

$$\boxed{s \mid 0 \dots 0 \mid d_{M-1} d_{M-2} \dots d_0} \mathrel{\widehat{=}} (-1)^s \sum_{i=0}^{M-1} d_i 2^{i-M} \cdot 2^{e_{min}} \tag{22}$$

Auf diese Weise wird die Lücke zur Null weiter geschlossen, was jedoch auf Kosten der Genauigkeit geschieht. Der Versuch, noch kleinere Zahlen darzustellen, führt zu *Unterlauf*.

Damit gilt:

Typ	x_{max}	$x_{\|min\|}$	eps
float	$(2 - 2^{-23}) \cdot 2^{127} \approx 3.4 \cdot 10^{38}$	$2^{-23} \cdot 2^{-126} \approx 1.4 \cdot 10^{-45}$	$2^{-23+1} \approx 6.0 \cdot 10^{-8}$
double	$(2 - 2^{-52}) \cdot 2^{1023} \approx 1.8 \cdot 10^{308}$	$2^{-52} \cdot 2^{-1022} \approx 4.9 \cdot 10^{-324}$	$2^{-52+1} \approx 1.1 \cdot 10^{-16}$

Hinweis zum Schluss: Ganze Zahlen, Festkommazahlen, Gleitkommazahlen, etc. lassen sich auch in einer anderen Basis als 2 definieren, z.B.

$$\boxed{d_{N-1} d_{N-2} \dots d_0} \mathrel{\widehat{=}} \sum_{i=0}^{N-1} d_i q^i, \; d_i \in \{0, 1, \dots q-1\} \tag{23}$$

Beispiele: Ternärzahlen ($q = 3$), Quaternärzahlen ($q = 4$), Oktalzahlen ($q = 8$), Dezimalzahlen ($q = 10$), Hexadezimalzahlen ($q = 16$), ...

1.2 Rundungsfehler

1.2.1 Zahldarstellung

Jede reelle Zahl x im Darstellungsbereich hat in der Menge der Gleitkommazahlen $\mathbb{G}$ genau einen rechten und einen linken Nachbar

$$g_L := \max\{g \in \mathbb{G}, g \leq x\},\ g_R := \min\{g \in \mathbb{G}, g \geq x\} \tag{24}$$

Insbesondere gilt: $x \in \mathbb{G} \Rightarrow g_L = g_R = x$

Damit lässt sich *Rundung* in $\mathbb{G}$ definieren als

Abrunden	$rd_- : \mathbb{R} \to \mathbb{G}, x \mapsto g_L$	
Aufrunden	$rd_+ : \mathbb{R} \to \mathbb{G}, x \mapsto g_R$	
Abhacken	$rd_o : \mathbb{R} \to \mathbb{G}, x \mapsto$	$\begin{cases} g_L & \text{falls} \quad x \geq 0 \\ g_R & \text{falls} \quad x \leq 0 \end{cases}$
Korrektes Runden	$rd_* : \mathbb{R} \to \mathbb{G}, x \mapsto$	$\begin{cases} g_L & \text{falls} \quad x \leq (g_L + g_R)/2 \\ g_R & \text{falls} \quad x \geq (g_L + g_R)/2 \end{cases}$

Abrunden, Aufrunden und Abhacken werden unter dem Begriff *gerichtetes Runden* zusammengefasst.

Anmerkung: im IEEE-Standard wird beim korrekten Runden für $\geq$ und $\leq$ auch noch gerader/ungerader erster Ziffer unterschieden.

Alle vier Rundungsarten erfüllen als Abbildung $rd : \mathbb{R} \to \mathbb{G}$:

1. Idempotenz: $x \in \mathbb{G} \Rightarrow rd(x) = x$
2. Monotonie: $x \leq y \Rightarrow rd(x) \leq rd(y)$
3. Projektivität: $rd(rd(x)) = rd(x)$

Anmerkung: Die Abbildung rd ist nicht injektiv.

Zusammenfassung:

Für Festkommazahlen mit N Bits und K Nachkommastellen gilt im Darstellungsbereich für den absoluten Fehler

$$|x - rd(x)| \leq \begin{cases} 2^{-K} & \text{für gerichtete Rundung} \\ \frac{1}{2}2^{-K} & \text{für korrekte Rundung} \end{cases} \tag{25}$$

Für normalisierte Gleitkommazahlen mit Mantissenlänge M, Exponentlänge E und Bias B gilt im Darstellungsbereich für den relativen Fehler

$$\frac{|x - rd(x)|}{|x|} \leq \begin{cases} 2^{-M} & \text{für gerichtete Rundung} \\ \frac{1}{2}2^{-M} (= eps) & \text{für korrekte Rundung} \end{cases} \tag{26}$$

1.2.2 Elementare Operationen

Elementare Operationen $+, -, \times, \div$ sind in $\mathbb{G}$ nicht exakt ausführbar, statt dessen erhält man Näherungen, also für alle $a, b \in \mathbb{G}$ und für $* \in \{+, -, \times, \div\}$:

- exaktes Resultat: $a * b \in \mathbb{R}$, i.A. $\notin \mathbb{G}$
- berechnete Näherung: $a \dot{*} b \in \mathbb{G}$

Für einen Rechner mit *idealer Arithmetik* gilt

$$a \dot{*} b = rd(a * b) \tag{27}$$

für alle $a, b \in \mathbb{G}$ und $* \in \{+, -, \times, \div\}$, d.h. die berechnete Näherung ist allein Funktion des exakten Resultats.

Der IEEE-Standard verlangt eine ideale Arithmetik für alle arithmetischen Grundoperationen und für alle Genauigkeitsstufen.

Die Implementierung einer idealen Arithmetik erfordert Sorgfalt beim Chip-Design, z.B. je ein zusätzliches Schutzbit und Rundungsbit für die Mantisse.

Für ideale Arithmetik gelten folgende Rundungsfehler-Schranken:

$$a \dot{+} b = (a + b)(1 + \alpha) \tag{28}$$
$$a \dot{-} b = (a - b)(1 + \sigma) \tag{29}$$
$$a \dot{\times} b = (a \times b)(1 + \mu) \tag{30}$$
$$a \dot{\div} b = (a \div b)(1 + \delta) \text{ (für } b \neq 0) \tag{31}$$

wobei α, σ, μ und δ von a und b abhängen und im Betrag beschränkt sind, d.h.

$$|\alpha|, |\sigma|, |\mu|, |\delta| \leq \begin{cases} 2 \cdot eps & \text{für gerichtete Rundung} \\ eps & \text{für korrekte Rundung} \end{cases} \tag{32}$$

Beispiel ($q = 10, M = 4, eps = 0.00005$):

a	b	$a \times b$	$a \dot{\times} b$	μ
20.29	49.31	1000.4999	1000	$-0.4999/1000.4999$
28.75	34.80	1000.5000	1000 oder 1001	$\pm 0.5/1000.5$
20.71	48.31	1000.5001	1001	$+0.4999/1000.5001$

$\alpha, \sigma, \mu, \delta$ sind also die relativen Fehler der berechneten Näherungen (sofern kein Über- oder Unterlauf auftritt).

Die arithmetischen Vergleiche $<, \leq, =, \neq, \geq, >: \mathbb{G} \times \mathbb{G} \to \{true, false\}$ müssen ebenfalls fehlerfrei implementiert sein, also nie zu einer Bereichsüberschreitung führen. Daher sollte man nie $(a - b) > 0$ statt $a > b$ implementieren.

Da die Rundung nicht injektiv ist, repräsentiert jede Gleitkommazahl x ein Intervall reeller Zahlen $[x - r, x + r]$. Gleichheit zweier Gleitkommazahlen $x, \tilde{x}$ bedeutet also

$$x \in [\tilde{x} - r, \tilde{x} + r] \Rightarrow x \doteq \tilde{x} \tag{33}$$

Daher sollte man nie auf $x = \tilde{x}$ abfragen, sondern stets Ausdrücke der Form $|x - \tilde{x}| \leq r$ verwenden. Rechnen mit Gleitkommazahlen ist Rechnen mit Intervallen.

1.3 Fehlerfortpflanzung

1.3.1 Stabilität von Algorithmen

Ein *Algorithmus* (Rechenverfahren) besteht aus einer endlichen Folge von Grundoperationen, deren Ablaufreihenfolge eindeutig festgelegt ist.

Neben der Entwicklung möglichst effizienter Algorithmen beschäftigt sich die Numerik mit der Analyse der auftretenden Rundungsfehler. Die grundlegenden Vorgehensweisen sind dabei:

- *Vorwärtsanalyse*: vergleiche das berechnete Ergebnis mit dem exakten Resultat. Dabei wird in jedem Schritt der größtmögliche Fehler angenommen (worst case error) $\rightarrow$ Überschätzung des wahren Fehlers (Korrelationen werden nicht berücksichtigt), oft leider unpraktikabel
- *Rückwärtsanalyse*: interpretiere das berechnete Ergebnis als exaktes Resultat zu geeignet veränderten Eingabedaten $\rightarrow$ erlaubt den Vergleich mit Fehlern in Eingabedaten (z.B. Messfehlern)

Beispiel: $y = f(x)$, y=Ergebnis, f Algorithmus, x Eingabedaten

- Vorwärtsfehler: $\Delta y = \tilde{y} - y$
- Rückwärtsfehler: kleinstes Δx sodass $f(x + \Delta x) = \tilde{y}$

Beispiel: $f(a, b, c) = a + b + c$

- Algorithmus 1: $f(a, b, c) = (a + b) + c$
- Algorithmus 2: $f(a, b, c) = a + (b + c)$

Beispieldaten: $a := 2.3371258 \times 10^{-5}, b := 3.3678429 \times 10^{1}, c := -3.3677711 \times 10^{1}$, 8 Stellen Genauigkeit

$$\begin{aligned} a\dot{+}(b\dot{+}c) &= 2.3371258 \times 10^{-5}\dot{+}6.1800000 \times 10^{-4} = 6.4137126 \times 10^{-4} \\ (a\dot{+}b)\dot{+}c &= 3.3678452 \times 10^{1}\dot{+}3.3677811 \times 10^{1} = 6.4100000 \times 10^{-4} \end{aligned}$$

Das exakte Resultat ist $a + b + c = 6.41371258 \times 10^{-4}$.

- relativer Fehler von Algorithmus 1: $\approx 3.11 \times 10^{-9}$
- relativer Fehler von Algorithmus 2: $\approx 5.78 \times 10^{-4}$

Rückwärtsanalyse von Algorithmus 1:

$$\begin{aligned} \eta &= (a + b)(1 + \varepsilon_1) \\ \tilde{y} &= (\eta + c)(1 + \varepsilon_2) = ((a + b)(1 + \varepsilon_1) + c)(1 + \varepsilon_2) = (a + b + c)(1 + \frac{a + b}{a + b + c}\varepsilon_1(1 + \varepsilon_2) + \varepsilon_2) \\ \frac{\tilde{y} - y}{y} &= \frac{a + b}{a + b + c}\varepsilon_1(1 + \varepsilon_2) + \varepsilon_2 \approx \frac{a + b}{a + b + c} \cdot \varepsilon_1 + 1 \cdot \varepsilon_2 \end{aligned}$$

wobei im letzten Schritt der Term höherer Ordnung vernachlässigt wurde. Die Verstärkungsfaktoren $(a+b)/(a+b+c)$ und 1 geben an, wie stark sich Rundungsfehler $\varepsilon_1, \varepsilon_2$ im relativen Fehler des Resultats auswirken.

Für Algorithmus 2 gilt analog:

$$\frac{\tilde{y} - y}{y} \approx \frac{b + c}{a + b + c} \cdot \varepsilon_1 + 1 \cdot \varepsilon_2 \tag{34}$$

Je nachdem ob $|a + b|$ oder $|b + c|$ kleiner ist, ist es günstiger, Algorithmus 1 bzw. 2 zu verwenden. Im Beispiel:

- Algorithmus 1: $\frac{a+b}{a+b+c} \approx 5 \times 10^4$
- Algorithmus 2: $\frac{b+c}{a+b+c} \approx 0.97$

Fazit: Die Subtraktion annähernd gleich großer Zahlen ist in jedem Fall zu verhindern, da in diesem Fall *Auslöschung* führender Ziffern auftritt.

Ein durch einen Algorithmus berechnetes Ergebnis $\tilde{y}$ heisst *akzeptabel*, wenn es als exaktes Ergebnis zu gestörten Eingabewerten verstanden werden kann, d.h.

$$\tilde{y} = f(\tilde{x}) \text{ mit } |\tilde{x} - x| \leq C \cdot eps \quad \text{bzw.} \quad \frac{|\tilde{x} - x|}{|x|} \leq C \cdot eps \tag{35}$$

Ein Algorithmus $\tilde{f}$ heißt *numerisch stabil*, wenn er für alle zulässigen und in der Größenordnung der Maschinengenauigkeit gestörten Eingabedaten akzeptable Resultate liefert, d.h.

$$|f(\tilde{x}) - \tilde{f}(x)| \leq C \cdot eps \quad \text{bzw.} \quad \frac{|f(\tilde{x}) - \tilde{f}(x)|}{|f(x)|} \leq C \cdot eps \tag{36}$$

Die arithmetischen Grundoperationen sind stabil, aber die Hintereinanderausführung stabiler Operationen ist nicht automatisch stabil.

1.3.2 Kondition von Problemen

Die *Kondition* eines Problems ist die Empfindlichkeit der Resultate des Problems gegenüber den Eingabedaten. Bei hoher Empfindlichkeit spricht man von schlechter Kondition, bei geringer von guter Kondition.

Beispiel: Berechnung des Schnittpunkts zweier Geraden.

- sind die Geraden fast orthogonal, so verändert ein leichtes Ändern an den Parametern auch den Schnittpunkt nur leicht
- sind die Geraden fast parallel, so führt eine leichte Änderung an den Parametern zu großen Änderungen am Schnittpunkt

In der Praxis sind schlecht konditionierte Probleme nur schwer (im Extremfall gar nicht) lösbar, da jede kleine Störung (z.B. durch Rundungsfehler) das Ergebnis unbrauchbar macht.

Nur für gut konditionierte Probleme ist es daher sinnvoll, stabile Algorithmen zu entwickeln.

In erster Näherung gilt für infinitesimale Änderungen δx (und falls f differenzierbar):

$$\delta y = \frac{df}{dx} \delta x \tag{37}$$

Die wichtigsten Konditionszahlen sind daher

- absolute Kondition: $cond_{abs}(f) = \left| \frac{df}{dx} \right|$

- relative Kondition: $cond_{rel}(f) = \left|\frac{x}{y}\frac{df}{dx}\right|$

Für die vier Grundoperationen gilt für absolute Änderungen δx und relative Änderungen $\rho x = \delta x/x$:

$$\begin{aligned}
\delta(a+b) &= \delta a + \delta b & \rho(a+b) &= \rho a \cdot \frac{a}{a+b} + \rho b \cdot \frac{b}{a+b} \\
\delta(a-b) &= \delta a - \delta b & \rho(a-b) &= \rho a \cdot \frac{a}{a-b} + \rho b \cdot \frac{b}{a-b} \\
\delta(a\times b) &= b\cdot\delta a + a\cdot\delta b & \rho(a\times b) &= \rho a + \rho b \\
\delta(a\div b) &= \frac{\delta a}{b} - \frac{a\delta b}{b^2} & \rho(a\div b) &= \rho a - \rho b
\end{aligned}$$

Insbesondere sieht man, dass die relativen Konditionszahlen für die Multiplikation und Division 1 (die Operationen also gut konditioniert) sind. Bei Addition und Subtraktion sind die absoluten Konditionszahlen zwar klein, aber die relativen Konditionszahlen unbegrenzt (Auslöschung kann auftreten).

Wenn ein Problem aus mehreren Teilproblemen besteht, z.B. $f(x) = g(h(x))$, so lässt sich die Gesamtkondition auf die Kondition der Teilprobleme zurückführen, z.B.

$$cond(g\circ h) = \left.\frac{dg(z)}{dz}\right|_{z=h(x)} \frac{dh(x)}{dx} \tag{38}$$

Die Gesamtkondition von f ist von der Zerlegung unabhängig, aber die Kondition der Teilprobleme hängt von der Zerlegung ab. Eine schlechte Zerlegung kann zu instabilen Algorithmen führen.

Beispiel: $f(x) = g(h(x)), cond(f) = 1, cond(g) = 10^9, cond(h) = 10^{-9}$

Allgemeiner nimmt man an, dass Ein- und Ausgabe eines Algorithmus Vektoren von Zahlen sind, also Eingabe: $\mathbf{x} \in D \subseteq \mathbb{R}^n$, Ausgabe: $\mathbf{y} \in \mathbb{R}^m$, Algorithmus: $f : D \to \mathbb{R}^m$ mit $\mathbf{y} = f(\mathbf{x})$

Beispiele:

- Auflösen eines quadratischen linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ nach $\mathbf{x}$:
  Eingabe: $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{b} \in \mathbb{R}^n$, Ausgabe: $\mathbf{x} \in \mathbb{R}^n$
- Berechnung der Nullstellen eines Polynoms in Koeffizientendarstellung:
  Eingabe: $\mathbf{c} \in \mathbb{C}^n, c_n \neq 0$ sodass $p(z) = c_0 + c_1 z + \ldots c_n z^n$, Ausgabe: eine oder alle Nullstellen z mit $p(z) = 0$
- Berechnung der Eigenwerte und Eigenvektoren einer Matrix:
  Eingabe: $\mathbf{A} \in \mathbb{C}^{n\times n}$, Ausgabe: alle $\lambda \in \mathbb{C}, x \in \mathbb{C}^n$ mit $Ax = \lambda x$, $x \neq 0$

In obigen Definitionen müssen dann Beträge $|\cdot|$ durch geeignete Normen $\|\cdot\|$ ersetzt werden, z.B.:

- Stabilität (absolut): $\|f(\tilde{\mathbf{x}}) - \tilde{f}(\tilde{\mathbf{x}})\| \leq C \cdot eps$
- Konditionszahl (absolut): $cond_{abs}(f) = \left\|\frac{\partial f}{\partial x}\right\|$
- Konditionszahlen (relativ): $cond_{rel}(f) = \frac{\|x\|}{\|y\|}\left\|\frac{\partial f}{\partial x}\right\|$

Zusammenfassend gilt (mit absoluten Fehlern):

$$\begin{aligned} \|f(\mathbf{x}) - \tilde{f}(\tilde{\mathbf{x}})\| &= \|f(\mathbf{x}) - f(\tilde{\mathbf{x}}) + f(\tilde{\mathbf{x}}) - \tilde{f}(\tilde{\mathbf{x}})\| && (39) \\ &\leq \underbrace{\|f(\mathbf{x}) - f(\tilde{\mathbf{x}})\|}_{\text{Kondition}} + \underbrace{\|f(\tilde{\mathbf{x}}) - \tilde{f}(\tilde{\mathbf{x}})\|}_{\text{Stabilität}} && (40) \end{aligned}$$

bzw. (mit relativen Fehlern):

$$\frac{\|f(\mathbf{x}) - \tilde{f}(\tilde{\mathbf{x}})\|}{\|f(\mathbf{x})\|} \leq \underbrace{\frac{\|f(\mathbf{x}) - f(\tilde{\mathbf{x}})\|}{\|f(\mathbf{x})\|}}_{\text{Kondition}} + \underbrace{\frac{\|f(\tilde{\mathbf{x}}) - \tilde{f}(\tilde{\mathbf{x}})\|}{\|f(\mathbf{x})\|}}_{\text{Stabilität}} \quad (41)$$

2 Interpolation

Sei f eine Funktion einer Variablen x

$$f(x, a_0, \ldots, a_n) \quad (42)$$

die von $n+1$ weiteren reellen Parametern $a_0, \ldots, a_n$ abhängt.

Das *Interpolationsproblem* ist es nun, die Parameter a_i so zu bestimmen, dass für gegebene Stützstellen x_i und zugehörige Funktionswerte $f_i, i = 0, \ldots, n$, mit $x_i \neq x_j$ für $i \neq j$ gilt

$$f(x_i, a_0, \ldots, a_n) = f_i \text{ für } i = 0, \ldots n \quad (43)$$

Man bezeichnet (x_i, f_i) als *Stützpunkte*.

Bei einem *linearen Interpolationsproblem* hängt f linear von den Parametern a_i ab:

$$f(x, a_0, \ldots, a_n) = a_0 g_0(x) + a_1 g_1(x) + \ldots + a_n g_n(x) \quad (44)$$

Beispiele:

- Interpolation mit Polynomen: $g_j(x) = x^j$ (z.B.)
- trigonometrische Interpolation: $g_j(x) = e^{jix}$ $(i^2 = -1)$
- Spline-Interpolation vom Grad k: f ist $k-1$-mal stetig differenzierbar für $x \in [x_0, x_k]$ und in jedem Teilintervall $[x_i, x_{i+1}]$ stimmt f mit einem Polynom vom Grad k überein

Es gibt auch nichtlineare Interpolationsprobleme, zum Beispiel die Interpolation mit rationalen Funktionen oder durch sogenannte Exponentialsummen.

Anwendungen:

- Beispiel 1: Interpolation von Tabellenwerten, Messdaten, u.a.

  Bild mit $n+1$ Stützpunkten (x_0, f_0) bis (x_n, f_n)

  Ziel: legen einer kontinuierlichen Kurve durch die Messpunkte.

- Beispiel 2: Geometrisches Modellieren (z.B. in Computergraphik und CAD):

  Bild mit 2D Parameterbereich der durch f auf ein Flächenstück abgebilet wird

  Ziel: interaktive Modifikation der Führungspunkte → Modellierung von Oberflächen, z.B. im Automobilbau, Flugzeugbau, u.a.

- Beispiel 3: Relief- und Volumenmodellierung (Geographie, Geophysik)

  Bild eines digitalen Höhenmodells

  Ziel: Darstellung der Landschaftsoberfläche, Gesteinsschichten, etc. → „nicht-glatte“ Interpolation nötig

Weitere Anwendungen:

- Gewinnen numerischer Integrationsformeln
- Konverzengbeschleunigungsverfahren (Extrapolation)
- Zeitreihenanalyse (trigonometrische Interpolation)
- Analyse von Zerfallsreihen (Exponentialsummen)

2.1 Polynominterpolation

2.1.1 Grundlagen

Die Menge aller reellen Polynome P vom Grad $\leq n$

$$P(x) = a_0 + a_1 x + \ldots + a_n x^n \tag{45}$$

sei bezeichnet als Π_n.

Anmerkung: Π_n ist ein $n+1$-dimensionaler Vektorraum, somit lässt sich $P(x)$ als Linearkombination von Basispolynomen ϕ_i und Koeffizienten c_i darstellen

$$P(x) = c_0 + c_1\phi_1(x) + \ldots + c_n\phi_n(x) \tag{46}$$

wobei es beliebig viele Möglichkeiten der Basiswahl (abhängig von der jeweiligen Anwendung) gibt. Die Basis in obiger Definition ist die *Taylor- oder Monombasis* $\phi_i(x) = x^i$.

Satz 2.1 (Existenz und Eindeutigkeit der Polynominterpolation): Zu beliebigen $n+1$ Stützpunkten $(x_i, f_i), i = 0, \ldots, n$, mit $x_i \neq x_j$ für $i \neq j$ gibt es genau ein Polynom $P \in \Pi_n$ mit

$$P(x_i) = f_i \text{ für } i = 0, 1, \ldots, n \tag{47}$$

Beweis (eigentlich klar): Der Ansatz $P(x) = a_0 + a_1 x + \ldots + a_n x^n$ ergibt das lineare Gleichungssystem

$$a_0 + a_1 x_j + \ldots + a_n x_j^n = f_j, \text{ für } 0 \leq j \leq n \tag{48}$$

zur Bestimmung der Koeffizienten $a_0, \ldots, a_n$ des Interpolationspolynoms:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix} \tag{49}$$

Die Determinante des Gleichungssystems ist gerade die *Vandermonde-Determinante*

$$det(x_j^k)_{j,k=0,\ldots,n} = \prod_{0 \le j < l \le n} (x_l - x_j) \tag{50}$$

die wegen $x_j \neq x_l$ für $j \neq l$ von Null verschieden ist. □

(alternativer Beweis: $span(x^0, \ldots, x^n) = \Pi_n$)

Beispiel zur Vandermonde-Determinante ($n = 2$):

$$\begin{aligned} det \begin{pmatrix} 1 & x_0 & x_0^2 & x_0^2 \\ 1 & x_1 & x_1^2 & x_1^2 \\ 1 & x_2 & x_2^2 & x_2^2 \end{pmatrix} &= x_1x_2^2 + x_0x_1^2 + x_0^2x_2 - x_0^2x_1 - x_1^2x_2 - x_0x_2^2 \\ &= (x_2^2 - x_1x_2 - x_0x_2 + x_1x_0)(x_1 - x_0) = (x_2 - x_1)(x_2 - x_0)(x_1 - x_0) \end{aligned}$$

Verschiedene Wahlen von Polynombasen führen nun zu verschiedenen Interpretationen der Koffizienten.

2.1.2 Lagrange-Interpolation

Die *Lagrange-Polynome* sind definiert als

$$L_i(x) = \prod_{\substack{0 \le j \le n \\ j \neq i}} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)} \tag{51}$$

$$= \frac{w(x)}{(x - x_i)w'(x_i)} \text{ mit } w(x) = \prod_{i=0}^{n} (x - x_i) \text{ für } x \neq x_i, i = 0, \ldots, n \tag{52}$$

Beispiele: ($L_i(x)$ für $x_i = i, 0 \le i \le n$):

2 Stützpunkte ($n = 1$):

$$\begin{aligned} L_0(x) &= \frac{x - x_1}{x_0 - x_1} = \frac{x - 1}{0 - 1} = 1 - x \\ L_1(x) &= \frac{x - x_0}{x_1 - x_0} = \frac{x - 0}{1 - 0} = x \end{aligned}$$

3 Stützpunkte ($n = 2$):

$$\begin{aligned} L_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 1)(x - 2)}{(0 - 1)(0 - 2)} = \frac{1}{2}(x^2 - 3x + 2) \\ L_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x - 0)(x - 2)}{(1 - 0)(1 - 2)} = -x^2 + 2x \\ L_2(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x - 0)(x - 1)}{(2 - 0)(2 - 1)} = \frac{1}{2}(x^2 - x) \end{aligned}$$

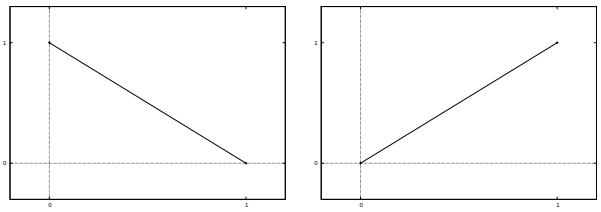

Abbildung 3: $L_0(x)$ und $L_1(x)$ für $n = 1$

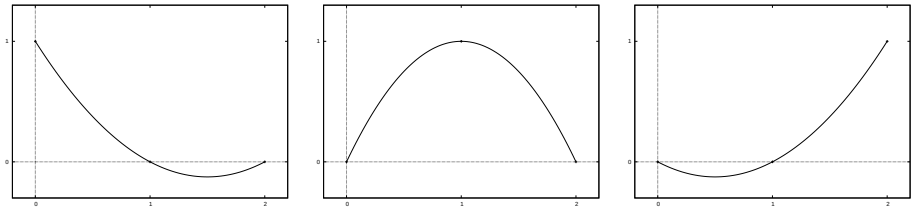

Abbildung 4: $L_0(x)$, $L_1(x)$ und $L_2(x)$ für $n = 2$

Eigenschaften der Lagrange-Polynome:

- $L_i(x) = \begin{cases} 1 & \text{für} \quad x = x_i \\ 0 & \text{für} \quad x = x_j, j \neq i, 0 \leq j \leq n \end{cases}$
- $L_i(x)$ wechselt das Vorzeichen (unduliert) an $x = x_j, 1 \leq j \leq n-1$

Die Lagrange-Polynome sind die Bausteine für die *Lagrange-Interpolationsformel*

$$f(x, f_0, \ldots, f_n) = P(x) = \sum_{i=0}^{n} f_i L_i(x) = \sum_{i=0}^{n} f_i \prod_{\substack{0 \leq j \leq n \\ j \neq i}} \frac{x - x_j}{x_i - x_j} \tag{53}$$

Auf diese Weise nehmen die Parameter $a_0, \ldots, a_n$ des linearen Interpolationsproblems gerade die Werte f_i, d.h. die Funktionswerte an den Stützstellen, an. Die Lagrange-Polynome $L_0, \ldots, L_n$ bilden wie die Taylor-Polynome eine Basis von Π_n, denn sie sind aufgrund ihrer Definition offensichtlich linear unabhängig.

Die direkte Auswertung des Interpolationspolynoms in der Lagrange-Darstellung ist jedoch mühsam (der Aufwand ist von der Ordnung $O(n^2)$, also quadratisch in der Zahl der Stützstellen).

Beispiele (Lagrange-Interpolationsformel für $x_i = i$):

2 Stützstellen ($n = 1$):

$$P(x) = f_0 L_0(x) + f_1 L_1(x) = f_0(1 - x) + f_1(x) = (f_1 - f_0)x + f_0 \tag{54}$$

damit gilt: $P(0) = f_0$ und $P(1) = f_1$ wobei P ein lineares Polynom ist.

Setze $f_0 = 1$ und $f_1 = 2 \rightarrow P(x) = x + 1$

3 Stützstellen ($n = 2$):

$$P(x) \quad = \quad f_0 L_0(x) + f_1 L_1(x) + f_2 L_2(x) =$$

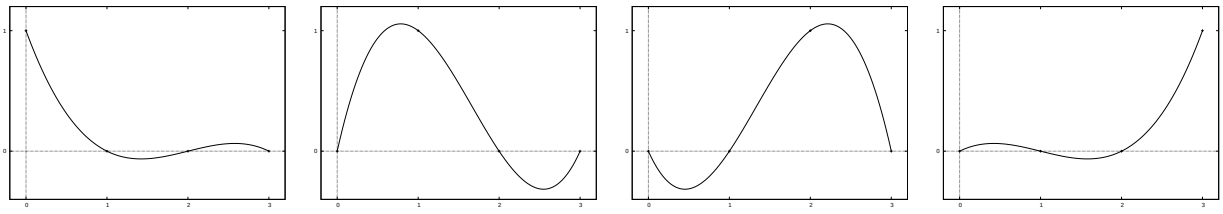

Abbildung 5: $L_0(x)$, $L_1(x)$, $L_2(x)$ und $L_3(x)$ für $n = 3$

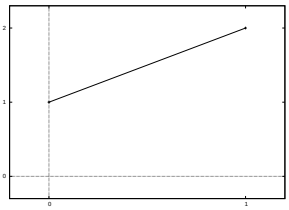

Abbildung 6: Interpolationspolynom $P(x)$ durch die Punkte (0,1) und (1,2)

$$
\begin{aligned}
&= f_0\frac{1}{2}(x^2 - 3x + 2) + f_1(-x^2 + 2x) + f_2\frac{1}{2}(x^2 - x) = \\
&= \frac{1}{2}(f_0 - 2f_1 + f_2)x^2 + \frac{1}{2}(-3f_0 + 4f_1 - f_2)x + f_0
\end{aligned}
$$

damit gilt: $P(0) = f_0$, $P(1) = f_1$ und $P(2) = f_2$ wobei P ein quadratisches Polynom ist.

Setze $f_0 = 1$, $f_1 = 3$ und $f_2 = 2 \rightarrow P(x) = -\frac{3}{2}x^2 + \frac{7}{2}x + 1$

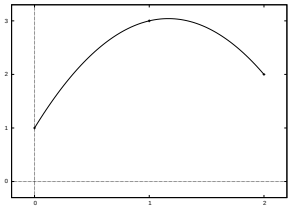

Abbildung 7: Interpolationspolynom $P(x)$ durch die Punkte (0,1), (1,3) und (2,2)

2.1.3 Aitken-Neville-Algorithmus

Die Idee des Aitken-Neville-Algorithmus ist die Lösung des Interpolationsproblems schrittweise aus den Lösungen für weniger Punkte aufzubauen.

Satz 2.2 (Rekursionsformel für Polynome von Aitken): Seien Stützwerte $(x_i, f_i), i = 0, \ldots, n$ gegeben und bezeichne $P_{m,m+1,\ldots,m+k} \in \Pi_k$ das Polynom aus Π_k mit

$$P_{m,m+1,\ldots,m+k}(x_j) = f_j, \text{ für } j = m, \ldots, k \tag{55}$$

dann gilt die Rekursionsformel

$$P_i(x) \equiv f_i \tag{56}$$

$$P_{m,m+1,\ldots,m+k}(x) \equiv \frac{(x - x_m)P_{m+1,m+2,\ldots,m+k}(x) - (x - x_{m+k})P_{m,m+1,\ldots,m+k-1}(x)}{x_{m+k} - x_m} \tag{57}$$

Beweis: Bezeichne $R(x)$ die rechte Seite der Rekursionsformel und zeige dass R die Interpolationseigenschaft von $P_{m,\ldots,k}$ besitzt:

- $Grad(R) \leq k$
- nach Definition von $P_{m+1,\ldots,k}$ und $P_{m,\ldots,k-1}$ ist

$$\begin{aligned} R(x_m) &= P_{m,\ldots,m+k-1}(x_m) = f_m \\ R(x_{m+k}) &= P_{m+1,\ldots,m+k}(x_{m+k}) = f_k \\ R(x_j) &= \frac{(x_j - x_m)f_j - (x_j - x_{m+k})f_j}{x_{m+k} - x_m} = f_j \text{ für } j = m+1, m+2, \ldots, k-1 \end{aligned}$$

Mit der Eindeutigkeit der Polynominterpolation gilt somit $R = P_{m,\ldots,m+k}$ □

Beispiel: (Aitken-Rekursionsformel für $n = 2$):

$$\begin{aligned} P_0(x) &= f_0 \\ P_1(x) &= f_1 \\ P_2(x) &= f_2 \\ P_{01}(x) &= [(x - x_0)P_1(x) - (x - x_1)P_0(x)]/(x_1 - x_0) = \\ &= [(x-0)f_1 - (x-1)f_0]/(1-0) = (f_1 - f_0)x + f_0 \\ P_{12}(x) &= [(x - x_1)P_2(x) - (x - x_2)P_1(x)]/(x_2 - x_1) = \\ &= [(x-1)f_2 - (x-2)f_1]/(2-1) = (f_2 - f_1)x + (2f_1 - f_2) \\ P_{012}(x) &= [(x - x_0)P_{12}(x) - (x - x_2)P_{01}(x)]/(x_2 - x_0) = \\ &= [(x-0)((f_2 - f_1)x + (2f_1 - f_2))] - (x-2)((f_1 - f_0)x + f_0)]/(2-0) = \\ &= \frac{1}{2}(f_0 - 2f_1 + f_2)x^2 + \frac{1}{2}(-3f_0 + 4f_1 - f_2)x + f_0 \end{aligned}$$

Der Algorithmus von Aitken-Neville konstruiert mit dieser Rekursionsformel ein Tableau, das die Werte der interpolierten Polynome $P_{i,i+1,\ldots,i+k}$ an einem festen Punkt x enthält:

$$\begin{array}{c|cccc} & k=0 & k=1 & k=2 & k=3 \\ x_0 & f_0 = P_0(x) & & & \\ & & P_{01}(x) & & \\ x_1 & f_1 = P_1(x) & & P_{012}(x) & \\ & & P_{12}(x) & & P_{0123}(x) \\ x_2 & f_2 = P_2(x) & & P_{123}(x) & \\ & & P_{23}(x) & & \\ x_3 & f_3 = P_0(x) & & & \end{array} \tag{58}$$

- der Algorithmus wird mit den Funktionswerten in Spalte 0 initialisiert
- die Werte in den Spalten $k > 0$ werden der Reihe nach mit Hilfe der Rekursionsformel aus ihren zwei linken Nachbarn errechnet, z.B.

$$P_{123}(x) = \frac{(x - x_1)P_{23}(x) - (x - x_3)P_{12}(x)}{x_3 - x_1} \tag{59}$$

Beispiel: (Aitken-Neville-Algorithmus für $n = 2$, Stützstellen: $(0,1),(1,3),(2,2)$, Auswerten bei $x = \frac{1}{2}$):

$$\begin{array}{c|ccc} & k=0 & k=1 & k=2 \\ x_0 & 1 & & \\ & & 2 & \\ x_1 & 3 & & \frac{19}{8} \\ & & \frac{7}{2} & \\ x_2 & 2 & & \end{array} \tag{60}$$

$$\begin{aligned}
P_{01}\left(\frac{1}{2}\right) &= \left[\left(\frac{1}{2}-x_0\right)P_1\left(\frac{1}{2}\right) - \left(\frac{1}{2}-x_1\right)P_0\left(\frac{1}{2}\right)\right] / (x_1 - x_0) = \\
&= \left[\left(\frac{1}{2}-0\right)3 - \left(\frac{1}{2}-1\right)1\right] / (1-0) = \frac{3}{2} + \frac{1}{2} = 2 \\
P_{12}\left(\frac{1}{2}\right) &= \left[\left(\frac{1}{2}-x_1\right)P_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}-x_2\right)P_1\left(\frac{1}{2}\right)\right] / (x_2 - x_1) = \\
&= \left[\left(\frac{1}{2}-1\right)2 - \left(\frac{1}{2}-2\right)3\right] / (2-1) = -1 + \frac{9}{2} = \frac{7}{2} \\
P_{012}\left(\frac{1}{2}\right) &= \left[\left(\frac{1}{2}-x_0\right)P_{12}\left(\frac{1}{2}\right) - \left(\frac{1}{2}-x_2\right)P_{01}\left(\frac{1}{2}\right)\right] / (x_2 - x_0) = \\
&= \left[\left(\frac{1}{2}-0\right)\frac{7}{2} - \left(\frac{1}{2}-2\right)2\right] / (2-0) = \left(\frac{7}{4}+3\right)/2 = \frac{19}{8}
\end{aligned}$$

(vgl.: $-\frac{3}{2}(\frac{1}{2})^2 + \frac{7}{2}\frac{1}{2} + 1 = -\frac{3}{8} + \frac{7}{4} + 1 = \frac{19}{8}$)

Anmerkungen:

- der Aitken-Neville-Algorithms stellt das Interpolationspolynom nicht explizit auf, sondern wertet es nur an der Stelle x aus
- der Aufwand ist immer noch quadratisch in der Zahl der Stützstellen, es werden aber viel weniger Operationen benötigt, als eine direkte Auswertung in der Lagrange-Darstellung
- der Algorithmus ist jedoch unpraktisch, wenn man ein Interpolationspolynom an mehreren Punkten auswerten will

2.1.4 Newton-Algorithmus

Der Aitken-Neville-Algorithmus ist unpraktisch,

- wenn man ein Interpolationspolynom an mehreren Stellen auswerten möchte oder
- wenn man Stützstellen nachträglich hinzufügen möchte

Der *Newton-Algorithmus* behebt diese beiden Probleme durch den Ansatz:

$$P(x) = P_{01\ldots n}(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \ldots + a_n(x-x_0)\ldots(x-x_{n-1}) \tag{61}$$

für das interpolierende Polynom $P \in \Pi_n$ mit $P(x_i) = f_i$ für $i = 0, \ldots, n$.

Die Auswertung von P an der Stelle $x = \xi$ ist dann mit dem *Horner-Schema* effizient möglich:

$$P(\xi) = (\dots(a_n(\xi - x_{n-1}) + a_{n-1})(\xi - x_{n-2}) + \dots a_1)(\xi - x_0) + a_0 \tag{62}$$

Beispiel für die Funktionsweise des Horner-Schemas:

$$4x^3 + 3x^2 + 2x + 1 = ((4x + 3)x + 2)x + 1 \tag{63}$$

Im Newton-Algorithmus werden die Koeffizienten $a_i, 0 \leq i \leq n$, sukzessive aus den folgenden Beziehungen ermittelt:

$$\begin{aligned} f_0 &= P(x_0) = a_0 \\ f_1 &= P(x_1) = a_0 + a_1(x_1 - x_0) \\ f_2 &= P(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \end{aligned}$$

Der Aufwand des Newton-Algorithmus ist quadratisch in n, die Auswertung des Interpolationspolynoms an einer Stelle ξ ist aber nur noch linear in n.

Beispiel: ($n = 2$, Stützstellen: $(0, 1), (1, 3), (2, 2)$)

$$\begin{aligned} 1 &= a_0 \Rightarrow a_0 = 1 \\ 3 &= a_0 + a_1(x_1 - x_0) = 1 + a_1(1 - 0) \Rightarrow a_1 = 2 \\ 2 &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = 1 + 2(2 - 0) + a_2(2 - 0)(2 - 1) \Rightarrow a_2 = -\frac{3}{2} \\ &\Rightarrow P(x) = 1 + 2(x - 0) - \frac{3}{2}(x - 0)(x - 1) = 1 + \frac{7}{2}x - \frac{3}{2}x^2 \end{aligned}$$

Auswerten bei $\xi = \frac{1}{2}$ mit dem Horner-Schema:

$$P(\xi) = (a_2(\xi - x_1) + a_1)(\xi - x_0) + a_0 = (-\frac{3}{2}(\frac{1}{2} - 1) + 2)(\frac{1}{2} - 0) + 1 = (\frac{3}{4} + 2)(\frac{1}{2} - 0) + 1 = \frac{11}{8} + 1 = \frac{19}{8} \tag{64}$$

2.1.5 Dividierte Differenzen

Die Polynome $P_{m,m+1,\dots,m+k-1}(x)$ und $P_{m,m+1,\dots,m+k}(x)$ unterscheiden sich um ein Polynom vom Grad k mit den Nullstellen $x_m, x_{m+1}, \dots, x_{m+k-1}$ da beide Polynome an diesen Stellen den gleichen Wert annehmen.

Bezeichne mit $f_{m,m+1,\dots,m+k}$ den eindeutig bestimmten Koeffizienten von x^k des Polynoms $P_{m,m+1,\dots,m+k}(x)$, so gilt:

$$P_{m,m+1,\dots,m+k}(x) = P_{m,m+1,\dots,m+k-1}(x) + f_{m,m+1,\dots,m+k}(x - x_m)(x - x_{m+1}) \dots (x - x_{m+k-1}) \tag{65}$$

Auf diese Weise erhält man die *Newton-Darstellung* des Interpolationspolynoms $P_{01\dots k}(x)$:

$$P_{01\dots k}(x) = f_0 + f_{01}(x - x_0) + \dots + f_{01\dots k}(x - x_0)(x - x_1) \dots (x - x_{k-1}) \tag{66}$$

Die Zahlen $f_{m,\dots,m+k}$ heissen die k-ten *dividierten Differenzen*.

Satz 2.3: Für die dividierten Differenzen gilt die Rekursionsformel:

$$f_{m,\dots,m+k} = \frac{f_{m+1,\dots,m+k} - f_{m,\dots,m+k-1}}{x_{m+k} - x_m} \tag{67}$$

Beweis: Koeffizientenvergleich von x^k auf beiden Seiten in der Rekursionsformel von Aitken. □

Dividierte Differenzen sind invariant gegenüber Permutation der Indizes $m, m+1, \ldots, m+k$.

Dividierte Differenzen mit aufeinander folgenden Indizes erlauben das sogenannte *Differenzenschema*:

$$\begin{array}{c|cccc} & k=0 & k=1 & k=2 & \\ x_0 & f_0 & & & \\ & & f_{01} & & \\ x_1 & f_1 & & f_{012} & \ddots \\ & & f_{12} & & \\ x_2 & f_2 & & \ddots & \\ & & \ddots & & \\ \vdots & \vdots & & & \end{array} \tag{68}$$

Analog zum Aitken-Neville-Algorithmus lassen sich die Einträge spaltenweise über die Rekursionsformel

$$f_{i,i+1,\ldots,i+k} = \frac{f_{i+1,\ldots,i+k} - f_{i,\ldots,i+k-1}}{x_{i+k} - x_i} \tag{69}$$

berechnen. Aus der obersten Schrägzeile kann man direkt die Koeffizienten der Newton-Darstellung des Interpolationspolynoms ablesen.

Beispiel: ($n = 2$, Stützstellen: $(0,1), (1,3), (2,2)$)

$$\begin{array}{c|ccc} & k=0 & k=1 & k=2 \\ x_0 & 1 & & \\ & & 2 & \\ x_1 & 3 & & -\frac{3}{2} \\ & & -1 & \\ x_2 & 2 & & \end{array} \tag{70}$$

$$\begin{aligned} f_{01} &= \frac{f_1 - f_0}{x_1 - x_0} = \frac{3-1}{1-0} = 2 \\ f_{12} &= \frac{f_2 - f_1}{x_2 - x_1} = \frac{2-3}{2-1} = -1 \\ f_{012} &= \frac{f_{12} - f_{01}}{x_2 - x_0} = \frac{(-1)-2}{2-0} = -\frac{3}{2} \end{aligned}$$

und das resultierende Newton-Interpolationspolynom

$$P_{012}(x) = f_0 + f_{01}(x - x_1) + f_{012}(x - x_0)(x - x_1) = 1 - 2(x-1) - \frac{3}{2}(x-0)(x-1) \tag{71}$$

lässt sich analog zu oben mit dem Horner-Schema auswerten.

Algorithmus 2.5 (Bestimmung der Polynomkoeffizienten in Newton-Darstellung mit Dividierten Differenzen):

```
for i:=0 to n
  t[i]:=f[i];
  for j:=i-1 to 0 step -1
    t[j]:=(t[j+1]-t[j])/(x[i]-x[j]);
  a[i]:=t[0];
```

Algorithmus 2.6 (Auswertung des Interpolationspolynoms in Newton-Darstellung mit dem Horner-Schema):

```
p:=a[n];
for i:=n-1 to 0 step -1
  p:=p*(x-x[i])+a[i];
```

2.1.6 Interpolationsfehler

Sind die f_i die Werte einer Funktion f, d.h. $f(x_i) = f_i, 0 \leq i \leq n$, die interpoliert werden soll, dann entsteht die Frage:

Wie nah ist das interpolierende Polynom

$$P(x) = P_{0\ldots n}(x) \in \Pi_n \text{ mit } P(x_i) = f_i \text{ für } 0 \leq i \leq n \tag{72}$$

an der Funktion f an Stellen $x \neq x_i$ dran?

Es ist klar, dass der Fehler bei geeigneter Wahl von f beliebig groß sein kann, wenn nicht weitere Forderungen an f gestellt werden. Mit zusätzlichen Bedingungen an f sind *Fehlerabschätzungen* möglich.

Satz 2.7: Ist f $(n+1)$-mal differenzierbar, so gibt es zu jedem $\bar{x}$ eine Zahl ξ aus dem kleinsten Intervall $I[x_0, \ldots x_n, \bar{x}]$, das alle x_i und $\bar{x}$ enhält, so dass

$$f(\bar{x}) - P_{0\ldots n}(\bar{x}) = \frac{w(\bar{x}) f^{(n+1)}(\xi)}{(n+1)!} \tag{73}$$

gilt, wobei $w(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$.

Beweis: Betrachte mit $P(x) := P_{0\ldots n}(x)$ für ein beliebiges festes $\bar{x} \neq x_i, 0 \leq i \leq n$, die Funktion

$$F(x) = f(x) - P(x) - kw(x) \tag{74}$$

und bestimme k so, dass F an der Stelle $x = \bar{x}$ verschwindet. Dann besitzt F in $I[x_0, \ldots x_n, \bar{x}]$ mindestens die $n+2$ Nullstellen $x_0, \ldots x_n, \bar{x}$. Nach dem Satz von Rolle besitzt dann $F'(x)$ mindestens $n+1$ Nullstellen, $F''(x)$ mindestens n Nullstellen, ... und $F^{(n+1)}(x)$ mindestens eine Nullstelle $\xi \in I[x_0, \ldots x_n, \bar{x}]$.

Nun ist aber $P^{(n+1)}(x) \equiv 0$, also

$$F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - k(n+1)! = 0 \tag{75}$$

bzw.

$$k = \frac{f^{(n+1)}(\xi)}{(n+1)!} \tag{76}$$

und damit

$$f(\bar{x}) - P(\bar{x}) = kw(\bar{x}) = \frac{w(\bar{x})}{(n+1)!} f^{(n+1)}(\xi) \tag{77}$$

□

Eine andere Fehlerdarstellung ergibt sich aus der Newton-Interpolationsformel. Die dividierten Differenzen $f_{0\ldots k}$ lassen sich als Funktionen der Argumente x_j auffassen, für die historisch die Bezeichnung $f[x_0, x_1, \ldots, x_k]$ üblich ist. Wählt man zusätzlich zu den $n+1$ Stützstellen (x_i, f_i) eine $(n+2)$-te Stützstelle (x_{n+1}, f_{n+1}), sodass $x_{n+1} = \bar{x}, f_{n+1} = f(\bar{x})$ und $\bar{x} \neq x_i, 0 \leq i \leq n$, dann folgt

$$f(\bar{x}) = P_{0\ldots n+1}(\bar{x}) = P_{0\ldots n}(\bar{x}) + f[x_0, \ldots, x_n, \bar{x}] w(\bar{x}) \tag{78}$$

bzw.

$$f(\bar{x}) - P_{0\ldots n}(\bar{x}) = f[x_0, \ldots, x_n, \bar{x}]w(\bar{x}) \tag{79}$$

Aus dem Vergleich mit der Aussage von Satz 2.7 folgt:

$$f[x_0, \ldots, x_n, \bar{x}] = \frac{f^{(n+1)}(\xi)}{(n+1)!} \tag{80}$$

für ein $\xi \in I[x_0, \ldots, x_n, \bar{x}]$. Damit gilt generell für die n-te dividierte Differenz

$$f[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!} \tag{81}$$

für ein $\xi \in I[x_0, \ldots x_n]$.

Beispiel: $f(x) = \sin x, n = 6, x_i = \frac{\pi}{10} \cdot i, 0 \leq i \leq 5$:

$$\begin{aligned} \sin x - P(x) &= (x - x_0)(x - x_1)\ldots(x - x_5)\frac{-\sin\xi}{720}, \xi = \xi(x) \\ |\sin x - P(x)| &\leq \frac{1}{720}|(x - x_0)(x - x_1)\ldots(x - x_5)| = \frac{|w(x)|}{720} \end{aligned}$$

Anmerkungen:

- Außerhalb des Intervalls $I[x_0, \ldots x_n]$ wächst $|w(x)|$ sehr schnell an, eine Verwendung des Interpolationspolynoms P zur Approximation von f an einer Stelle $\bar{x}$ außerhalb von $I[x_0, \ldots, x_n]$ (sogenannte *Extrapolation*) sollte möglichst vermieden werden.
- Man sollte nicht annehmen, dass man mit wachsender Zahl von Stützstellen n innerhalb eines festen Intervalls $[a, b]$ immer bessere Approximationen an f erhält. Sei Δ_m eine Folge von Stützpunkten

  $$\Delta_m = \{a = x_{m,0} < x_{m,1} < \ldots < x_{m,n_m} = b\} \tag{82}$$

  und P_m eine entsprechende Folge von Interpolationspolynomen. Man kann zeigen, dass man zu jeder Folge Δ_m von Stützpunkten auf $[a, b]$ eine auf $[a, b]$ stetige Funktion f finden kann, sodass die Interpolationspolynome P_m für $m \to \infty$ auf $[a, b]$ *nicht* gleichmäßig gegen f konvergieren.

Weitere Verfahren:

- Bei der *Hermite-Interpolation* werden zusätzlich zu den Funktionswerten f_i Ableitungen von f vorgeschrieben. Die Stützstellen sind dann:

  $$(x_i, f_i^{(k)}) \text{ für } 0 \leq i \leq m, 0 \leq k \leq n_i - 1 \tag{83}$$

  womit genau ein Polynom $P \in \Pi_n$ mit $n + 1 = n_1 + \ldots + n_m$, das die Interpolationsvorschriften

  $$P^{(k)}(x_i) = f_i^{(k)} \text{ für } 0 \leq i \leq m, 0 \leq k \leq n_i - 1 \tag{84}$$

  erfüllt, bestimmt werden kann. Es gibt auch eine verallgemeinerte Lagrange-Darstellung des Interpolationspolynoms

  $$P^{(k)}(x) = \sum_{i=0}^{m} \sum_{k=0}^{n_i - 1} f_i^{(k)} L_{ik}(x) \tag{85}$$

  sowie eine verallgemeinerte Newton-Darstellung

  $$P^{(k)}(x) = \sum_{i=0}^{n} f[t_0, \ldots, t_i] \prod_{j=0}^{i-1} (x - t_i) \tag{86}$$

wobei

$$(t_0, \ldots t_n) = (\underbrace{x_0, \ldots, x_0}_{n_0 - mal}, \ldots \underbrace{x_m, \ldots, x_m}_{n_m - mal}) \tag{87}$$

und verallgemeinte Aitken-Neville-, Newton- und Dividierte Differenzen-Algorithmen.

- Unter *rationaler Interpolation* versteht man die Interpolation mit rationalen Funktionen

$$\frac{P(x)}{Q(x)} = \frac{a_0 + a_1 x + \ldots + a_n x^n}{b_0 + b_1 x + \ldots + b_m x^m} \tag{88}$$

womit (bis zu) $n + m + 1$ Interpolationsbedingungen

$$\frac{P(x_i)}{Q(x_i)} = f_i, 0 \leq i \leq n + m + 1 \tag{89}$$

erfüllt werden können. Auch hier existieren zur Bestimmung der Polynomkoeffizienten von P und Q Aitken-Neville- und Newton-artige Algorithmen. Die rationale Interpolation zählt zu den nichtlinearen Interpolationsproblemen.

2.2 Trigonometrische Interpolation

Fourier-Analyse: Zerlegung eines zeitlich variablen Mess-Signals (Funktion) $f(t)$ in seine spektralen Anteile.

Varianten:

Name	Definitionsgebiet	Periodizität	Frequenzspektrum	Kap.
Fourier-Reihe	$f : [-T/2, T/2] \to \mathbb{R}$	periodisch	diskret	2.2.1
kontinuierliche Fourier-Transformation	$f : \mathbb{R} \to \mathbb{R}$	aperiodisch	kontinuierlich	2.2.2
diskrete Fourier-Transformation	$f \in \mathbb{R}^n$	periodisch	diskret, endlich	2.2.3

2.2.1 Fourier-Reihen

Die Funktion $f(t)$ sei reell mit Periode T, dann ist die *Fourier-Reihe* von f (komplexe Schreibweise)

$$f(t) = \sum_{k=-\infty}^{\infty} C_k e^{i\omega_k t} \text{ mit } \omega_k = \frac{2\pi k}{T} \tag{90}$$

und

$$C_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-i\omega_k t} \, dt \tag{91}$$

bzw. (reelle Schreibweise)

$$f(t) = \sum_{k=0}^{\infty} (A_k \cos(\omega_k t) + B_k \sin(\omega_k t)) \tag{92}$$

mit

$$A_k = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(\omega_k t) \, dt \text{ für } k \neq 0 \text{ und } A_0 = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \, dt \tag{93}$$

$$B_k = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(\omega_k t) \, dt \text{ für } k \neq 0 \text{ und } B_0 = 0 \tag{94}$$

Vorraussetzungen für die Konvergenz der Fourier-Reihe:

Konvergenzart	Vorraussetzung
punktweise + gleichmäßig	f stetig und stückweise stetig differenzierbar
punktweise	f hat beschränkte Variation und linke+rechte Grenzwerte stimmen mit dem Mittel überein
im L_2-Sinn	$f \in L_2$

Satz 2.8 (Eigenschaften der Fourier-Reihe): Sei $\{C_k\}$ die Fourier-Reihe von $f(t)$ und $\{D_k\}$ die Fourier-Reihe von $g(t)$, dann gilt:

a)	Linearität:	Die Fourier-Reihe von $h(t) = af(t) + bg(t)$ ist $\{aC_k + bD_k\}$
b)	Verschiebung (Zeit):	Die Fourier-Reihe von $f(t-a)$ ist $\{C_k e^{-i\omega_k a}\}$
c)	Verschiebung (Frequenz):	Die Fourier-Reihe von $f(t)e^{i(2\pi at)/T}$ ist $\{C_{k-a}\}$
d)	Skalierung:	Die Fourier-Reihe von $f(at)$ ist $\{C_k\}$ bezüglich der Periode $\frac{\omega_k}{a}$

Beweis:

a) folgt direkt aus der Linearität der Fourier-Reihe

b) $C_k^{neu} = \frac{1}{T}\int_{-T/2}^{T/2} f(t-a)e^{-i\omega_k t}\ dt \overset{(s=t-a)}{=} \frac{1}{T}\int_{-T/2-a}^{T/2-a} f(s)e^{-i\omega_k s}e^{-i\omega_k a}\ ds = e^{-i\omega_k a}C_k^{alt}$

c) Umkehrung von b)

d) $C_k^{neu} = \frac{a}{T}\int_{-T/2a}^{T/2a} f(at)e^{-i\omega_k t}\ dt \overset{(s=at)}{=} \frac{a}{T}\int_{-T/2}^{T/2} f(s)e^{-i\omega_k s/a}\frac{1}{a}\ ds = C_k^{alt}$ mit $\omega_k^{neu} = \omega_k^{alt}/a$

Die *N-te Partialsumme* einer Fourier-Reihe ist definiert als

$$S_N(t) = \sum_{k=0}^{N}(A_k\cos(\omega_k t) + B_k\sin(\omega_k t)) \tag{95}$$

Satz 2.9 (Abbruchfehler): Für den mittleren quadratischen *Abbruchfehler*

$$\delta_N^2(t) = \frac{1}{T}\int_{-T/2}^{T/2}(f(t) - S_N(t))^2\ dt \tag{96}$$

gilt

$$\delta_N^2(t) = \frac{1}{T}\int_{-T/2}^{T/2} f^2(t)\ dt - A_0^2 - \frac{1}{2}\sum_{k=1}^{N}(A_k^2 + B_k^2) \tag{97}$$

Der mittlere quadratische Fehler wird also mit zunehmendem N *monoton* kleiner!

Beweis:

$$\begin{aligned}
\delta_N^2(t) &= \frac{1}{T}\left(\int_{-T/2}^{T/2} f^2(t)\ dt - 2\int_{-T/2}^{T/2} f(t)S_N(t)\ dt + \int_{-T/2}^{T/2} S_N^2(t)\ dt\right) = \\
&= \frac{1}{T}\left(\int_{-T/2}^{T/2} f^2(t)\ dt - 2\int_{-T/2}^{T/2}\sum_{k=0}^{\infty}(A_k\cos\omega_k t + B_k\sin\omega_k t)\sum_{k=0}^{N}(A_k\cos\omega_k t + B_k\sin\omega_k t)\ dt + \right. \\
&\quad \left. + \int_{-T/2}^{T/2}\sum_{k=0}^{N}(A_k\cos\omega_k t + B_k\sin\omega_k t)\sum_{k=0}^{N}(A_k\cos\omega_k t + B_k\sin\omega_k t)\ dt\right) =
\end{aligned}$$

$$
\begin{aligned}
&\text{(Orthogonalität von } \sin\omega_k t \text{ und } \cos\omega_k t \text{)} \\
= \quad & \frac{1}{T}\left(\int_{-T/2}^{T/2} f^2(t)\ dt - 2TA_0^2 - 2\frac{T}{2}\sum_{k=1}^{N}(A_k^2+B_k^2) + TA_0^2 + \frac{T}{2}\sum_{k=1}^{N}(A_k^2+B_k^2)\right) = \\
= \quad & \frac{1}{T}\int_{-T/2}^{T/2} f^2(t)\ dt - A_0^2 - \frac{1}{2}\sum_{k=1}^{N}(A_k^2+B_k^2)
\end{aligned}
$$

□

Korollar (Bessel-Ungleichung): Es gilt

$$\frac{1}{T}\int_{-T/2}^{T/2} f^2(t)\ dt \geq A_0^2 + \frac{1}{2}\sum_{k=1}^{N}(A_k^2+B_k^2) \tag{98}$$

Beweis: Satz 2.9 und Positivität von $\delta_N^2(t)$

Korollar (Parseval-Gleichung): Es gilt

$$\frac{1}{T}\int_{-T/2}^{T/2} f^2(t)\ dt = A_0^2 + \frac{1}{2}\sum_{k=1}^{\infty}(A_k^2+B_k^2) \tag{99}$$

d.h. das mittlere quadratusche Signal (Informationsgehalt) bleibt erhalten.

Beweis: Satz 2.9 für $N \to \infty$

Eigenschaften der Fourier-Reihe:

- glatte Funktionen (C^∞) sind bandbreiten-limitiert, d.h. die Fourier-Reihe bricht nach einer endlichen Zahl von Gliedern ab.
- Funktionen mit Knick (C^0) benötigen unendlich viele Reihenglieder, die aber mit N abfallen.
- unstetige Funktionen können auch mit Hilfe unendlich vieler Reihenglieder dargestellt werden und der mittlere quadratische Fehler nimmt monoton ab, aber es treten sogenannte Über- und Unterschwinger auf (Gibbs-Phänomen).

  Beispiel: Stufe

Bild der Approximation einer Stufenfunktion von -1 nach 1 mit 1,2,3,4 Reihengliedern

  Es gilt: $\lim_{N\to\infty} S_n(t_1) = \frac{1}{T}\int_{-T/2}^{T/2} \frac{\sin t}{t}\ dt - \frac{1}{2} \approx 0.089490$ am 1. Extremum > 0, $t_1 = T/2N$. Der relative Fehler vom ca. 18% ist unabhängig von N!

2.2.2 Kontinuierliche Fourier-Transformation

Wir betrachten nun den Übergang von periodischen zu nicht-periodischen Funktionen und von der Reihe zum Integral.

Die *Fourier-Transformation* (Vorwärts-Transformation, Fourier-Analyse) $FT(f(t))$ einer reellen Funktion $f(t)$ ist definiert durch

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\ dt \tag{100}$$

die inverse Fourier-Transformation $IFT(F(\omega))$ (Rückwärts-Transformation, Fourier-Synthese) entsprechend durch

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, d\omega \tag{101}$$

Reelle Darstellung:

$$f(t) = \frac{1}{\pi} \int_{0}^{\infty} c(\omega) \cos \omega t + s(\omega) \sin \omega t \, d\omega \tag{102}$$

$$c(\omega) = \int_{-\infty}^{\infty} f(t) \cos \omega t \, dt \tag{103}$$

$$s(\omega) = \int_{-\infty}^{\infty} f(t) \sin \omega t \, dt \tag{104}$$

Hinreichende Vorraussetzung für die Existenz der FT ist $f \in L_1$, d.h. $\int_{\infty}^{\infty} |f(t)| dt < \infty$, dann folgt: $F(\omega)$ ist stetig und beschränkt.

Die *δ-Funktion* ist eine Distribution (verallgemeinerte Funktion) definiert durch

$$\delta(t) = \lim_{a \to \infty} f_a(t) \text{ mit } f_a(t) = \begin{cases} a & \text{für } -1/2a \leq t \leq 1/2a \\ 0 & \text{sonst} \end{cases} \tag{105}$$

Bild Delta-Funktion (bzw. f_a) für $a = 1, 2, 4, 8$

Sie hat die Eigenschaften

$$\int_{-\infty}^{\infty} f(t)\delta(t) \, dt = f(0) \tag{106}$$

und

$$\delta(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega t} \, dt \tag{107}$$

Satz 2.10 (Inverse Fourier-Transformation): Falls FT, IFT existieren, dann gilt

$$IFT(FT(f(t))) = f(t) \tag{108}$$

Beweis:

$$\begin{aligned} f(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s) e^{i\omega s} e^{i\omega t} \, ds \, d\omega = \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} f(s) \, ds \int_{-\infty}^{\infty} e^{i\omega(t-s)} \, d\omega = \int_{-\infty}^{\infty} f(s)\delta(t-s) \, ds = f(t) \end{aligned}$$

□

Beispiele:

Rechteck-Funktion	$f(t) = \begin{cases} 1 & \text{für } -T/2 \leq t \leq T/2 \\ 0 & \text{sonst} \end{cases}$	$\Rightarrow F(\omega) = T\frac{\sin(\omega T/2)}{\omega T/2}$
Gauß-Funktion	$f(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-t^2/2\sigma^2}$	$\Rightarrow F(\omega) = e^{-\sigma^2\omega^2/2}$
Exponentialfunktion	$f(t) = e^{-\lvert t \rvert/\tau}$	$\Rightarrow F(\omega) = \frac{2\tau}{1+\omega^2\tau^2}$

Satz 2.11 (Eigenschaften der Fourier-Transformation):

Sei $F(\omega) = FT(f(t))$ und $G(\omega) = FT(g(t))$, dann gilt:

a)	Linearität:	$FT(af(t) + bg(t))$	$=$	$aF(\omega) + bG(\omega)$
b)	Verschiebung (Zeit):	$FT(f(t-a))$	$=$	$F(\omega)e^{-i\omega a}$
c)	Verschiebung (Frequenz):	$FT(f(t)e^{-i\omega_0 t})$	$=$	$F(\omega + \omega_0)$
d)	Skalierung:	$FT(f(at))$	$=$	$\frac{1}{\lvert a\rvert}F(\frac{\omega}{a})$

Beweis: analog zu Satz 2.9. □

Die *Faltung* einer Funktion $f(t)$ mit einer Funktion $g(t)$ ist definiert als

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(\xi)g(t-\xi)\ d\xi \tag{109}$$

Die Faltung ist

a)	kommutativ	$f * g = g * f$	(Substitution)
b)	distributiv	$f * (g * h) = f * g + f * h$	(wegen Linearität)
c)	assoziativ:	$f * (g * h) = (f * g) * h$	(wegen Vertauschbarkeit der Integration)

Beispiel:

$$f(t) = \begin{cases} 1 & -T/2 \le t \le T/2 \\ 0 & \text{sonst} \end{cases},\ g(t) = \begin{cases} 1 & 0 \le t \le T \\ 0 & \text{sonst} \end{cases}, h(t) = f(t) * g(t) \tag{110}$$

Bild zur Faltung: $f(\xi)$ und $g(t-\xi)$ für $t = 0$ und t variabel zwischen $-T/2$ und $3T_2$ sowie $h(t)$

Satz 2.12 (Faltungssatz): Sei $F(\omega) = FT(f(t)), G(\omega) = FT(g(t))$ und $H(\omega) = FT(h(t))$, dann gilt für $h(t) = f(t) * g(t)$

$$H(\omega) = F(\omega) \cdot G(\omega) \tag{111}$$

d.h. aus dem Faltungsintegral wird durch die Fourier-Transformation ein Produkt von Fourier-Transformierten.

Beweis:

$$\begin{aligned} H(\omega) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\xi)g(t-\xi)e^{-i\omega t}\ dt\ d\xi = \\ &= \int_{-\infty}^{\infty} f(\xi)e^{-i\omega\xi}\left(\int_{-\infty}^{\infty} g(t-\xi)e^{-i\omega(t-\xi)}\ dt\right)d\xi = \\ &= \int_{-\infty}^{\infty} f(\xi)e^{-i\omega\xi}\cdot G(\omega)\ d\xi = F(\omega)\cdot G(\omega) \end{aligned}$$

□

Satz 2.13 (Ableitungssätze): Es gilt, sofern die Ableitungen existieren:

a) $FT(f'(t)) = i\omega F(\omega)$

b) $F'(\omega) = -iFT(tf(t))$

Beweis:

a) $FT(f'(t)) = \int_{-\infty}^{\infty} f'(t)e^{-i\omega t}\ dt = \left[f(t)e^{-i\omega t}\right]_{-\infty}^{\infty} - (-i\omega)\int_{-\infty}^{\infty} f(t)e^{-i\omega t}\ dt = i\omega F(\omega)$

b) $F'(\omega) = \int_{-\infty}^{\infty} f(t) \frac{d}{d\omega}(e^{-i\omega t})\ dt = -i \int_{-\infty}^{\infty} f(t) t e^{-i\omega t}\ dt = -iFT(tf(t))$ □

2.2.3 Diskrete Fourier-Transformation

Annahme: man kennt die Funktion $f(t)$ gar nicht als kontinuierliche Funktion, sondern nur an N diskreten Zeitpunkten $f(t_j) = f_j, t_j = j\Delta t, j = 0, \ldots, N-1$. Außerhalb des Intervalls $[0, T], T = N\Delta T$ wird die Funktion f als periodisch fortgesetzt angesehen. Gesucht ist die FT dieser diskreten Funktion. Äquivalent dazu ist die Fragestellung zu den N Stützstellen (t_j, f_j) ein trigonometrisches Interpolationspolynom zu finden.

Satz 2.14 (Diskrete Fourier-Transformation, DFT): Die diskrete FT $DFT(f_j)$ der N Samplingpunkte f_j ergibt sich zu

$$F_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \omega_N^{-kj} \text{ für } k = 0 \ldots N-1 \tag{112}$$

mit $\omega_N = e^{2\pi i/N}$. Die entsprechende inverse Transformation $IDFT(F_k)$ ergibt sich zu

$$f_j = \sum_{k=0}^{N-1} F_k \omega_N^{kj} \text{ für } j = 0 \ldots N-1 \tag{113}$$

In reeller Schreibweise: falls $N = 2M+1$ ungerade

$$f_j = \frac{A_0}{2} + \sum_{k=1}^{M} A_k \cos(\omega_k t) + B_k \sin(\omega_k t) \tag{114}$$

und falls $N = 2M$ gerade

$$f_j = \frac{A_0}{2} + \sum_{k=1}^{M-1} (A_k \cos(\omega_k t) + B_k \sin(\omega_k t)) + \frac{A_M}{2} \cos(\omega_M t) \tag{115}$$

wobei jeweils

$$A_k = \frac{2}{N} \sum_{j=0}^{N-1} f_j \cos(\omega_k t_j) \text{ und} \tag{116}$$

$$B_k = \frac{2}{N} \sum_{j=0}^{N-1} f_j \sin(\omega_k t_j) \tag{117}$$

mit $\omega_k = 2\pi k/T$.

Beweis: Es gilt mit der Definition der Fourier-Reihe:

$$\begin{aligned} C_k &= \frac{1}{T} \int_{\infty}^{\infty} f(t) e^{-i\omega_k t}\ dt = \frac{1}{N} \sum_{j=0}^{N-1} f(t_j) e^{-i2\pi k t_j/T} = \\ &= \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-i2\pi kj\Delta t/N\Delta t} = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-i2\pi kj/N} = F_k \end{aligned}$$

und alle Glieder der Fourier-Reihe C_k mit $k \geq N$ ergeben sich zu Null. Weiterhin gilt für ungerades N

$$A_0 = 2F_0 \text{ sowie } A_k = F_k + F_{N-k}, B_k = i(F_k - F_{N-k}) \text{ für } j = 1 \ldots M$$

und für gerades N

$$A_0 = 2F_0, A_M = 2F_M \text{ sowie } A_k = F_k + F_{N-k}, B_k = i(F_k - F_{N-k}) \text{ für } j = 1 \dots M-1$$

was man über

$$\cos kt_j = \frac{e^{kit_j} + e^{(N-k)it_j}}{2} \text{ und } \sin kt_j = \frac{e^{kit_j} - e^{(N-k)it_j}}{2i}$$

sieht □

Anmerkung: es gilt $F_{N-k} = \overline{F_k}$ für $k = 1, \dots N-1$, d.h. F hat hermitesche Symmetrie. Das bedeutet, dass nur etwa die Hälfte der Koeffizienten in F unabhängig sind und gespeichert werden müssen.

Beispiele ($N = 4$): Zu berechnen sind:

$$\begin{aligned}
F_0 &= \frac{1}{4}(f_0\omega_4^{-0\cdot0} + f_1\omega_4^{-0\cdot1} + f_2\omega_4^{-0\cdot2} + f_3\omega_4^{-0\cdot3}) \\
F_1 &= \frac{1}{4}(f_0\omega_4^{-1\cdot0} + f_1\omega_4^{-1\cdot1} + f_2\omega_4^{-1\cdot2} + f_3\omega_4^{-1\cdot3}) \\
F_2 &= \frac{1}{4}(f_0\omega_4^{-2\cdot0} + f_1\omega_4^{-2\cdot1} + f_2\omega_4^{-2\cdot2} + f_3\omega_4^{-2\cdot3}) \\
F_3 &= \frac{1}{4}(f_0\omega_4^{-3\cdot0} + f_1\omega_4^{-3\cdot1} + f_2\omega_4^{-3\cdot2} + f_3\omega_4^{-3\cdot3})
\end{aligned}$$

Benötigt werden die Faktoren ω_4^{-ik} für $i, k = 0, 1, 2, 3$:

Bild "Eponential-Uhr" auf dem Einheitskreis: $\omega_4^0, \omega_4^1, \omega_4^2, \omega_4^3$ mit Real- und Imaginärteil

	$\omega_4^{-0\cdot0}$	$\omega_4^{-0\cdot1}$	$\omega_4^{-0\cdot2}$	$\omega_4^{-0\cdot3}$	$\omega_4^{-1\cdot0}$	$\omega_4^{-1\cdot1}$	$\omega_4^{-1\cdot2}$	$\omega_4^{-1\cdot3}$	$\omega_4^{-2\cdot0}$	$\omega_4^{-2\cdot1}$	$\omega_4^{-2\cdot2}$	$\omega_4^{-2\cdot3}$	$\omega_4^{-3\cdot0}$	$\omega_4^{-3\cdot1}$	$\omega_4^{-3\cdot2}$	$\omega_4^{-3\cdot3}$
Re	1	1	1	1	1	0	-1	0	1	-1	1	-1	1	0	-1	0
Im	0	0	0	0	0	1	0	-1	0	0	0	0	0	-1	0	1

a) "Konstante": $f = (1,1,1,1) \Rightarrow F = \frac{1}{4}((1+1+1+1), (1+0-1+0), (1-1+1-1), (1+0-1+0)) = (1,0,0,0)$

b) "Kosinus": $f = (1,0,-1,0) \Rightarrow F = \frac{1}{4}((1-1), (1+1), (1-1), (1+1)) = (0, \frac{1}{2}, 0, \frac{1}{2})$

c) "Sinus": $f = (0,1,0,-1) \Rightarrow F = \frac{1}{4}((1-1), (i+i), (-1+1), (-i-i)) = (0, \frac{i}{2}, 0, -\frac{i}{2})$

In reller Darstellung $N = 4$, $M = 2$:

a) "Konstante": $A = (2,0,0), B = (-,0,-)$

b) "Kosinus": $A = (0,1,0), B = (-,0,-)$

c) "Sinus": $A = (0,0,0), B = (-,1,-)$

Satz 2.15 (Inverse diskrete Fourier-Transformation): Es gilt:

$$IDFT(DFT(f_j)) = f_j \text{ für } j = 0 \dots N-1 \tag{118}$$

Beweis:

$$\begin{aligned} f_j &= \sum_{k=0}^{N-1} F_k \omega_N^{kj} = \sum_{k=0}^{N-1} \frac{1}{N} \sum_{l=0}^{N-1} f_l \omega_N^{-jl} \omega_N^{kj} = \\ &= \frac{1}{N} \sum_{l=0}^{N-1} f_l \sum_{k=0}^{N-1} \omega_N^{(k-l)j} = \frac{1}{N} \sum_{l=0}^{N-1} f_l N \delta_{j,l} = f_j \end{aligned}$$

mit $\delta_{j,l} = \begin{cases} 1 & \text{für } j = l \\ 0 & \text{sonst} \end{cases}$ □

2.2.4 Trigonometrische Interpolation

Satz 2.16 (Existenz und Eindeutigkeit der trigonometrischen Interpolation): Zu den N Stützpunkten $(t_j, f_j), j = 0, \ldots N-1$ mit $t_j = 2\pi j/N$ gibt es genau ein *trigonometrisches Polynom* $P(t)$ vom Grad N der Form

$$P(t) = \sum_{k=0}^{N-1} c_k e^{ikt} \tag{119}$$

mit $c_k \in \mathbb{C}$ sodass

$$P(t_j) = f_j \text{ für } 0 \leq j \leq N-1 \tag{120}$$

Beweis:

1. Existenz: Sei $\omega = e^{it}$, dann hat das trigonometrische Polynom die Form

$$P(t) = \sum_{k=0}^{N-1} c_k \omega^k$$

und aus der Existenz der Polynominterpolation (Satz 2.1) und der Bijektivität der Abbildung $t \to \omega$ folgt direkt die Existenz des trigonometrischen Interpolationsproblems.

2. Eindeutigkeit: setze $\omega_j = e^{it_j} = e^{2j\pi i/N}$ welche die Eigenschaften

$$\omega_j^k = \omega_k^j \text{ und } \omega_k^{-j} = \overline{\omega_k^j}$$

sowie die Orthogonalitätseigenschaft

$$\sum_{k=0}^{N-1} \omega_k^j \omega_k^{-l} = \begin{cases} N & \text{für } j = l \\ 0 & \text{für } j \neq l \end{cases}$$

(Beweis siehe Übungsblatt) besitzen. ω_{j-k} ist eine Nullstelle des Polynoms $\omega^N - 1 = (\omega - 1)(\omega^{N-1} + \omega^{N-2} + \ldots + 1)$ sodass entweder $\omega_{j-l} = 1$, also $j = l$ gilt oder

$$\sum_{k=0}^{N-1} \omega_k^j \omega_k^{-l} = \sum_{k=0}^{N-1} \omega_k^{j-l} = \sum_{k=0}^{N-1} \omega_{j-l}^k = 0$$

□

Bild Stützstellen (reell) für $N = 4$ und Basispolynome $1, cos(x), cos(2x)$ und $sin(x)$

Mit dem üblichen *Skalarprodukt für Vektoren* $f = (f_0, f_1, \dots f_{N-1}) \in \mathbb{C}^N$

$$(f, g) = \frac{1}{N} \sum_{j=0}^{N-1} f_j \overline{g_j} \tag{121}$$

bilden die Vektoren

$$\omega^{(j)} = (\omega_0^j, \omega_1^j, \dots \omega_{N-1}^j) \text{ für } j = 0, 1 \dots N-1 \tag{122}$$

eine Orthogonalbasis des $\mathbb{C}^N$, d.h.

$$(\omega^{(j)}, \omega^{(l)}) = \begin{cases} 1 & \text{für } j = l \\ 0 & \text{für } j \neq l \end{cases} . \tag{123}$$

Satz 2.17 (Explizite Darstellung des trigonometrischen Interpolationspolynoms): Für das trigonometrische Polynom $P(t) = \sum_{k=0}^{N-1} c_k e^{ikt}$ vom Grad N gilt $P(t_j) = f_j$ für $0 \leq j < N$ für komplexes f_j und $t_j = 2\pi j/N$ genau dann wenn

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \omega_j^{-k} = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi i j k/N} \tag{124}$$

für $k = 0, \dots, N-1$.

Beweis: aus $P(t_j) = f_j$ gilt für den Vektor

$$f = (f_0, \dots f_{N-1}) = \sum_{k=0}^{N-1} c_k \omega^{(k)}$$

sodass

$$\frac{1}{N} \sum_{j=0}^{N-1} f_j \omega_j^{-k} = (f, \omega^{(k)}) = (c_0 \omega^{(0)} + \dots + c_{N-1} \omega^{(N-1)}, \omega^{(k)}) = c_k$$

□

Die *Abschnittspolynome* $P_s(t)$ von $P(t)$

$$P_s(t) = \sum_{k=0}^{s} c_k e^{sit} = \sum_{k=0}^{s} c_k \omega^s \text{ für } s \leq N-1 \tag{125}$$

besitzen folgende Minimaleigenschaft:

Satz 2.18 (Bestapproximation der trigonometrischen Interpolation): Für jedes $s = 0, \dots, N-1$ minimiert unter allen trigonometrischen Polynomen $Q(t)$ der Form

$$Q(t) = \sum_{k=0}^{s} d_k e^{sit} \tag{126}$$

gerade das s-te Abschnittpolynom $P_s(t)$ von $P(t)$ das *Fehlerfunktional*

$$S(Q) = \frac{1}{N} \sum_{k=0}^{N-1} |f_k - Q(t_k)|^2 \tag{127}$$

wobei insbesondere $S(P) = 0$ ist.

Beweis: mit den Vektoren

$$P_s = (P_s(x_0), \ldots, P_s(x_{N-1})) \text{ und } Q_s = (Q_s(x_0), \ldots, Q_s(x_{N-1}))$$

gilt

$$S(Q) = (f - Q, f - Q)$$

Nun ist wegen Satz 2.17

$$(f, w^{(k)}) = c_k \text{ für } k = 0, \ldots, N-1$$

und daher

$$(f - P_S, w^{(k)}) = (f - \sum_{l=0}^{s} c_l \omega^{(s)}, \omega^{(k)}) = c_k - c_k = 0$$

sowie

$$(f - P_S, P_S - Q) = \sum_{l=0}^{s} (f - P_s, (c_l - d_l)\omega^{(l)}) = 0$$

Damit gilt aber

$$\begin{aligned} S(Q) &= (f - Q, f - Q) \\ &= ((f - P_s) + (P_s - Q), (f - P_s) + (P_s - Q)) \\ &= (f - P_s, f - P_s) + (P_s - Q, P_s - Q) \\ &\geq (f - P_s, f - P_s) \\ &= S(P_s) \end{aligned}$$

wobei Gleichheit nur für $(P_s - Q, P_s - Q) = 0$ also $Q = P_s$ gilt. Aufgrund der Eindeutigkeit (Satz 2.16) gilt damit $P_s(t) = Q(t)$. □

Insgesamt ergeben sich die folgenden Algorithmen zur diskreten Fouriertransformation:

Algorithmus 2.19 (Reelle DFT, Analyse):

```
for (i=0; i<=N/2; i++)
  A[i]=0; B[i]=0;
  for (j=0; j<N; j++)
    t=2*PI*j/N;
    A[i]=A[i]+F[j]*cos(t*i)*2/N;
    B[i]=B[i]+F[j]*sin(t*i)*2/N;
```

Algorithmus 2.20 (Relle DFT, Synthese):

```
for (i=0; i<N; i++)
  t=2*PI*i/N;
  F[i]=A[0]/2;
  for (j=1; j<N/2; j++)
    F[i]=F[i]+A[j]*cos(t*j)+B[j]*sin(t*j);
  F[i]=F[i]+A[N/2]/2*cos(t*N/2);
```

Der Aufwand beider Algorithmen ist quadratisch in der Zahl der Stützstellen N.

2.2.5 Schnelle Fourier-Transformation

Die Schnelle Fourier-Transformation (FFT) ist einer der wichtigsten Algorithmen überhaupt. Grundlage ist die Zerlegung ($N = 2M$)

$$f_k = \sum_{j=0}^{N-1} c_j \omega_k^j = \sum_{j=0}^{M-1} c_{2j} \omega_k^{2j} + \sum_{j=0}^{M-1} c_{2j+1} \omega_k^{2j+1} = \tag{128}$$

$$= \sum_{j=0}^{M-1} c_{2j} \omega_{2k}^j + \omega_k \sum_{j=0}^{M-1} c_{2j+1} \omega_{2k}^j = f_{2k}^g + \omega_k f_{2k}^u \tag{129}$$

in einen geraden (g) und einen ungeraden Teil (u). Zur Berechnung sind somit zwei Fourier-Transformationen der halben Länge $M = N/2$ für die geraden und die ungeraden Indizes notwendig.

Weiterhin gilt die Symetrieeigenschaft

$$f_{k+M} = f_{2k}^g + \omega_{k+M} f_{2k}^u = f_{2k}^g - \omega_k f_{2k}^u \tag{130}$$

Die Idee der FFT ist nun die rekursive Anwendung dieser Zerlegung für $N = 2^m$, also

$$f_k = f_{2k}^g + \omega_k f_{2k}^u = (f_{4k}^{gg} + \omega_{2k} f_{4k}^{gu}) + \omega_k (f_{4k}^{ug} + \omega_{2k} f_{4k}^{uu}) = \ldots \tag{131}$$

Mit Hilfe einer Umsortierung der Indizes kann die FFT am Platz ohne aufwändiges Kopieren durchgeführt werden:

$$(0,1,2,3) \to (0,2,1,3),\ (0,1,2,3,4,5,6,7) \to (0,4,2,6,1,5,3,7), \ldots \tag{132}$$

Diese Umsortierung entspricht einem Bit-Reversal in der Binärdarstellung der Indizes.

0	000	$\to$	000	0
1	001	$\to$	100	4
2	010	$\to$	010	2
3	011	$\to$	110	6
4	100	$\to$	001	1
5	101	$\to$	101	5
6	110	$\to$	011	3
7	111	$\to$	111	7

Abbildung 8: Bitreversal für $N = 8$.

Der Gesamtaufwand der FFT ist damit

$$O(N) \cdot 1 + \frac{O(N)}{2} \cdot 2 + \frac{O(N)}{4} \cdot 4 + \ldots + 2 \cdot \frac{N}{2} + 1 \cdot N = O(N \log N) \tag{133}$$

was asymptotisch besser als der Aufwand $O(N^2)$ der DFT ist.

Beispiel ($N = 8$): Die Diskrete Fourier-Transformation

$$\begin{aligned} c_0 &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 \\ c_1 &= f_0 + \frac{i+1}{\sqrt{2}} f_1 + i f_2 + \frac{i-1}{\sqrt{2}} f_3 - f_4 - \frac{i+1}{\sqrt{2}} f_5 - i f_6 - \frac{i-1}{\sqrt{2}} f_7 \\ c_2 &= f_0 + i f_1 - f_2 - i f_3 + f_4 + i f_5 - f_6 - i f_7 \end{aligned}$$

$$
\begin{aligned}
c_3 &= f_0 + \frac{i-1}{\sqrt{2}} f_1 - i f_2 + \frac{i+1}{\sqrt{2}} f_3 - f_4 - \frac{i-1}{\sqrt{2}} f_5 + i f_6 - \frac{i+1}{\sqrt{2}} f_7 \\
c_4 &= f_0 - f_1 + f_2 - f_3 + f_4 - f_5 + f_6 - f_7 \\
c_5 &= f_0 - \frac{i+1}{\sqrt{2}} f_1 + i f_2 - \frac{i-1}{\sqrt{2}} f_3 - f_4 + \frac{i+1}{\sqrt{2}} f_5 - i f_6 + \frac{i-1}{\sqrt{2}} f_7 \\
c_6 &= f_0 - i f_1 - f_2 + i f_3 + f_4 - i f_5 - f_6 + i f_7 \\
c_7 &= f_0 - \frac{i-1}{\sqrt{2}} f_1 - i f_2 - \frac{i+1}{\sqrt{2}} f_3 - f_4 + \frac{i-1}{\sqrt{2}} f_5 + i f_6 + \frac{i+1}{\sqrt{2}} f_7
\end{aligned}
$$

benötigt 64 Multiplikationen und 56 Additionen.

Für die schnelle Fourier-Transformation betrachte die Zerlegung in Teilsummen

$$
\begin{aligned}
d_0 &= \frac{1}{2}(c_0 + c_4) = f_0 + f_2 + f_4 + f_6 \\
d_1 &= \frac{1}{2}(c_0 - c_4) = f_1 + f_3 + f_5 + f_7 \\
d_2 &= \frac{1}{2}(c_1 + c_5) = f_0 + i f_2 - f_4 - i f_6 \\
d_3 &= \frac{1}{2}(c_1 - c_5) = \frac{i+1}{\sqrt{2}}(f_1 + i f_3 - f_5 - i f_7) \\
d_4 &= \frac{1}{2}(c_2 + c_6) = f_0 - f_2 + f_4 - f_6 \\
d_5 &= \frac{1}{2}(c_2 - c_6) = i(f_1 - f_3 + f_5 - f_7) \\
d_6 &= \frac{1}{2}(c_3 + c_7) = f_0 - i f_2 - f_4 + i f_6 \\
d_7 &= \frac{1}{2}(c_3 - c_7) = \frac{i-1}{\sqrt{2}}(f_1 - i f_3 - f_5 + i f_7)
\end{aligned}
$$

und diese weiter in:

$$
\begin{aligned}
e_0 &= \frac{1}{2}(d_0 + d_4) = f_0 + f_4 \\
e_1 &= \frac{1}{2}(d_0 - d_4) = f_2 + f_6 \\
e_2 &= \frac{1}{2}(i d_1 + d_5) = i(f_1 + f_5) \\
e_3 &= \frac{1}{2}(i d_1 - d_5) = i(f_3 + f_7) \\
e_4 &= \frac{1}{2}(d_2 + d_6) = f_0 - f_4 \\
e_5 &= \frac{1}{2}(d_2 - d_6) = i(f_2 - f_6) \\
e_6 &= \frac{1}{2}(i d_3 + d_7) = \frac{i-1}{\sqrt{2}}(f_1 - f_5) \\
e_7 &= \frac{1}{2}(i d_3 - d_7) = -\frac{i+1}{\sqrt{2}}(f_3 - f_7)
\end{aligned}
$$

und diese weiter in

$$
g_0 = \frac{1}{2}(e_0 + e_4) = f_0
$$

$$
\begin{aligned}
g_1 &= \frac{1}{2}(e_0 - e_4) = f_4 \\
g_2 &= \frac{1}{2}(ie_1 + e_5) = if_2 \\
g_3 &= \frac{1}{2}(ie_1 - e_5) = if_6 \\
g_4 &= \frac{1}{2}(\frac{i+1}{\sqrt{2}}e_2 + e_6) = \frac{i-1}{\sqrt{2}}f_1 \\
g_5 &= \frac{1}{2}(\frac{i+1}{\sqrt{2}}e_2 - e_6) = \frac{i-1}{\sqrt{2}}f_5 \\
g_6 &= \frac{1}{2}(\frac{i-1}{\sqrt{2}}e_3 + e_7) = -\frac{i+1}{\sqrt{2}}f_3 \\
g_7 &= \frac{1}{2}(\frac{i-1}{\sqrt{2}}e_3 - e_7) = -\frac{i+1}{\sqrt{2}}f_7
\end{aligned}
$$

Die FFT besteht dann aus den Berechnungen

$$
\begin{array}{llll}
g_0 = f_0 & g_1 = f_4 & g_2 = if_2 & g_3 = if_6 \\
g_4 = \frac{i-1}{\sqrt{2}}f_1 & g_5 = \frac{i-1}{\sqrt{2}}f_5 & g_6 = -\frac{i+1}{\sqrt{2}}f_3 & g_3 = -\frac{i+1}{\sqrt{2}}f_7 \\
e_0 = g_0 + g_1 & e_1 = -i(g_2 + g_3) & e_2 = -\frac{i-1}{\sqrt{2}}(g_4 + g_5) & e_3 = -\frac{i-1}{\sqrt{2}}(g_6 + g_7) \\
e_4 = g_0 - g_1 & e_5 = g_2 - g_3 & e_6 = g_4 - g_5 & e_7 = (g_6 - g_7) \\
d_0 = e_0 + e_1 & d_1 = -i(e_2 + e_3) & d_2 = -(e_4 + e_5) & d_3 = -i(e_6 + e_7) \\
d_4 = e_0 - e_1 & d_5 = e_2 - e_3 & d_6 = e_4 - e_5 & d_7 = e_6 - e_7 \\
c_0 = d_0 + d_1 & c_1 = d_2 + d_3 & c_2 = d_4 + d_5 & c_3 = d_6 + d_7 \\
c_4 = d_0 - d_1 & c_5 = d_2 - d_3 & c_6 = d_4 - d_5 & c_7 = d_6 - d_7
\end{array}
$$

Sie benötigt nur 24 Multiplikationen und 24 Additionen.

Algorithmus 2.21 (Reelle FFT, Analyse):

```
for (i=0; i<N, i++)
  B[i]=0; A[i]=F[i]*2/N;
for (i=0; i<N; i++)
  k=0;
  for (j=1; j<=N/2; j*=2)
    if (i&j!=0) k=k+N/(2*j);
  if (k>i) swap(A[k],A[i]);
for (i=0; i<N; i+=2)
  swap(A[i],A[i+1]);
for (i=1, m=n; i<N; i*=2; m/=2)
  for (j=0; j<i; j++)
    c=cos(PI*j/i);
    s=sin(PI*j/i);
    for (k=j; k<N; k+=2*i)
      ak=A[k]; aki=A[k+i]; bk=B[k]; bki=B[k+i];
```

```
      B[k]=bk+c*bki+s*aki;
      B[k+i]=bk-c*bki-s*aki;
      A[k]=ak+c*aki-s*bki;
      A[k+i]=ak-c*aki+s*bki;
```

Algorithmus 2.22 (Reelle FFT, Synthese):

```
for (i=0; i<N, i++)
  G[i]=B[i]; F[i]=A[i];
for (i=N/2, m=2; i>0; i/=2; m*=2)
  for (j=0; j<i; j++)
    c=cos(PI*j/i);
    s=sin(PI*j/i);
    d=s*s+c*c;
    for (k=j; k<N; k+=2*i)
      fk=F[k]; fki=F[k+i]; gk=G[k]; gki=G[k+i];
      F[k]=(fk+fki)/2;
      F[k+i]=(s*(gk-gki)/2+c*(fk-fki)/2)/d;
      G[k]=(gk+gki)/2;
      G[k+i]=(c*(gk-gki)/2+s*(fki-fk)/2)/d;
for (i=0; i<N; i+=2)
  swap(F[i],F[i+1]);
for (i=0; i<N; i++)
  F[i]=F[i]/(2*N);
for (i=0; i<N; i++)
  k=0;
  for (j=1; j<=N/2; j*=2)
    if (i&j!=0) k=k+N/(2*j);
  if (k>i) swap(F[k],F[i]);
```

2.3 Splines

Ziel der Spline-Interpolation ist es gegebene Punkte durch eine möglichst glatte Kurve zu verbinden.

Zur Herkunft des Wortes: engl. spline=dt. Stracklatte, Latte aus Stahl oder Holz zur Formung der Aussenwand von Schiffen und Häusern; durch Fixierung an bestimmten Stellen werden geschwungene Kurven erzeugt.

Anwendungen:

- CAD (computer aided design), CAM (computer aided manufacturing)
- Computergraphik, Visualisierung
- Finite-Elemente Methode zur Lösung partieller Differentialgleichungen

2.3.1 Interpolation mit Splines

Sei $\Delta = \{a = x_0 < x_1 < \ldots < x_n = b\}$ eine Unterteilung von $[a, b]$. Der zu Δ gehörige *kubische Spline* S_Δ ist eine reelle Funktion $S_\Delta : [a, b] \to \mathbb{R}$ mit

1. $S_\Delta \in C^2[a,b]$, d.h. S_Δ ist auf $[a,b]$ zweimal stetig differenzierbar und
2. auf jedem Teilintervall $[x_i, x_{i+1}], 0 \leq i \leq n-1$ stimmt S_Δ mit einem kubischen Polynom überein.

Die Splinefunktion ist aus n kubischen Polynomstücken so zusammengesetzt, dass sowohl die Funktion als auch ihre ersten beiden Ableitungen an den Knoten $x_i, 1 \leq i \leq n-1$ keine Sprungstellen hat.

Bild Beispiel einer kubischen Splinefunktion

Verallgemeinert ist eine *Splinefunktion k-ten Grades* eine $(k-1)$-mal stetig differenzierbare Funktion, die stückweise aus Polynomen k-ten Grades besteht. Im folgenden beschänken wir uns aber auf kubische Splines.

Sei $f := \{f_0, f_1, \ldots, f_n\}$ eine Menge von $n+1$ reellen Funktionswerten

Bild $n+1$ Stützstellen und Funktionswerte, n Intervalle, $n-1$ innere Punkte

dann bezeichnet $S_\Delta(f,x)$ eine interpolierende Splinefunktion S_Δ zu f mit

$$S_\Delta(f, x_i) = f_i \tag{134}$$

Damit ist jedoch noch keine Eindeutigkeit gegeben:

n kubische Polynomstücke	$4n$ Freiheitsgrade
jeweils linker+rechter Funktionswert gegeben	$-2n$ Freiheitsgrade
Gleichheit der 1. Ableitung an den inneren Punkten	$-(n-1)$ Freiheitsgrade
Gleichheit der 2. Ableitung an den inneren Punkten	$-(n-1)$ Freiheitsgrade
	2 Freiheitsgrade offen

Typische Wahlen der zwei Zusatzbedingungen sind:

a) natürlicher Spline: $S''_\Delta(f,a) = S''_\Delta(f,b) = 0$

b) periodischer Spline: $S^{(k)}_\Delta(f,a) = S^{(k)}_\Delta(f,b) = 0$ für $k = 0,1,2$ und periodische Funktionswerte $f_0 = f_n$

c) vollständiger Spline: $S'_\Delta(f,a) = f'(a)$ und $S'_\Delta(f,b) = f'(b)$ wobei $f'(a)$ und $f'(b)$ vorgegebene Werte sind

Eine Funktion $f : [a,b] \to \mathbb{R}$ heisst *absolutstetig* auf $[a,b]$ falls es zu jedem $\varepsilon > 0$ ein $\delta > 0$ gibt sodass $\sum_i |f(b_i) - f(a_i)| \leq \delta$ für alle a_i, b_i mit $a = a_1 < b_1 < a_2 < b_2 < \ldots < a_n < b_b = b$ mit $\sum_i |b_i - a_i| \leq \varepsilon$.

Der *Raum der Splinefunktionen vom Grad m* $K^m(a,b)$, ist die Menge aller reeller Funktionen f auf $[a,b]$ für die die $(m-1)$-te Ableitung $f^{(m-1)}$ auf $[a,b]$ noch absolutstetig ist und für die $f^{(m)}$ quadratisch integrierbar ist, d.h. $f^{(m)} \in L^2[a,b]$.

Weiterhin ist $K^m_p(a,b)$ der Raum der periodischen Splinefunktionen, d.h. die Menge der Funktionen aus $K^m(a,b)$ mit $f^{(k)}(a) = f^{(k)}(b)$ für $0 \leq k \leq m-1$.

Es gilt:

- $S_\Delta \in K^3(a,b)$
- $S_\Delta(f,\cdot) \in K_p^3(a,b)$ für periodische Splines (Fall b).

Für $f \in K^2(a,b)$ sei folgende Halbnorm definiert

$$\|f\|^2 = \int_a^b |f''(x)|^2 \, dx \tag{135}$$

Bemerkung: $\|\cdot\|$ ist nur eine Halbnorm, denn es gibt Funktionen $f(x) \neq 0$ mit $\|f\| = 0$, z.B. lineare Funktionen $f(x) = cx + d$.

Satz 2.23 (Identität von Holladay): Ist $f \in K^2(a,b)$, so gilt

$$\|f-S_\Delta\|^2 = \|f\|^2 - \|S_\Delta\|^2 - 2\left((f'(x) - S'_\Delta(x))(S''_\Delta(b) - S''_\Delta(a)) - \sum_{i=1}^{n}(f(x) - S_\Delta(x))(S'''_\Delta(x_{i-1}^+) - S'''_\Delta(x_i^-))\right) \tag{136}$$

wobei x_i^+ und x_i^- der links- bzw. der rechtsseitige Grenzwert von $S'''_\Delta(x)$ an der Stützstelle x_i ist.

Beweis:

$$\begin{aligned}
\|f - S_\Delta\|^2 &= \int_a^b |f''(x) - S''_\Delta(x)|^2 \, dx = \\
&= \|f\|^2 - 2\int_a^b f''(x)S''_\Delta(x) \, dx + \|S_\Delta\|^2 = \\
&= \|f\|^2 - 2\int_a^b (f''(x) - S''_\Delta(x))S''_\Delta(x) \, dx - \|S_\Delta\|^2
\end{aligned}$$

Partielle Integration ergibt für $1 \leq i \leq n$:

$$\begin{aligned}
\int_{x_{i-1}}^{x_i} (f''(x) - S''_\Delta(x))S''_\Delta(x) \, dx &= (f'(x) - S'_\Delta(x))S''_\Delta(x)|_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} (f'(x) - S'_\Delta(x))S'''_\Delta(x) \, dx = \\
&= (f'(x) - S'_\Delta(x))S''_\Delta(x)|_{x_{i-1}}^{x_i} - (f(x) - S_\Delta(x))S'''_\Delta(x)|_{x_{i-1}}^{x_i} + \int_{x_{i-1}}^{x_i} (f(x) - S_\Delta(x))S_\Delta^{(4)}(x) \, dx
\end{aligned}$$

Es ist $S_\Delta^{(4)} = 0$ auf den Teilintervallen (x_{i-1}, x_i) und f', S'_Δ und S''_Δ sind stetig auf $[a,b]$. Mit Summation über $i = 1 \ldots n$ und der Beziehung

$$\sum_{i=1}^{n} (f'(x) - S'_\Delta(x))S''_\Delta(x)|_{x_{i-1}}^{x_i} = (f'(x) - S'_\Delta(x))S''_\Delta(x)|_a^b$$

gilt die Behauptung. □

Satz 2.24 (Minimum-Norm-Eigenschaft von Splines): Für vorgegebene Werte $f = (f_0, \ldots, f_n)$ und für $f \in K^2(a,b)$ mit $f(x_i) = f_i$ für $0 \leq i \leq n$ gilt

$$\|S_\Delta(f,\cdot)\| \leq \|f\| \, . \tag{137}$$

und weiterhin

$$\|f - S_\Delta(f,\cdot)\|^2 = \|f\|^2 - \|S_\Delta(f,\cdot)\|^2 \geq 0 \tag{138}$$

für jede kubische Splinefunktion $S_\Delta(f,\cdot)$, die zusätzlich eine der drei Bedingungen a), b) oder c) erfüllt.

Beweis: In jedem der drei Fälle a), b), c) verschwindet in Satz 2.23 der Ausdruck

$$(f'(x) - S'_\Delta(x))(S''_\Delta(b) - S''_\Delta(a)) - \sum_{i=1}^{n}(f(x) - S_\Delta(x))(S'''_\Delta(x_{i-1}^+) - S'''_\Delta(x_i^-))$$

falls $S_\Delta \equiv S_\Delta(f, \cdot)$. □

Satz 2.25 (Eindeutigkeit der Spline-Interpolation): Für vorgegebene Werte $f = (f_0, \ldots, f_n)$ ist in jedem der Fälle a) bis c) die Splinefunktion $S_\Delta(f, \cdot)$ eindeutig bestimmt.

Beweis: Gäbe es eine weitere Splinefunktion $\bar{S}_\Delta(f, \cdot)$ mit den angegebenen Eigenschaften, so wäre $f(x) = \bar{S}_\Delta(f, \cdot)$ eine Funktkion f mit $f \in K^2(a, b)$ und $f(x_i) = f_i$ für $0 \leq i \leq n$. Daher

$$\|\bar{S}_\Delta(f, \cdot) - S_\Delta(f, \cdot)\|^2 = \|\bar{S}_\Delta(f, \cdot)\|^2 - \|S_\Delta(f, \cdot)\|^2 \geq 0$$

und somit, da $S_\Delta(f, \cdot)$ und $\bar{S}_\Delta(f, \cdot)$ vertauscht werden können

$$\|\bar{S}_\Delta(f, \cdot) - S_\Delta(f, \cdot)\|^2 = \int_a^b |\bar{S}''_\Delta(f, x) - S''_\Delta(f, x)|^2 \, dx = 0$$

Da $S''_\Delta(f, x)$ und $\bar{S}''_\Delta(f, x)$ stetig sind ist $S''_\Delta(f, x) \equiv \bar{S}''_\Delta(f, x)$ und daher durch Integration

$$\bar{S}_\Delta(f, x) \equiv S_\Delta(f, x) + cx + dS''_\Delta(f, x)|^2 \, dx = 0$$

und wegen $S_\Delta(f, x) = \bar{S}_\Delta(f, x)$ für $x = a, b$ gilt $c = d = 0$ und somit die Eindeutigkeit $S_\Delta(f, x) = \bar{S}_\Delta(f, x)$. □

Der Satz 2.24 beschreibt eine Minimaleigenschaft der Splinefunktion, z.B. für a) minimiert unter allen Funktionen $f \in K^2(a, b)$ mit $f(x_i) = f_i$ gerade die Splinefunktion $S_\Delta(f, \cdot)$ mit $S''_\Delta(f, x) = 0$ für $x = a, b$ das Integral

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

Da $|f''(x)|$ die Krümmung von f an der Stelle x approximiert, kann man $\|f\|$ als Maß der Gesamtkrümmung ansehen. Unter allen auf $[a, b]$ zweimal stetig differenzierbaren Funktionen f mit $f(x_i) = f_i$ ist so die natürliche Splinefunktion $S_\Delta(f, \cdot)$ die „glatteste“ Funktion im dem Sinne, dass sie die kleinste Gesamtkrümmung besitzt.

2.3.2 Berechnung kubischer Splines

Nun wenden wir uns der Berechnung von $S_\Delta(f, \cdot)$ für eine gegebene Unterteilung Δ von $[a, b]$ in n Teilintervalle und gegebene $n + 1$ Funktionswerte $f = (f_0, \ldots, f_n)$ zu.

Die *Momente* M_j mit $j = 0, \ldots, n$ sind definiert als die zweiten Ableitungen der gesuchten Splinefunktion $S_\Delta(f, \cdot)$ an den Knoten $x_j \in \Delta$, also

$$M_j = S''_\Delta(f, x_j) \tag{139}$$

Das weitere Vorgehen ist nun wie folgt:

1. die Splinefunktion $S_\Delta(f, \cdot)$ wird allein mittels der Momente M_j angegeben,
2. die Momente M_j werden dann als Lösung eines (noch aufzustellenden) linearen Gleichungssystems bestimmt.

Zu 1.: $S''_\Delta(f,\cdot)$ ist in jedem Teilintervall $[x_j, x_{j+1}], j = 0,\ldots,n-1$ eine lineare Funktion, die mittels M_j beschreibbar ist:

$$S''_\Delta(f,x) = M_j \frac{x_{j+1}-x}{h_{j+1}} + M_{j+1}\frac{x-x_j}{h_{j+1}} \text{ für } x \in [x_j, x_{j+1}] \tag{140}$$

wobei $h_{j+1} = x_{j+1} - x_j$ für $j = 0,\ldots,n-1$.

Durch Integration über $x \in [x_j, x_{j+1}]$ erhält man

$$S'_\Delta(f,x) = -M_j\frac{(x_{j+1}-x)^2}{2h_{j+1}} + M_{j+1}\frac{(x-x_j)^2}{2h_{j+1}} + a_j \tag{141}$$

$$S_\Delta(f,x) = M_j\frac{(x_{j+1}-x)^3}{6h_{j+1}} + M_{j+1}\frac{(x-x_j)^3}{6h_{j+1}} + a_j(x-x_j) + b_j \tag{142}$$

wobei a_j und b_j die Integrationskonstanten sind. Mit den Interpolationsbedingungen

$$S_\Delta(f,x_j) = f_j \text{ und } S_\Delta(f,x_{j+1}) = f_{j+1}$$

ergeben sich für a_j und b_j die Gleichungen

$$M_j\frac{h_{j+1}^2}{6} + b_j = f_j \tag{143}$$

$$M_{j+1}\frac{h_{j+1}^2}{6} + a_j h_{j+1} + b_j = f_{j+1} \tag{144}$$

und somit

$$b_j = f_j - M_j\frac{h_{j+1}^2}{6} \tag{145}$$

$$a_j = \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{h_{j+1}}{6}(M_{j+1}-M_j) \tag{146}$$

Damit gilt für $x \in [x_j, x_{j+1}]$

$$S_\Delta(f,x) = \alpha_j + \beta_j(x-x_j) + \gamma_j(x-x_j)^2 + \delta_j(x-x_j)^3 \text{ mit} \tag{147}$$

$$\alpha_j = f_j \tag{148}$$

$$\beta_j = S'_\Delta(f,x_j) = \frac{-M_j h_{j+1}}{2} + a_j = \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{2M_j + M_{j+1}}{6}h_{j+1} \tag{149}$$

$$\gamma_j = \frac{S''_\Delta(f,x_j)}{2} = \frac{M_j}{2} \tag{150}$$

$$\delta_j = \frac{S'''_\Delta(f,x_j^+)}{6} = \frac{M_{j+1}-M_j}{6h_{j+1}} \tag{151}$$

und so ist $S_\Delta(f,\cdot)$ mit Hilfe der Momente M_j ausgedrückt.

Zu 2.: zur Berechnung der Momente M_j nutzt man die Stetigkeit von $S'_\Delta(f,\cdot)$ an $x = x_j, 1 \le j \le n-1$ und erhält $n-1$ Bestimmungsgleichungen. Durch Einsetzen der Werte von a_j in die Gleichung für S'_Δ erhält man

$$S'_\Delta(f,x) = -M_j\frac{(x_{j+1}-x)^2}{2h_{j+1}} + M_{j+1}\frac{(x-x_j)^2}{2h_{j+1}} + \frac{f_{j+1}-f_j}{h_{j+1}} - \frac{h_{j+1}}{6}(M_{j+1}-M_j) \tag{152}$$

und damit für $j = 1,\ldots,n-1$

$$S'_\Delta(f,x_j^-) = \frac{f_j - f_{j-1}}{h_j} + \frac{h_j}{3}M_j + \frac{h_j}{6}M_{j-1} \tag{153}$$

$$S'_\Delta(f,x_j^+) = \frac{f_{j-1}-f_j}{h_{j+1}} - \frac{h_{j+1}}{3}M_j - \frac{h_{j+1}}{6}M_{j+1} \tag{154}$$

und wegen $S'_\Delta(f, x_j^+) = S'_\Delta(f, x_j^-)$

$$\frac{h_j}{6}M_{j-1} + \frac{h_j + h_{j+1}}{3}M_j + \frac{h_{j+1}}{6}M_{j+1} = \frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j} \tag{155}$$

für $j = 1, \ldots, n-1$. Auf diese Weise erhält man $n-1$ Bedingungen an den inneren Punkten für die $n+1$ Unbekannten $M_0, \ldots, M_n$. Je nach Randbedingungen a), b), c) erählt man die fehlenden zwei Bedingungen über

a) natürliche Randbedingungen:

$$S''_\Delta(f, a) = M_0 = 0 = M_N = S''_\Delta(f, b) \tag{156}$$

b) periodische Randbedingungen:

$$S''_\Delta(f, a) = S''_\Delta(f, b) \Rightarrow M_0 = M_N \text{ und} \tag{157}$$

$$S'_\Delta(f, a) = S'_\Delta(f, b) \Rightarrow \frac{h_n}{6}M_{n-1} + \frac{h_n + h_1}{3}M_n + \frac{h_1}{6}M_1 = \frac{f_1 - f_n}{h_1} - \frac{f_n - f_{n-1}}{h_n} \tag{158}$$

was zu den inneren Bedingungen mit $j = n$ für $h_{n+1} = h_1, M_{n+1} = M_1$ und $f_{n+1} = f_1$ (es gilt ja $f_n = f_0$) passt

c) vollständige Randbedingungen:

$$S'_\Delta(f, a) = f'_0 \Rightarrow \frac{h_1}{3}M_0 + \frac{h_1}{6}M_1 = \frac{f_1 - f_0}{h_1} - f'_0 \tag{159}$$

$$S'_\Delta(f, b) = f'_n \Rightarrow \frac{h_n}{6}M_{n-1} + \frac{h_n}{3}M_n = f'_n - \frac{f_n - f_{n-1}}{h_n} \tag{160}$$

2.3.3 Entstehende lineare Gleichungssysteme

In kompakter Matrixschreibweise erhält man damit im Fall a) und c):

$$\begin{pmatrix} 2 & \lambda_0 & & \\ \mu_1 & \ddots & \ddots & \\ & \ddots & \ddots & \lambda_{n-1} \\ & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_n \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_n \end{pmatrix} \tag{161}$$

mit

$$\lambda_j = \frac{h_{j+1}}{h_j + h_{j+1}} \tag{162}$$

$$\mu_j = 1 - \lambda_j = \frac{h_j}{h_j + h_{j+1}} \tag{163}$$

$$d_j = \frac{6}{h_j + h_{j+1}} \left(\frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j} \right) \tag{164}$$

für $j = 1 \ldots n-1$ und

$$\text{im Fall a)} \quad \lambda_0 = 0, d_0 = 0, \mu_n = 0, d_n = 0 \tag{165}$$

$$\text{im Fall c)} \quad \lambda_0 = 1, d_0 = \frac{6}{h_1} \left(\frac{f_1 - f_0}{h_1} - f'_0 \right) \tag{166}$$

$$\mu_n = 1, d_n = \frac{6}{h_n} \left(f'_n - \frac{f_n - f_{n-1}}{h_n} \right) \tag{167}$$

Im Fall b) erhält man:

$$\begin{pmatrix} 2 & \lambda_1 & & \mu_1 \\ \mu_2 & \ddots & \ddots & \\ & \ddots & \ddots & \lambda_{n-1} \\ \lambda_n & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} \tag{168}$$

mit

$$\lambda_n = \frac{h_1}{h_n + h_1} \tag{169}$$

$$\mu_n = 1 - \lambda_n = \frac{h_n}{h_n + h_1} \tag{170}$$

$$d_n = \frac{6}{h_n + h_1} \left(\frac{f_1 - f_n}{h_1} - \frac{f_n - f_{n-1}}{h_1} \right) \tag{171}$$

und zudem $M_0 = M_n$.

Satz 2.25: Die obigen Matrizen sind für jede Zerlegung Δ von $[a, b]$ nichtsingulär.

Beweis (Skizze):

- die Koeffizienten der Matrizen sind wohlbestimmt und hängen nur von Δ ab.
- es gilt für alle λ_j, μ_j: $\lambda_j \geq 0, \mu_j \geq 0$ und $\lambda_j + \mu_j = 1$ wodurch die Matrizen sogenannte *positive Matrizen* sind.

Damit sind die resultierenden linearen Gleichungssysteme für beliebige rechte Seiten eindeutig lösbar und das Spline-Interpolationsproblem besitzt in den Fällen a), b) und c) eine eindeutig bestimmte Lösung.

Die Gleichungssysteme lassen sich mit dem Gauss-Eliminationsverfahren in $O(n)$ Operationen lösen.

Algorithmus 2.26: (Berechnung eines interpolierenden kubischen Splines in Momenten-Darstellung):

```
q[0]=-lambda[0]/2; u[0]=d[0]/2; lambda[n]=0;
for k=1 to n
  p[k]=mu[k]*q[k-1]+2;
  q[k]=-lambda[k]/p[k];
  u[k]=(d[k]-mu[k]*u[k-1])/p[k];
M[n]=u[n];
for k=n-1 to 0
  M[k]=q[k]*M[k+1]+u[k];
```

Dieser Algorithmus funktioniert für die Fälle a) und c), den Fall b) kann man nach dem gleichen Prinzip lösen, allerdings nicht ganz so einfach. Man kann zeigen, dass $p[k] > 0$ ist, also der Algorithmus wohldefiniert ist.

2.3.4 Konvergenzeigenschaften kubischer Splines

Die Frage ist nun: konvergiert der interpolierende Spline $S_\Delta(f, \cdot)$ gegen die kontinuierliche Funktion f, wenn $f_i = f(x_i)$ und $h_\Delta = \max\limits_{1 \leq j \leq n} h_j \to 0$?

Zunächst: die Momente M_j einer interpolierenden Splinefunktion konvergieren gegen die zweite Ableitung von f. Im Fall c) sei $M = (M_0, \ldots, M_N)^T$, $d = (d_0, \ldots, d_n)^T$ sodass

$$AM = d \tag{172}$$

gilt. Für $F = (f''(x_0), \ldots, f''(x_n))^T$ betrachte das Residuum $r = d - AF = A(M - F)$, dann gilt:

Satz 2.27: Für $f \in C^4[a,b]$ und $|f^{(4)}(x)| \leq c$ für $x \in [a,b]$ gilt

$$\|M - F\| \leq \|r\| \leq \frac{3}{4} \cdot c \cdot h_\Delta^2 \tag{173}$$

wobei $\|v\| = \max_j |v_j|$.

Ohne Beweis (Hinweis: Taylor-Entwicklung)

Satz 2.28: Für $f \in C^4[a,b]$ und $|f^{(4)}(x)| \leq c$ für $x \in [a,b]$, der zu f interpolierenden Splinefunktion $S_\Delta(f,x)$ und einer Konstante κ sodass

$$\frac{h_\Delta}{h_j} \leq \kappa \text{ für } j = 0, \ldots, n-1 \tag{174}$$

dann gibt es von Δ unabhängige Konstanten $c_k \leq 2$, sodass für $x \in [a,b]$ gilt:

$$|f^{(k)}(x) - S_\Delta^{(k)}(x)| \leq c_k \cdot c \cdot \kappa \cdot h^{4-k} \text{ für } k = 0, 1, 2, 3 \tag{175}$$

Ohne Beweis (wieder über Taylor-Entwicklung)

Folgerung: für eine Folge von Zerlegungen Δ_m mit $h_{\Delta_m} \to 0$ und zugehörigen $\kappa_m < \infty$ konvergieren die Splinefunktionen S_{Δ_m} und ihre ersten drei Ableitungen gegen die Funktion f und ihre Ableitungen.

3 Numerische Integration

Ziel der numerischen Integration ist die Berechnung bestimmter Integrale der Form

$$\int_a^b f(x)\, dx\ . \tag{176}$$

Idealerweise werden solche Integrale in geschlossener Form über die Stammfunktion F von f als $F(b) - F(a)$ berechnet. Eine numerische Integration ist sinnvoll/notwendig, wenn

- keine analytische Lösung möglich ist (Beispiel: e^{-x^2}) oder
- die analytische Lösung zu aufwändig auszuwerten wäre

3.1 Quadraturformeln

3.1.1 Elementare Quadraturformeln

Die Idee der elementaren Quadraturformeln ist es

- den exakten Integranden f durch einen Interpolanden P bzgl. einer Partition Δ von $[a,b]$ zu ersetzen (siehe Kapitel 2) und
- den Interpolanden exakt zu integrieren

Im Prinzip kann hierzu jedes im vorigen Kapitel besprochene Interpolationsscheme verwendet werden.

Wählt man eine äquidistante Zerlegung $\Delta = \{x_i : x_i = a + ih, i = 0 \dots n\}$ mit $h = (b-a)/n$ und den Polynominterpoland: $P_n(x_i) = f_i = f(x_i)$ für $i = 0 \dots n$, in Lagrange-Darstellung

$$P_n(x) = \sum_{i=0}^{n} f_i L_i(x) \text{ mit } L_i(x) = \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \tag{177}$$

dann ist:

$$\int_a^b f(x)\ dx \approx \int_a^b P_n(x)\ dx = \sum_{i=0}^{n} f_i \int_a^b L_i(x)\ dx = \sum_{i=0}^{n} \alpha_i f_i \tag{178}$$

mit Gewichten

$$\alpha_i = \int_a^b L_i(x)\ dx = h \int_0^n \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{t-j}{i-j}\ dt \text{ (Substition } x = a + ht) \tag{179}$$

Beispiel: $(n = 2)$:

- $a_0 = h \displaystyle\int_0^2 \frac{t-1}{0-1} \cdot \frac{t-2}{0-2}\ dt = \frac{h}{2} \int_0^2 (t-1)(t-2)\ dt = \frac{h}{2} \cdot \frac{2}{3} = \frac{h}{3}$
- $a_1 = h \displaystyle\int_0^2 \frac{t-0}{1-0} \cdot \frac{t-2}{1-2}\ dt = -h \int_0^2 t(t-2)\ dt = \frac{4h}{3}$
- $a_2 = h \displaystyle\int_0^2 \frac{t-0}{2-0} \cdot \frac{t-1}{2-1}\ dt = \frac{h}{2} \int_0^2 t(t-1)\ dt = \frac{h}{3}$

und damit

$$\int_a^b P_2(x)\ dx = \frac{h}{3}(f_0 + 4f_1 + f_2) \tag{180}$$

was die sogenannte *Simpson-Regel* darstellt. Die Wahl von allgemeinem n führt zu den *Newton-Cotes-Formeln*

$$\int_a^b P_n(x)\ dx = \sum_{i=0}^{n} f_i \alpha_i = \frac{b-1}{n \cdot s} \sum_{i=0}^{n} \sigma_i f_i \tag{181}$$

wobei $\alpha_i = h\frac{\sigma_i}{s}$ und s der Hauptnenner der rationalen Zahlen α_i/h darstellt. Auf diese Weise sind die Gewichte σ_i ganze Zahlen. Die Gewichte α_i bzw. σ_i muss man nur einmalig pro Quadraturformel ausrechnen.

Satz 3.1: Die Gewichte α_i bzw. σ_i bilden eine Partition der Eins, d.h.

$$\sum_{i=0}^{n} \alpha_i = 1 \cdot (b-a) \tag{182}$$

$$\sum_{i=0}^{n} \sigma_i = sn \tag{183}$$

Beweis: folgt aus der Eindeutigkeit des Interpolationspolynoms mit $f(x) = 1$ also $P_n(x) = 1$ und

$$\sum_{i=0}^{n} \sigma_i = \sum_{i=0}^{n} \frac{\alpha_i s}{h} = \frac{s}{h} \sum_{i=0}^{n} \alpha_i = sn \frac{b-a}{b-a} = sn$$

□

Beispiel ($n = 2$): $s = 3$, $\alpha = (\frac{h}{3}, \frac{4h}{3}, \frac{h}{3})$ und $\sigma = (1, 4, 1)$

Satz 3.2: Falls f genügend oft differenzierbar ist, dann besitzen die Newton-Cotes-Formeln folgende Fehlerdarstellung:

$$\int_a^b P_n(x)\ dx - \int_a^b f(x)\ dx = Kh^{p+1} f^{(p)}(\xi) \text{ mit } \xi \in [a, b] \tag{184}$$

wobei K und p von n aber nicht von f abhängen.

Beweis: Taylor-Entwicklung (siehe auch Satz 2.7). □

Alle Newton-Cotes-Formeln:

n	σ_i	$n \cdot s$	Fehler	Name
1	1 1	2	$h^3 \frac{1}{12} f^{(2)}(\xi)$	Trapezregel
2	1 4 1	6	$h^5 \frac{1}{90} f^{(4)}(\xi)$	Simpsonregel
3	1 3 3 1	8	$h^5 \frac{3}{80} f^{(4)}(\xi)$	Fassregel (3/8-Regel)
4	7 32 12 32 7	90	$h^7 \frac{8}{945} f^{(6)}(\xi)$	Milne-Regel
5	19 75 50 50 75 19	288	$h^7 \frac{275}{12096} f^{(6)}(\xi)$	-
6	41 216 27 272 27 216 41	840	$h^9 \frac{9}{1400} f^{(8)}(\xi)$	Weddle-Regel

Für größere $n > 6$ treten leider negative Gewichte σ_i auf, was zu numerisch instabilen Verfahren führt.

In analoger Weise lassen sich auch Quadraturformeln für Hermite-Interpolanden berechnen.

3.1.2 Iterierte Quadraturformeln

Statt immer höhere Polynome zur Quadratur zu verwenden, wird stattdessen oft stückweise vorgegangen:

- das Gesamtintervall $[a, b]$ wird in N Teilintervalle $[x_i, x_{i+1}]$, $x_i = a + ih$, $0 \le i \le N$ zerlegt
- in den Teilintervallen werden jeweils die elementaren Quadraturformeln angewandt
- das Gesamtintegral ergibt sich dann als Summe der Teilintegrale

Bei der *Trapezsumme* ($n = 1$) ergibt sich als Näherungsformel für jedes der N Teilintervalle $[x_i, x_{i+1}]$

$$\int_{x_i}^{x_{i+1}} f(x)\ dx \approx I_i = \frac{h}{2}(f(x_i) + f(x_{i+1})) \tag{185}$$

und somit für das gesamte Intervall $[a, b]$:

$$T(h) = \sum_{i=0}^{N-1} I_i = h\left(\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \ldots + f(b-h) + \frac{1}{2}f(b)\right) \tag{186}$$

Beispiel: $(n = 1, N = 2, h = (b-a)/2)$

$$T(h) = I_0 + I_1 = \frac{(b-a)}{2}\left(\frac{1}{2}f(a) + f\left(\frac{a+b}{2}\right) + \frac{1}{2}f(b)\right) \tag{187}$$

Satz 3.3: Für $f \in C^2[a,b]$ gilt für den Quadraturfehler der Trapezsumme:

$$T(h) - \int_a^b f(x)\ dx = \frac{h^2(b-a)}{12} f^{(2)}(\xi) \text{ mit } \xi \in [a,b] \tag{188}$$

Beweis: In jedem Teilintervall $[x_i, x_{i+1}]$ ist

$$I_i - \int_{x_i}^{x_{i+1}} f(x)\ dx = \frac{h^3}{12}\ f^{(2)}(\xi_i) \text{ mit } \xi_i \in [x_i, x_{i+1}]$$

und somit für $f \in C^2[a,b]$

$$T(h) - \int_a^b f(x)\ dx = \frac{h^2}{12}(b-a) \cdot \sum_{i=0}^{N-1} \frac{1}{N} f^{(2)}(\xi_i) = \frac{h^2(b-a)}{12}\ f^{(2)}(\xi) \text{ mit } \xi \in [a,b]$$

da

$$\min_{0 \le i \le N-1} f^{(2)}(\xi_i) \le \frac{1}{N} \sum_{i=0}^{N-1} f^{(2)}(\xi_i) \le \max_{0 \le i \le N-1} f^{(2)}(\xi_i)$$

und da $f^{(2)}$ stetig ist, gibt es ein

$$\xi \in \left(\min_i \xi_i, \max_i \xi_i\right) \subset [a,b]$$

mit

$$f^{(2)}(\xi) = \frac{1}{N} \sum_{i=0}^{N-1} f^{(2)}(\xi_i)$$

Anmerkung: Die Trapezsumme als wiederholte (iterierte) Anwendung der Trapezregel ist ein Verfahren zweiter Ordnung (h^2) und exakt für alle Polynome vom Grad 2.

Bei der *Simpsonsumme* $(n = 2)$ ergibt sich als Näherungsformel für jedes der $N/2$ (N gerade) Teilintervalle $[x_{2i}, x_{2i+2}]$, $i = 0, \ldots N/2 - 1$

$$I_i = \frac{h}{3}(f(x_{2i}) + 4f(x_{2i+1}) + f(x_{2i+2})) \tag{189}$$

und somit für das gesamte Intervall $[a,b]$:

$$S(h) = \sum_{i=0}^{N/2-1} I_i = \frac{h}{3}\left(f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + \ldots + 2f(b-2h) + 4f(b-h) + f(b)\right) \tag{190}$$

Satz 3.4: Für $f \in C^4[a,b]$ gilt für den Quadraturfehler der Simpsonsumme:

$$S(h) - \int_a^b f(x)\ dx = \frac{(b-a)}{180} h^4 f^{(4)}(\xi) \text{ mit } \xi \in [a,b] \tag{191}$$

Beweis: wie Satz 3.3. □

Die Simposonsumme ist also ein Verfahren vierter Ordnung.

Algorithmus 3.5 (Trapezsumme)
Eingabe: Funktion `f`, Intervallgrenzen `a`, `b`, Zahl der Teilintervalle `N`
Ausgabe: Näherungswert `T` für das Integral von f in $[a, b]$

```
h=(b-a)/N
T=0.5*(f(a)+f(b))
for i=1 to N-1
  T=T+f(a+i*h)
T=T*h
```

Algorithmus 3.6 (Simpsonsumme)
Eingabe: Funktion `f`, Intervallgrenzen `a`, `b`, Zahl der Teilintervalle `N`
Ausgabe: Näherungswert `S` für das Integral von f in $[a, b]$

```
h=(b-a)/N
S=f(a)+f(b)
for i=1 to N/2
  S=S+4*f(a+(2*i-1)*h)
for i=1 to N/2-1
  S=S+2*f(a+2*i*h)
S=S*h/3
```

3.2 Fehlerdarstellung

Der Quadraturfehler $R(f)$ einer Quadraturformel $Q_n(f)$ mit n Stützstellen x_i und Gewichten α_i zur Berechnung des Integrals $I(f)$ ist gegeben als

$$R(f) = Q_n(f) - I(f) = \sum_{i=1}^{n} \alpha_i f(x_i) - \int_a^b f(x)\ dx \tag{192}$$

Den Integrationsfehler $R(f)$ kann man als lineares Funktional auffassen, d.h. $R(af + bg) = aR(f) + bR(g)$ für $f, g \in V$ wobei V ein beliebiger Funktionenraum (z.B. Π_n oder $C^n[a, b]$) sei, in dem R definiert ist und $a, b \in \mathbb{R}$. Nun werden Abschätzungen des Quadraturfehlers unter bestimmten Glattheitsbedingungen an f gesucht.

3.2.1 Peano-Fehlerdarstellung

Satz 3.7: Für alle Polynome $P \in \Pi_n$ gelte $R(P) = 0$, d.h. alle Polynome P vom Grad n werden durch die Quadraturformel $Q_n(f)$ exakt integriert, dann gilt für alle Funktionen $f \in C^{n+1}[a, b]$:

$$R(f) = \int_a^b f^{(n+1)}(t) K(t)\ dt \tag{193}$$

mit

$$K(t) = \frac{1}{n!} R_x((x - t)_+^n) \tag{194}$$

wobei

$$(x - t)_+^n = \begin{cases} (x - t)^n & \text{für } x \geq t \\ 0 & \text{sonst} \end{cases} \tag{195}$$

sogenannte *abgeschnittene Potenzfunktionen* sind. Dabei bedeutet, $R_x((x - t)_+^n)$, dass das Funktional R auf $(\cdot - t)_+^n$ als Funktion in x anzuwenden ist. $K(t)$ heisst *Peano-Kern* des Funktionals.

Beweis: Die Taylorentwicklung von $f(x)$ um $x = a$ ergibt

$$f(x) = f(a) + f'(a)(x-a) + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + r_n(x) \tag{196}$$

wobei das Restglied

$$r_n(x) = \frac{1}{n!}\int_a^x f^{(n+1)}(t)(x-t)^n \; dt \tag{197}$$

nach Definition der abgeschnittenen Potenzfunktionen auch in der Form

$$r_n(x) = \frac{1}{n!}\int_a^b f^{(n+1)}(t)(x-t)_+^n \; dt \tag{198}$$

geschrieben werden kann. Wendet man das das Funktional R auf die Taylor-Entwicklung an, so folgt wegen $R(P) = 0$ für $P \in \Pi_n$ sofort

$$R(f) = R(r_n) = \frac{1}{n!} R_x \left(\int_a^b f^{(n+1)}(t)(x-t)_+^n \; dt \right) \tag{199}$$

Nun ist der Integrand in $r_n(x)$ eine $(n-1)$-mal stetig differenzierbar und es gilt daher

$$\int_a^b \left(\int_a^b f^{(n+1)}(t)(x-t)_+^n \; dt \right) \; dx = \int_a^b f^{(n+1)}(t) \left(\int_a^b (x-t)_+^n \; dx \right) \; dt \tag{200}$$

sowie für $k < n$, da $(x-t)_+^n$ ebenfalls $(n-1)$-mal stetig differenzierbar ist

$$\frac{d^k}{dx^k} \left(\int_a^b f^{(n+1)}(t)(x-t)_+^n \; dt \right) = \int_a^b f^{(n+1)}(t) \frac{d^k}{dx^k} \left((x-t)_+^n \right) \; dt \tag{201}$$

und letztere Beziehung auch noch für $k = n$ richtig ist (1x partiell integrieren). Aufgrund der Struktur von R kann man das Funktional R_x mit dem Integral vertauschen, d.h.

$$R(f) = \frac{1}{n!}\int_a^b f^{(n+1)}(t) R_x((x-t)_+^n) \; dt = \int_a^b f^{(n+1)}(t) K(t) \; dt \tag{202}$$

□

In vielen Fällen besitzt der Peano-Kern K auf $[a, b]$ ein konstantes Vorzeichen, dann gilt mit dem Mittelwertsatz

$$R(f) = f^{(n+1)}(\xi) \int_a^b K(t) \; dt \text{ für } \xi \in (a, b) \tag{203}$$

Da das Integral über K nicht von f abhängt, gilt für $f \in C^{n+1}(a, b)$

$$R(f) = \frac{R(x^{n+1})}{(n+1)!} f^{(n+1)}(\xi) \text{ für } \xi \in (a, b) \tag{204}$$

Beispiel (Simpson-Regel in $[-1, 1]$):

$$\begin{aligned} I(f) &= \int_{-1}^1 f(x) \; dx \\ Q_3(f) &= \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1) \end{aligned}$$

ist exakt für alle kubischen Polynome. Die Anwendung von Satz 3.7 mit $n = 3$ führt zu

$$K(t) = \frac{1}{6} R_x((x-t)_+^3) = \frac{1}{6}\left(\frac{1}{3}(-1-t)_+^3 + \frac{4}{3}(0-t)_+^3 + \frac{1}{3}(1-t)_+^3 - \int_{-1}^{1} (x-t)_+^3 \, dx\right)$$

und

$$\int_{-1}^{1} (x-t)_+^3 \, dx = \int_{-1}^{1} (x-t)^3 \, dx = \frac{(1-t)^4}{4},$$

sowie

$$(-1-t)_+^3 = 0, \quad (1-t)_+^3 = (1-t)^3, \quad (-t)_+^3 = \begin{cases} 0 & \text{für } t \geq 0 \\ -t^3 & \text{sonst} \end{cases}$$

Der Peano-Kern der Simpson-Regel für das Intervall $[-1, 1]$ ist daher

$$K(t) = \begin{cases} \frac{1}{72}(1-t)^3(1+3t) & \text{für } 0 \leq t \leq 1 \\ K(-t) & \text{für } -1 \leq t \leq 0 \end{cases}$$

Da $K(t) \geq 0$ für $-1 \leq t \leq 1$ ist der Mittelwertsatz anwendbar und mit

$$\frac{R(x^4)}{4!} = \frac{1}{24}\left(\frac{1}{3} \cdot 1 + \frac{4}{3} \cdot 0 + \frac{1}{3} \cdot 1 - \int_{-1}^{1} x^4 \, dx\right) = \frac{1}{90}$$

gilt für das Restglied der Simpsonregel

$$Q_3(f) - I(f) = \frac{1}{90} f^{(4)}(\xi) \text{ mit } \xi \in (a, b)$$

Allgemein integrieren die Newton-Cotes-Formeln

- für ungerades n alle Polynome bis zum Grad n
- für gerades n alle Polynome bis zum Grad $n + 1$

und die Peano-Kerne aller Newton-Cotes-Formeln besitzen konstantes Vorzeichen, sodass gilt

$$R_n(f) = \begin{cases} \frac{R_n(x^{n+1})}{(n+1)!} f^{(n+1)}(\xi) & \text{für ungerades } n \\ \frac{R_n(x^{n+2})}{(n+2)!} f^{(n+2)}(\xi) & \text{für gerades } n \end{cases} \tag{205}$$

womit die Fehlerterme der Tabelle in Kapitel 3.1.1 bewiesen werden können.

3.2.2 Euler-McLaurin-Summenformel

Satz 3.8: Sei $g \in C^{2m+2}[0, 1]$, dann ist

$$\int_0^1 g(t) \, dt = \frac{g(0)}{2} + \frac{g(1)}{2} + \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!}(g^{(2k-1)}(0) - g^{(2k-1)}(1)) - \frac{B_{2m+2}}{(2m+2)!} g^{(2m+2)}(\xi) \tag{206}$$

mit $0 < \xi < 1$ und den *Bernoulli-Zahlen* B_k.

Die Bernoulli-Zahlen B_k sind die Werte der *Bernoulli-Polynome* $B_k(x)$ an der Stelle 0, also $B_k = B_k(0)$, wobei die Bernoulli-Polynome rekursiv durch $B_0(x) = 1$ und $B_k'(x) = kB_{k-1}(x)$ wobei $\int_0^1 B_k(x) \, dx = 0$ definiert sind.

Beispiele:

- $B_0(x) = 1, B_0 = 1$
- $B_1(x) = x - \frac{1}{2}, B_1 = -\frac{1}{2}$
- $B_2(x) = x^2 - x + \frac{1}{6}, B_2 = \frac{1}{6}$
- $B_3(x) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x, B_2 = 0$ (und alle ungeraden $B_{2k+1} = 0$ für $k > 1$)

Beweis: Es wird $\int_0^1$ durch partielle Integration sukzessive umgeformt:

$$\begin{aligned}
\int_0^1 g(t)\ dt &= [B_1(t)g(t)]_0^1 - \int_0^1 B_1(t)g'(t)\ dt \\
\int_0^1 B_1(t)g'(t)\ dt &= \left[\frac{1}{2}B_2(t)g'(t)\right]_0^1 - \frac{1}{2}\int_0^1 B_2(t)g''(t)\ dt \\
&\cdots \\
\int_0^1 B_{k-1}(t)g^{(k-1)}(t)\ dt &= \left[\frac{1}{k}B_k(t)g^{(k-1)}(t)\right]_0^1 - \frac{1}{k}\int_0^1 B_k(t)g^{(k)}(t)\ dt
\end{aligned}$$

Nun gilt wegen $B_k(0) = B_k(1) = B_k$ für $k > 1$

$$\left[\frac{1}{k}B_k(t)g^{(k-1)}(t)\right]_0^1 = -\frac{B_k}{k}(g^{(k-1)}(0) - g^{(k-1)}(1))$$

und wegen $B_{2k+1} = 0$

$$\int_0^1 g(t)\ dt = \frac{g(0)}{2} + \frac{g(1)}{2} + \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!}(g^{(2k-1)}(0) - g^{(2k-1)}(1)) + r_{m+1}$$

mit dem Restglied

$$r_{m+1} = \frac{-1}{(2m+1)!}\int_0^1 B_{2m+1}(t)g^{(2m+1)}(t)\ dt$$

Nochmalige partielle Integration führt zu

$$\begin{aligned}
\int_0^1 B_{2m+1}(t)g^{(2m+1)}(t)\ dt &= \left[\frac{1}{2m+2}(B_{2m+2}(t) - B_{2m+2})g^{(2m+1)}(t)\right]_0^1 \\
&\quad -\frac{1}{2m+2}\int_0^1 (B_{2m+2}(t) - B_{2m+2})g^{(2m+2)}(t)\ dt
\end{aligned}$$

wobei der erste Term wegen $B_{2m+2}(0) = B_{2m+2}(1) = 0$ verschwindet und da $B_{2m+2}(t) - B_{2m+2}$ das Vorzeichen auf $[0, 1]$ nicht ändert (kann man durch Induktion zeigen) und $\int_0^1 B_{2m+2}(t)\ dt = 0$ folgt mit dem Mittelwertsatz die Behauptung. □

Wendet man die Euler-McLaurin-Summenformel wiederholt auf $\int_{x_i}^{x_{i-1}} g(t)\ dt$ an und summiert über i, dann erhält man

$$\frac{g(0)}{2} + g(1) + \ldots + g(N-1) + \frac{g(N)}{2} = \int_0^N g(t)\ dt + \sum_{k=1}^{m} \frac{B_{2k}}{(2k)!}(g^{(2k-1)}(N) - g^{(2k-1)}(0)) + \frac{B_{2m+2}}{(2m+2)!}Ng^{(2m+2)}(\xi) \tag{207}$$

mit $0 < \xi < N$. Für ein beliebiges Intervall $[a, b]$ mit äquidistanten Punkten $x_i = a + ih$, $i = 0, \ldots, N$, $h = (b-a)/N$ erhält man für $f \in C^{2m+2}[a, b]$

$$T(h) = \int_a^b f(t)\ dt + \sum_{k=1}^{m} h^{2k}\frac{B_{2k}}{(2k)!}(f^{(2k-1)}(b) - f^{(2k-1)}(a)) + h^{2m+2}\frac{B_{2m+2}}{(2m+2)!}(b-a)f^{(2m+2)}(\xi) \tag{208}$$

mit $a < \xi < b$, wobei $T(h)$ die Trapezsumme zur Schrittweite h ist.

Anmerkungen:

- Damit ist eine Entwicklung von $T(h)$ in Potenzen von h gegeben, was die Grundlage für Extrapolationsverfahren bildet
- Spezialfälle der Euler-McLaurin-Formel bilden die Grundlage für Fehlerdarstellungen von Quadraturformeln basierend auf Hermite-Interpolation
- für periodische Funktionen $f^{(k)}(a) = f^{(k)}(b)$ für $0 \leq k \leq K$ fällt der mittlere Term weg und der letzte Term definiert die Ordnung des Quadraturverfahrens.

3.2.3 Extrapolation

Extrapolation ist ein allgemeines Konzept um die Ordnung von Approximationsverfahren zu erhöhen.

Vorraussetzungen:

- Approximationen auf verschiedenen Gittern mit Maschenweiten h
- Resultat der Approximation $T(h)$ besitzt eine sogenannte *asymptotische Entwicklung* der Form

$$T(h) = \tau_0 + \tau_1 h^{\gamma_1} + \tau_2 h^{\gamma_2} + \ldots + \tau_m h^{\gamma_m} + \alpha_{m+1}(h) h^{\gamma_{m+1}} \tag{209}$$

  mit
  - den Ordnungen der Entwicklung γ_i bekannt (oft sogar ganzzahlig)
  - den Koeffizienten der Entwicklung τ_i (von h unabhängig)
  - dem höchsten Koeffizient $\alpha_{m+1}(h)$ beschränkt für $h \to 0$ ($|\alpha_{m+1}(h)| \leq M_{m+1}$ für $|h| \leq H$) und
  - der exakten Lösung des kontinuierlichen Problems $\tau_0 = \lim_{h \to 0} T(h)$
- die Existenz einer solchen Entwicklung ist i.a. von bestimmten Glattheitsbedingungen an die Lösung τ_0 abhängig

Der *führende Term* des Approximationsfehlers $T(h) - \tau_0$ zum Gitter mit Maschenweite h ist $\tau_1 h^{\gamma_1}$. Die Idee von Extrapolationsverfahren ist nun die Elimination von $\tau_1 h^{\gamma_1}$ durch Linearkombination zweier Approximationen

$$\alpha T(h_i) + \beta T(h_{i+1}) \tag{210}$$

sodass

- τ_0 erhalten bleibt und
- der führende Term verschwindet.

Auf diese Weise hat die Linearkombination bezüglich des Fehlers eine höhere Fehlerordnung bei gleicher Aufwandsordnung (Aufwand etwa doppelt so hoch).

Beispiel: $h_i = 2h_{i+1}$, $\gamma_1 = 2$, $\gamma_2 = 4$

$$\begin{aligned} T(h_i) &= \tau_0 + \tau_1 h_i^2 + \tau_2 h_i^4 + \text{Terme höherer Ordnung} \\ T(\frac{h_i}{2}) &= \tau_0 + \tau_1 \frac{h_i^2}{4} + \tau_2 \frac{h_i^4}{16} + \text{Terme höherer Ordnung} \end{aligned}$$

Für die Linearkombination $\alpha T(h_i) + \beta T(\frac{h_i}{2})$ muss gelten:

$$\begin{aligned} \alpha\tau_0 + \beta\tau_0 &= \tau_0 \\ \alpha\tau_1 h_i^2 + \beta\tau_1 \frac{h_i^2}{4} &= 0 \end{aligned}$$

also

$$\begin{aligned} \alpha + \beta &= 1 \\ \alpha + \frac{\beta}{4} &= 0 \end{aligned}$$

und somit $4\alpha = \beta \Rightarrow \alpha = -\frac{1}{3}, \beta = \frac{4}{3}$ mit dem Ergebnis

$$\frac{4}{3}T(h_{i+1}) - \frac{1}{3}T(h_i) = \tau_0 + 0 - 4h_{i+1}^4 + \text{Terme höherer Ordnung}$$

Diese Idee kann auch sukzessive angewandt werden, um höhere Fehlerterme verschwinden zu lassen.

Für die numerische Integration einer Funktion $f \in C^{2m+2}[a,b]$ ist die Euler-McLaurin Summenformel genau so eine asymptische Entwicklung

$$T(h) = \tau_0 + \tau_1 h^2 + \ldots + \tau_m h^{2m} + \alpha_{m+1}(h)h^{2m+2} \tag{211}$$

mit

$$\begin{aligned} \tau_0 &= \int_a^b f(x)\ dx \\ \tau_k &= \frac{B_{2k}}{(2k)!}(f^{(2k-1)}(b) - f^{(2k-1)}(a)) \text{ für } k = 1 \ldots m \text{ unabhängig von } h \\ \alpha_{m+1}(h) &= \frac{B_{2m+2}}{(2m+2)!}(b-a)f^{(2m+2)}(\xi(h)) \text{ mit } a < \xi(h) < b \end{aligned}$$

wobei $\alpha_{m+1}(h)$ eine beschränkte Funktion von h ist, d.h. es existiert eine Konstante M_{m+1}

$$M_{m+1} = \left|\frac{B_{2m+2}}{(2m+2)!}(b-a)\right| \cdot \max\left|f^{(2m+2)}(x)\right| \tag{212}$$

mit $|\alpha_{m+1}(h)| \leq M_{m+1}$ für alle $h = (b-a)/n, n = 1, 2, \ldots$.

Anmerkungen:

- selbst für $f \in C^\infty[a,b]$ kann die Reihe $\tau_0 + \tau_1 h^2 + \tau_2 h^4$ für jedes $h \neq 0$ divergieren.
- trotzdem behält die asymptotische Entwicklung ihren Wert, da man den Fehlerterm für kleines h gegenüber den übrigen Termen vernachlässigen kann
- die Approximation $T(h)$ verhält sich für kleines h wie ein Polynom in h^2 dessen Wert an der Stelle $h = 0$ das Integral τ_0 liefert

3.2.4 Romberg-Integration

Die Romberg-Integration ist eine Anwendung der Extrapolation auf die Integration. Dazu betrachtet man

- eine Folge $n_0, n_1, n_2, \ldots n_m$ ganzer Zahlen

- zu einer Reihe von Maschenweiten
$$h_0 = \frac{b-a}{n_0},\ h_1 = \frac{h_0}{n_1},\ \ldots h_m = \frac{h_0}{n_m},\ \ldots \tag{213}$$
- man errechnet die zugehörigen Trapezsummen $T(h_0), T(h_1), \ldots T(h_m)$ und setzt $T_{i0} = T(h_i), 0 \leq i \leq m$
- nun bestimmt man durch Interpolation das Polynom in h^2 höchstens m-ten Grades
$$\tilde{T}_{mm}(h) = a_0 + a_1 h^2 + \ldots + a_m h^{2m} \tag{214}$$
für das gilt
$$\tilde{T}_{mm}(h_i) = T(h_i),\ 0 \leq i \leq m \tag{215}$$
- der extrapolierte Wert $\tilde{T}_{mm}(0)$ ist dann eine bessere Näherung für den Wert des Integrals.

Bild Berechnete und interpolierte Werte von $T(h)$ für $h = h_0, h_1, \ldots h_m$ und $h = 0$

Konkret wird $\tilde{T}_{mm}(h)$ durch ein Aitken-Neville-Schema berechnet. Dabei sei $\tilde{T}_{ik}(h)$ mit $1 \leq k \leq i \leq m$ das Polynom vom Grad $\leq k$ in h^2 mit
$$\tilde{T}_{ik}(h_j) = T(h_j) \text{ für } j = i-k, i-k-1, \ldots, i \tag{216}$$
Dann gilt für die extrapolierten Werte
$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{h_{i-k}}{h_i}\right)^2 - 1} \text{ für } 1 \leq k \leq i \leq m \tag{217}$$
die spaltenweise über das Tableau
$$\begin{array}{c|ccccc} h_0^2 & T(h_0) = T_{00} & & & & \\ & & T_{11} & & & \\ h_1^2 & T(h_1) = T_{10} & & T_{22} & & \\ & & T_{21} & & \ddots & \\ h_2^2 & T(h_2) = T_{20} & & \ddots & & \\ & & \ddots & & & \\ \vdots & \vdots & & & & \end{array} \tag{218}$$
berechnet werden können.

Anmerkung: Manche T_{ik} enstsprechen Newton-Cotes-Formeln, z.B.

- T_{11} ist für $h_1 = h_0/2 = (b-a)/2$ die Simpson-Regel und
- T_{22} ist für $h_2 = h_1/2 = h_0/4 = (b-a)/4$ die Milne-Regel

Algorithmus 3.9 (Romberg-Verfahren)
Eingabe: Funktion `f`, Intervallgrenzen `a`, `b`, Zahl der Extrapolationsschritte `N`
Ausgabe: Näherungswert `S` für das Integral von f in $[a, b]$

```
T[0,0]=0.5*h*(f(a)+f(b))
for n=1 to N-1
  T[n][0]=trapezsumme(f,a,b,n)
  for k=1 to n
    s=1+n-k
    p=pow(4,k)
    T[s][k] = (p*I[s][k-1]-I[s-1][k-1])/(p-1.0)
S=T[N-1][0]
```

4 Lineare Gleichungssysteme

4.1 Vektoren und Matrizen

4.1.1 Vektoren und Matrizen

Grundlegende Definitionen (tiefgestellte Indizes bezeichnen Vektor/Matrix-Einträge, hochgestellte verschiedene Vektoren/Matrizen):

- Vektor $\mathbf{x} \in \mathbb{R}^n$: $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ mit $x_i \in \mathbb{R}$.
- Matrix $\mathbf{A} \in \mathbb{R}^{m,n}$: $\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$ mit $a_{ij} \in \mathbb{R}$.
- transponierte Matrix $\mathbf{A}^T$: $(\mathbf{A}^T)_{ij} = a_{ji}$:
- quadratische Matrix der Ordnung/Größe m: $\mathbf{A} \in \mathbb{R}^{m,m}$
- $span\{\mathbf{x}^1, \dots, \mathbf{x}^n\}$: von $\mathbf{x}^1, \dots, \mathbf{x}^n$ aufgespannter Untervektorraum
- Einheits-Matrix $\mathbf{I}_n = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}$ mit Einheitsvektoren $\mathbf{e}^1, \dots, \mathbf{e}^n$ als Spalten
- Diagonal-Matrix: $\mathbf{D} = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix} = diag(d_1, \dots, d_n)$
- Tridiagonal-Matrix: $\mathbf{T} = \begin{pmatrix} t_{11} & t_{12} & & 0 \\ t_{21} & t_{22} & t_{23} & \\ & \ddots & \ddots & \ddots \\ 0 & & t_{n,n-1} & t_{nn} \end{pmatrix}$, d.h. $t_{ij} = 0$ für $|i-j| > 1$
- untere Dreiecksmatrix: $\mathbf{L} = \begin{pmatrix} l_{11} & & 0 \\ \vdots & \ddots & \\ l_{n1} & \dots & l_{nn} \end{pmatrix}$, d.h. $l_{ij} = 0$ für $i < j$
- strikt untere Dreiecksmatrix: $l_{11} = \ldots = l_{nn} = 0$
- normalisierte untere Dreiecksmatrix: $l_{11} = \ldots = l_{nn} = 1$
- obere Dreiecksmatrix: $\mathbf{R} = \mathbf{L}^T$, Definitionen von strikt und normalisiert analog
- inverse einer quadratischen Matrix $\mathbf{A}^{-1}$: $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$
- Determinante einer quadratischen Matrix:

$$det(\mathbf{A}) = \sum_p (-1)^p a_{1,p_1} \dots a_{n,p_n} \tag{219}$$

wobei die Summe über alle $n!$ Permutationen $(p_1, \ldots, p_n)$ der Zahlen $(1, \ldots, n)$ geht und $(-1)^p = \pm 1$, je nachdem ob eine gerade oder ungerade Zahl paarweise Tausche angewandt wurde

Determinanten-Regeln:

- $A, B \in \mathbb{R}^{n,n} \Rightarrow det(\mathbf{AB}) = det(\mathbf{BA}) = det(\mathbf{A}) \cdot det(\mathbf{B})$
- $det(\mathbf{D}) = d_1 \cdot \ldots \cdot d_n$
- $det(\mathbf{L}) = l_{11} \cdot \ldots \cdot l_{nn}$
- $det(\mathbf{R}) = r_{11} \cdot \ldots \cdot r_{nn}$

4.1.2 Normen

Im folgenden sind $|\cdot|, <, \leq, >, \geq$ komponentenweise zu verstehen, d.h.

$$|\mathbf{x}| = (|\mathbf{x}_1|, \ldots, |\mathbf{x}_n|)^T, |\mathbf{A}| \leq |\mathbf{B}| \Leftrightarrow |a_{ij}| \leq |b_{ij}| \text{ für alle } i, j \text{ usw.} \tag{220}$$

Die wichtigsten Normen:

- Summennorm: $\|\mathbf{x}\|_1 = \sum_i |\mathbf{x}_i|$
- Euklidische Norm: $\|\mathbf{x}\|_2 = \sqrt{\sum_i |\mathbf{x}_i|^2}$
- Maximumsnorm: $\|\mathbf{x}\|_\infty = \max_i |\mathbf{x}_i|$

sind jeweils Abbildungen: $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ mit Eigenschaften

- definit: $\mathbf{x} \neq 0 \Rightarrow \|\mathbf{x}\| > 0$
- homogen: $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$ für alle $\lambda \in \mathbb{R}$
- subadditiv: $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ (Dreiecksungleichung)

und es gilt die Cauchy-Schwarzsche Ungleichung

$$|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|_2\|\mathbf{y}\|_2. \tag{221}$$

Normen erlauben, den Abstand zwischen zwei Punkten eines Vektorraums zu definieren und somit das Einführen einer Metrik ($\to$ Topologisierung des Raums).

Die Menge $\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq 1\}$ heisst *Normkugel* (ohne i.a. rund zu sein).

Bild Normkugeln in 2 und 3 Raumdimensionen für $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$

In endlich-dimensionalen Vektorräumen sind alle Normen äquivalent, d.h. für zwei Normen $\|\cdot\|_a$, $\|\cdot\|_b$ gibt es zwei positive Konstanten α, β, sodass

$$\alpha\|\cdot\|_a \leq \|\cdot\|_b \leq \beta\|\cdot\|_a \tag{222}$$

Für Matrizen werden entsprechende *Operatornormen* durch

$$\|\mathbf{A}\| = \max_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \tag{223}$$

definiert, wobei hier für $\|\cdot\|$ jede der obigen Vektornormen eingesetzt werden kann.

Eigenschaften der Operatornorm:

- definit: $\mathbf{A} \neq 0 \Rightarrow \|\mathbf{A}\| > 0$
- homogen: $\|\lambda\mathbf{A}\| = |\lambda|\|\mathbf{A}\|$ für alle $\lambda \in \mathbb{R}$
- subadditiv: $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$
- submultiplikativ: $\|\mathbf{A} \cdot \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$
- Konsistenz von Matrix- und Vektornorm: $\|\mathbf{A} \cdot \mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$

Die wichtigsten Normen für $m \times n$-Matrizen sind

- Summennorm: $\|\mathbf{A}\|_1 = \max_j \sum_i |\mathbf{a_{ij}}|$
- Maximumsnorm: $\|\mathbf{A}\|_\infty = \max_i \sum_j |\mathbf{a_{ij}}|$
- Euklidische Norm: $\|\mathbf{A}\|_2 = \sqrt{\lambda_1}$ wobei λ_1 der größte Eigenwert von $\mathbf{A}^T\mathbf{A}$ ist

Für Operatornormen $\|\cdot\|$ gilt für alle Eigenwerte λ einer Matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ die Schranke

$$|\lambda| \leq \|\mathbf{A}\| \tag{224}$$

Die Euklidische Norm ist in der Regel aufwändig zu berechnen, deswegen wird oft als Ersatz die *Frobenius-Norm* verwendet

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}^2|} \tag{225}$$

also die Euklidische Norm, wenn $\mathbf{A}$ als Vektor aufgefasst wird. Sie ist keine Operatornorm, hat aber folgende Eigenschaften:

- $\|\mathbf{A}\|_F = spur(\mathbf{A}^T\mathbf{A}) = \lambda_1 + \lambda_2 + \ldots$
- $1 \leq \|\mathbf{A}\|_F / \|\mathbf{A}\|_2 \leq \sqrt{\min(m,n)}$
- definit, homogen, subadditiv
- submultiplikativ

Die Euklidische Norm kann weiterhin abgeschätzt werden durch

$$\|\mathbf{A}\|_2^2 \leq \|\mathbf{A}^T\mathbf{A}\|_\infty \leq \|\mathbf{A}^T\|_\infty \|\mathbf{A}\|_\infty = \|\mathbf{A}\|_1 \|\mathbf{A}\|_\infty \tag{226}$$

d.h. die Euklidische Norm ist nie größer als das geometrische Mittel aus Summen- und Maximumsnorm.

4.1.3 Kondition

Die *Kondition* einer Matrix ist definiert durch

$$\kappa(\mathbf{A}) = \frac{\max\limits_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|}{\min\limits_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|} \tag{227}$$

Sie gibt die Verzerrung der Normkugel unter der Abbildung $\mathbf{A}$ an.

Bild Verzerrung der Normkugel

Im Idealfall ist $\kappa(\mathbf{A}) = 1$, z.B. für orthogonale Matrizen. Umgekehrt ist $\kappa(\mathbf{A}) = \infty$ wenn $\mathbf{A}\mathbf{x} = 0$ für ein $\mathbf{x} \neq 0$.

Existiert $\mathbf{A}^{-1}$, dann gilt

$$\frac{1}{\min\limits_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|} = \max_{\|\mathbf{x}\|\neq 0} \frac{\|\mathbf{x}\|}{\|\mathbf{A}\mathbf{x}\|} = \max_{\|\mathbf{y}\|\neq 0} \frac{\|\mathbf{A}^{-1}\mathbf{y}\|}{\|\mathbf{y}\|} = \|\mathbf{A}^{-1}\| \tag{228}$$

also die Kondition ist

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \tag{229}$$

4.2 Elementare Matrix-Transformationen

4.2.1 Skalierung

Definition	$\mathbf{D} = diag(d_1, \ldots, d_n)$ mit $d_i \neq 0$
Anwendung	$(\mathbf{D}\mathbf{x})_i = d_i x_i$
	$(\mathbf{D}\mathbf{A})_{ij} = d_i a_{ij}$, d.h. die i-te Zeile wird mal d_i genommen
	$(\mathbf{A}\mathbf{D})_{ij} = a_{ij} d_j$, d.h. die j-te Spalte wird mal d_j genommen
Inverse	$\mathbf{D}^{-1} = diag(\frac{1}{d_1}, \ldots, \frac{1}{d_n})$
Determinante	$det(\mathbf{D}) = d_1 \cdot \ldots \cdot d_n$

Die folgenden drei Transformationsmatrizen stimmen mit der Einheitsmatrix I bis auf Extraelemente in den vier Positionen $(i,i), (i,j), (j,i), (j,j)$ überein.

4.2.2 Vertauschung

Definition	$\mathbf{P}(i,j)$ hat Extra-Elemente $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Anwendung	$\mathbf{P}(i,j)\mathbf{x}$ vertauscht Komponenten x_i und x_j
	$\mathbf{P}(i,j)\mathbf{A}$ vertauscht Zeilen i und j

$\mathbf{AP}(i,j)$ vertauscht Spalten i und j

Inverse $\mathbf{P}(i,j)^{-1} = P(i,j)$

Determinante $det(\mathbf{P}(i,j)) = -1$

4.2.3 Reihen-Operation

Definition $\mathbf{N}(i,j,\alpha)$ hat Extra-Element α an Position (i,j), d.h.
$\begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}$ falls $i > j$ und $\begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$ falls $i < j$

Anwendung $\mathbf{N}(i,j,\alpha)\mathbf{x})$ addiert α-mal Komponenten x_j zu Komponente x_i, d.h. $x_i \to x_i + \alpha x_j$

$\mathbf{N}(i,j,\alpha)\mathbf{A}$ addiert α-mal Zeile j zu Zeile i, d.h. $a_{ik} \to a_{ik} + \alpha a_{jk}$

$\mathbf{AN}(i,j,\alpha)$ addiert α-mal Spalte i zu Spalte j, d.h. $a_{kj} \to a_{kj} + \alpha a_{ki}$

Inverse $\mathbf{N}(i,j,\alpha)^{-1} = \mathbf{N}(i,j,-\alpha)$

Determinante $det(\mathbf{N}(i,j,\alpha)) = 1$

Die Reihen-Operationen $\mathbf{N}(i,j)$ zu einem festen j sind miteinander vertauschbar und es gilt

$$\prod_{\substack{1\le i\le n\\ i\neq j}} \mathbf{N}(i,j,\alpha_i) = \begin{pmatrix} 1 & & \alpha_1 & & 0 \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ 0 & & \alpha_n & & 1 \end{pmatrix} \tag{230}$$

und

$$\left(\prod_{\substack{1\le i\le n\\ i\neq j}} \mathbf{N}(i,j,\alpha_i)\right)^{-1} = \begin{pmatrix} 1 & & -\alpha_1 & & 0 \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ 0 & & -\alpha_n & & 1 \end{pmatrix} \tag{231}$$

sowie

$$\prod_{i>1} \mathbf{N}(i,1,\alpha_{i1}) \prod_{i>2} \mathbf{N}(i,2,\alpha_{i2}) \dots \prod_{i>n-1} \mathbf{N}(i,n-1,\alpha_{i,n-1}) = \begin{pmatrix} 1 & & & 0 \\ \alpha_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ \alpha_{n1} & \alpha_{n2} & \dots & 1 \end{pmatrix} \tag{232}$$

wobei hier die Reihenfolge der $n-1$ Teilprodukte wichtig ist.

4.2.4 Ebene Rotation

Definition $\mathbf{Q}_{ij}(\phi)$ hat Extra-Elemente $\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}$

Anwendung	$(\mathbf{Q}_{ij}(\phi)\mathbf{x})_i = x_i \cos\phi + x_j \sin\phi$
	$(\mathbf{Q}_{ij}(\phi)\mathbf{x})_j = -x_i \sin\phi + x_j \cos\phi$
	$(\mathbf{Q}_{ij}(\phi)\mathbf{A})$: Zeile i = cos Zeile i + sin ϕ Zeile j
	$(\mathbf{Q}_{ij}(\phi)\mathbf{A})$: Zeile j = − sin Zeile i + cos ϕ Zeile j
	$(\mathbf{A}\mathbf{Q}_{ij}(\phi))$: Spalte i = cos Spalte i − sin ϕ Spalte j
	$(\mathbf{A}\mathbf{Q}_{ij}(\phi))$: Spalte j = − sin Spalte i + cos ϕ Spalte j
Inverse	$\mathbf{Q}_{ij}(\phi)^{-1} = Q_{ij}(-\phi) = Q_{ij}(\phi)^T$, d.h. Q_{ij} ist orthogonal
Determinante	$det(\mathbf{Q}_{ij}(\phi)) = 1$

4.2.5 Spiegelung

Definition	normalisiert: $\mathbf{T} = I - 2\mathbf{v}\mathbf{v}^T$, wobei $\mathbf{v} \in \mathbb{R}^n$ und $\|\mathbf{v}\|_2 = 1$
	nicht normalisiert: $\mathbf{T} = I - 2\mathbf{u}\mathbf{u}^T/\kappa$, wobei $\mathbf{u} \in \mathbb{R}^n \subset \{0\}$ und $\kappa = \frac{1}{2}\mathbf{u}^T\mathbf{u}$
	Die Transformationsmatrix $\mathbf{T}$ ist voll: $t_{ij} = \delta_{ij} - 2v_iv_j = \delta_{ij} - u_iu_j/\kappa$
	Zur Anwendung von T müssen aber nur die Einträge von $\mathbf{v}$ bzw. $\mathbf{u}$ und κ gespeichert werden
Anwendung	$\mathbf{y} = \mathbf{T}\mathbf{x} = \mathbf{x} - \mathbf{u} \cdot (\mathbf{u}^T\mathbf{x})/\kappa$
	1. Schritt: $\sigma = \mathbf{u}^T\mathbf{x}/\kappa \in \mathbb{R}$
	2. Schritt: $\mathbf{y} = \mathbf{x} - \mathbf{u} \cdot \sigma$
	$\mathbf{TA} = \mathbf{A} - \mathbf{u} \cdot (\mathbf{u}^T\mathbf{A}/\kappa)$ jede Spalte von $\mathbf{TA}$ wie Vektor $\mathbf{y}$
	$\mathbf{AT} = \mathbf{A} - (\mathbf{A}\mathbf{u}/\kappa \cdot \mathbf{u}^T)$ jede Zeile von $\mathbf{AT}$ wie Vektor $\mathbf{y}$
Inverse	$\mathbf{T}^{-1} = T$, denn $\mathbf{T}^2 = \mathbf{I} - 2\mathbf{v}\mathbf{v}^T - 2\mathbf{v}\mathbf{v}^T + 4\mathbf{v}(\mathbf{v}^T\mathbf{v})\mathbf{v}^T = \mathbf{I}$ (wegen $\mathbf{v}^T\mathbf{v} = 1$)
Determinante	$det(\mathbf{T}) = 1$

Dass es sich hierbei um eine Spiegelung handelt, sieht man durch die Zerlegung $\mathbf{x} = \mathbf{s} + \mathbf{p}$ mit $\mathbf{p}$ parallel zu $\mathbf{v}$ und $\mathbf{s}$ senkrecht zu $\mathbf{v}$, also

$$\mathbf{p} = \mathbf{v}(\mathbf{v}^T\mathbf{x}) \text{ und } \mathbf{s} = \mathbf{x} - \mathbf{p} \tag{233}$$

denn dann gilt

$$\mathbf{v}^T\mathbf{s} = \mathbf{v}^T\mathbf{x} - \mathbf{v}^T\mathbf{p} = \mathbf{v}^T\mathbf{x} - \mathbf{v}^T\mathbf{v}\mathbf{v}^T\mathbf{x} = 0 \tag{234}$$

Damit ist

$$\mathbf{y} = \mathbf{T}\mathbf{x} = (\mathbf{I} - \mathbf{v}\mathbf{v}^T)(\mathbf{p} + \mathbf{s}) = \mathbf{p} + \mathbf{s} - 2\mathbf{p} = \mathbf{s} - \mathbf{p} \tag{235}$$

und $\mathbf{T}$ bildet $\mathbf{s}+\mathbf{p}$ nach $\mathbf{s}-\mathbf{p}$ ab, das einer Spiegelung von $\mathbf{x}$ an der Hyperebene senkrecht zu $\mathbf{v}$ entspricht.

Bild Darstellung einer Spiegelung

Umgekehrt erhält man aus $\mathbf{x}$ und $\mathbf{y} = \mathbf{T}\mathbf{x}$ den Vektor $\mathbf{v}$ (vorrausgesetzt $\|\mathbf{x}\| = \|\mathbf{y}\|$) da

$$\mathbf{x} - \mathbf{y} = (\mathbf{s} + \mathbf{p}) - (\mathbf{s} + \mathbf{p}) = 2\mathbf{p} \tag{236}$$

parallel zu $\mathbf{v}$ durch Normierung von $\mathbf{x} - \mathbf{y}$ auf 1:

$$\mathbf{v} = \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|} \tag{237}$$

4.3 Matrix-Zerlegungen

4.3.1 LR-Zerlegung

Ziel der Dreiecks- oder LR-Zerlegung ist die Faktorisierung einer Matrix

$$\mathbf{A} = \mathbf{L}\mathbf{R} \tag{238}$$

wobei $\mathbf{L}$ eine linke untere Dreiecksmatrix und $\mathbf{R}$ eine rechte obere Dreiecksmatrix sind. Der Hintergrund dabei ist, dass die Invertierung $\mathbf{L}^{-1}$ bzw. $\mathbf{R}^{-1}$ zur Lösung linearer Gleichungssysteme leicht durchführbar ist.

Das Vorgehen ist dabei wie folgt:

- sei $a_{11} \neq 0$ und $l_{21} = a_{21}/a_{11}$, dann besitzt $\mathbf{N}(2, 1, -l_{21})\mathbf{A}$ eine Null an der Position $(2, 1)$.
- durch analoge elementare Operationen werden weitere Nullen in der 1. Spalte abwärts erzeugt
- dann wird die zweite Spalte unterhalb der Diagonale reduziert
- usw. bis zur vorletzten Spalte

Algorithmus 4.1 (Transformation einer Matrix auf obere Dreiecksgestalt durch Reihenoperationen)

```
for j=1 to n-1
  for i=j+1 to n
    l(i,j)=A[i][j]/A[i][i]
    A=N(i,j,l(i,j))*A
```

Bild Gestalt von $\mathbf{A}$ vor dem (i, j)-ten Schritt

Die End-Matrix mit nur Nullen unterhalb der Diagonalen heisse $\mathbf{R}$ (als zusätzlichen Trick kann man auf den berechneten l-Werte an den 0 Plätzen speichern), also

$$\mathbf{R} = \mathbf{N}(n, n-1, -l(n, n-1)) \cdot \ldots \cdot \mathbf{N}(2, 1, -l(2, 1)) \cdot \mathbf{A} \tag{239}$$

und wegen $\mathbf{N}^{-1}(i, j, l) = \mathbf{N}(i, j, -l)$ gilt

$$\mathbf{A} = \mathbf{N}(2, 1, l(2, 1)) \cdot \ldots \cdot \mathbf{N}(n, n-1, l(n, n-1)) \cdot \mathbf{R} = \mathbf{L} \cdot \mathbf{R} \tag{240}$$

4.3.2 QR-Zerlegung

Ziel der orthogonalen Dreiecks- oder QR-Zerlegung ist die Faktorisierung einer Matrix

$$\mathbf{A} = \mathbf{QR} \tag{241}$$

wobei $\mathbf{Q}$ eine orthogonale Matrix und $\mathbf{R}$ eine rechte obere Dreiecksmatrix ist. Hintergrund ist die Berechnung der Eigenwerte (und Eigenvektoren) einer Matrix. Die Matrix $\mathbf{A}$ muss dabei nicht notwendigerweise quadratisch sein, meist ist $\mathbf{A} \in \mathbb{R}^{m,n}$ mit $m \geq n$.

Im Prinzip wird dabei wie bei der LR-Zerlegung vorgegangen, nur dass die Reihenoperationen $\mathbf{N}(i, j, -l(i, j)$ durch ebene Rotationen $\mathbf{Q}^T(i, j, \phi)$ ersetzt werden. Dabei wird der Drehwinkel ϕ in $\mathbf{Q}^T(i, j)$ so gesetzt, dass wieder an der Stelle (i, j) eine Null entsteht:

$$0 = -sa_{jj} + ca_{ij} \Leftrightarrow \begin{cases} c = a_{jj}/\sqrt{a_{jj}^2 + a_{ij}^2} \\ s = a_{ij}/\sqrt{a_{jj}^2 + a_{ij}^2} \end{cases} \tag{242}$$

(für $a_{ij} \neq 0$, ansonsten wähle $c = 1, s = 0$, d.h. $\mathbf{Q}_{ij} = \mathbf{I}$).

Algorithmus 4.2 (Transformation einer Matrix auf obere Dreiecksgestalt durch ebene Rotationen)

```
for j=1 to n-1
  for i=j+1 to m
    w = sqrt(A[j][j]*A[j][j]+A[i][j]*A[i][j])
    c = A[j][j]/w
    s = A[i][j]/w
    A=Q(i,j,c,s)^T*A
```

Auch hier stellt die Reihenfolge sicher, dass ein zu Null reduziertes Element nicht wieder später von Null verschieden wird. Es gilt:

$$\mathbf{R} = \mathbf{Q}^T(m, n) \cdot \ldots \cdot \mathbf{Q}^T(2, 1) \cdot \mathbf{A} \tag{243}$$

und wegen $\mathbf{Q}^{-1}(i, j) = \mathbf{Q}^T(i, j)$

$$\mathbf{A} = \mathbf{Q}(2, 1) \cdot \ldots \cdot \mathbf{Q}(m, n) \cdot \mathbf{R} = \mathbf{Q} \cdot \mathbf{R} \tag{244}$$

Dabei sind, falls $m > n$, die $m - n$ untersten Zeilen von $\mathbf{R}$ Null und $\mathbf{Q}$ ist eine orthogonale $m \times m$-Matrix

Die QR-Zerlegung ist immer möglich, aber nicht immer eindeutig.

In der Praxis wird die Matrix $\mathbf{Q}$ selten benötigt sondern fast immer nur die Anwendung von $\mathbf{Q}$ auf einen festen Vektor $\mathbf{b}$

$$\mathbf{Q}^T\mathbf{b} = \mathbf{Q}^T(m, n) \cdot \ldots \cdot \mathbf{Q}^T(2, 1) \cdot \mathbf{b} \tag{245}$$

dabei wird $\mathbf{Q}$ nicht ausmultipliziert sondern die Teilmatrizen auf $\mathbf{b}$ angewandt.

Alternative (*Householder-Reduktion*): wähle in der Spiegelungsmatrix $\mathbf{T} = I - 2\mathbf{u}\mathbf{u}^T/\kappa$ den Vektor $\mathbf{u}$ so, dass die ersten k Komponenten Null sind. Dann lässt die die Transformation $\mathbf{TA}$ die ersten k Zeilen von $\mathbf{A}$ unverändert. Mit $\mathbf{T}^1$ wird dann die erste Spalte von $\mathbf{A}$ zu Null gesetzt, mit $\mathbf{T}^2$ die zweite, usw., sodass insgesamt

$$\mathbf{T}^n \ldots \mathbf{T}^1\mathbf{A} = \mathbf{R} \tag{246}$$

bzw.

$$\mathbf{A} = \mathbf{T}^1 \ldots \mathbf{T}^n\mathbf{R} = \mathbf{QR} \tag{247}$$

eine QR-Zerlegung von $\mathbf{A}$ ist.

Siehe auch: Gram-Schmidt-Orthogonalisierung.

4.3.3 Ähnlichkeitstransformationen

Givens-Rotationen sind Ähnlichkeitstransformationen von $\mathbf{A}$

$$\mathbf{Q}^T(i,j)\mathbf{A}\mathbf{Q}(i,j) \tag{248}$$

mit denen eine quadratische Matrix $\mathbf{A}$ auf obere Hessenberg-Form gebracht wird. Die Ähnlichkeitstransformationen verändern dabei die Eigenwerte von $\mathbf{A}$ nicht. Dazu sind $(n-2)(n-1)/2$ ebene Rotationen nötig, wobei beidseitige Transformationen angewandt werden.

Algorithmus 4.3 (Transformation einer Matrix auf obere Hessenbergform durch ebene Rotationen)

```
for j=1 to n-2
  for i=j+2 to n
    w = sqrt(A[j][j]*A[j][j]+A[i][j]*A[i][j])
    c = A[j][j]/w
    s = A[i][j]/w
    A=Q(i,j,c,s)^T*A*Q(i,j,c,s)
```

Symmetrische Matrizen behalten dabei ihre Symmetrie und es muss nur eine Hälfte berechnet werden.

Alternative (*Householder-Transformation*): Ähnlichkeitstransformation mittels Spiegelungen über zweiseitige Householder-Reduktionen

$$\mathbf{T}^{n-2}\dots\mathbf{T}^1\mathbf{A}\mathbf{T}^1\dots\mathbf{T}^{n-2} \tag{249}$$

Die obigen Ähnlichkeitstransformationen sind der erste Schritt zur Berechnung der Eigenwerte und -vektoren einer Matrix.

4.4 Lösung linearer Gleichungssysteme

Gegeben sei nun eine quadratische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ und ein Vektor (die rechte Seite) $\mathbf{b} \in \mathbb{R}^n$. Ziel ist es nun, den Lösungsvektor $\mathbf{x} \in \mathbb{R}^n$ des linearen Gleichungssystems

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{250}$$

zu berechnen. Die formale Lösung des linearen Gleichungssystems ist

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{251}$$

wobei $\mathbf{A}^{-1} \in \mathbb{R}^{n,n}$ die Inverse von $\mathbf{A}$ ist. Bekanntlich ist

$$det(\mathbf{A}) \neq 0 \tag{252}$$

eine notwendige und hinreichende Bedingung für die Existenz und Eindeutigkeit von $\mathbf{A}^{-1}$ und damit von $\mathbf{x}$.

Anmerkungen:

- es ist selten eine geschlossene Berechnung der Determinante von $\mathbf{A}$ möglich
- günstiger und meist leichter durchführbar ist der Nachweis von

  $$\mathbf{A}\mathbf{x} = \mathbf{0} \text{ nur für } \mathbf{x} = \mathbf{0} \tag{253}$$

  (was äquivalent zu $det(\mathbf{A}) \neq 0$ ist)

Die Berechnung von $\mathbf{x}$ in $\mathbf{Ax} = \mathbf{b}$ heisst *Auflösen* des Gleichungssystems. Dies kann immer mit $O(n^3/3)$ Subtraktionen und Multiplikationen und $O(n^2/2)$ Divisionen durchgeführt werden.

Die formale Angabe der Lösung kann durch die *Cramersche Regel*

$$x_i = det(\mathbf{A}^{i,b})/det(\mathbf{A}) \text{ für } i = 1, \ldots n \tag{254}$$

erfolgen, wobei die Matrix $\mathbf{A}^{i,b}$ durch Ersetzen der i-ten Spalte durch $\mathbf{b}$ entsteht. Sie ist aber als numerisches Verfahren ungeeignet.

Grundsatz 1: Die numerische Auflösung von $\mathbf{Ax} = \mathbf{b}$ niemals mit Hilfe der Cramerschen Regel durchführen.

Grundsatz 2: Nie eine Matrix numerisch invertieren, statt dessen immer ein Gleichungssystem lösen (d.h. die Aktion von $\mathbf{A}^{-1}$ auf einen Vektor).

Beispiel:

- berechne $\mathbf{y} = \mathbf{BA}^{-1}\mathbf{Cd}$ für gegebene $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n,n},\ \mathbf{d} \in \mathbb{R}^n$
- richtig: $\mathbf{B} = \mathbf{Cd}$; Löse $\mathbf{Ax} = \mathbf{b}$; $\mathbf{y} = \mathbf{Bx}$
- falsch (und wesentlich teurer): $\mathbf{X} = \mathbf{A}^{-1}$; $\mathbf{Y} = \mathbf{BX}$; $\mathbf{Z} = \mathbf{YC}$; $\mathbf{y} = \mathbf{Zd}$

4.4.1 Kondition linearer Gleichungssysteme

Wir vergleichen

$$\mathbf{x} \text{ in } \mathbf{Ax} = \mathbf{b} \tag{255}$$

mit

$$\mathbf{x} + \delta\mathbf{x} \text{ in } (\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = (\mathbf{b} + \delta\mathbf{b}) \tag{256}$$

dabei soll $\mathbf{A}$ nichtsingulär sein und $\delta\mathbf{A}$ klein genug, sodass $\mathbf{A} + \delta\mathbf{A}$ immer noch nichtsingulär ist.

Satz 4.4: Sei $\mathbf{A} \in \mathbb{R}^{n,n}$ nichtsingulär und $\delta\mathbf{A}$ so gewählt, dass

$$\|\delta\mathbf{A}\| \leq \frac{1}{\|\mathbf{A}^{-1}\|} \tag{257}$$

dann ist $\mathbf{A} + \delta\mathbf{A}$ ebenfalls nichtsingulär.

Beweis:

$$\begin{aligned} (\mathbf{A} + \delta\mathbf{A})\mathbf{x} = 0 \quad &\Rightarrow \quad \mathbf{x} = -\mathbf{A}^{-1}\delta\mathbf{A}\mathbf{x} \\ &\Rightarrow \quad \|\mathbf{x}\| \leq \|\mathbf{A}^{-1}\|\|\delta\mathbf{A}\|\|\mathbf{x}\| \\ &\Rightarrow \quad \left(1 - \|\mathbf{A}^{-1}\|\|\delta\mathbf{A}\|\right)\|\mathbf{x}\| \leq 0 \end{aligned}$$

nur möglich für $\mathbf{x} = \mathbf{0}$, denn sonst wären hier beide Faktoren $> \mathbf{0}$. □

Sei also $\mathbf{Ax} = \mathbf{b}$ und $(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = (\mathbf{b} + \delta\mathbf{b})$, dann gilt

$$\begin{aligned} \mathbf{A}\delta\mathbf{x} &= \mathbf{b} + \delta\mathbf{b} - (\mathbf{Ax} + \delta\mathbf{Ax} + \delta\mathbf{A}\delta\mathbf{x}) && (258) \\ &= \delta\mathbf{b} - \delta\mathbf{Ax} + \delta\mathbf{A}\delta\mathbf{x} && (259) \\ \delta\mathbf{x} &= \mathbf{A}^{-1}(\delta\mathbf{b} - \delta\mathbf{Ax} + \delta\mathbf{A}\delta\mathbf{x}) && (260) \end{aligned}$$

und damit ist

$$\|\delta\mathbf{x}\| \leq \frac{\|\mathbf{A}^{-1}\|}{1 - \|\mathbf{A}^{-1}\|\|\delta\mathbf{A}\|} \cdot (\|\delta\mathbf{b}\| + \|\delta\mathbf{A}\|\|\mathbf{x}\|) \tag{261}$$

eine Schranke für die Änderung $\delta\mathbf{x}$ der Lösung zu gegebenen Änderungen $\delta\mathbf{A}$, $\delta\mathbf{b}$ der Daten.

Insbesondere gilt mit der Kondition $\kappa = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$ und den relativen Fehlerschranken

$$\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} \leq \varepsilon \text{ und } \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \leq \varepsilon \tag{262}$$

die Abschätzung

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\varepsilon\kappa}{1-\varepsilon\kappa}\left(\frac{\|\mathbf{b}\|}{\|\mathbf{A}\|\|\mathbf{x}\|}+1\right) \tag{263}$$

(beachte: $\|\mathbf{b}\| = \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$).

Die Konditionszahl der Matrix gibt somit den Verstärkungsfaktor, mit dem Änderungen in den Daten sich auf Änderungen in der Lösung auswirken. Wenn $\mathbf{A}$ und $\mathbf{b}$ mit unbekannten zufälligen Fehlern der Größenordnung $\|\mathbf{A}\|\varepsilon$ und $\|\mathbf{b}\|\varepsilon$ versehen sind, dann ist in der Lösung mit der Unsicherheit $\|\mathbf{x}\|\varepsilon\kappa/(1-\varepsilon\kappa)$ unabhängig von Rundungsfehlern bei der Auflösung des linearen Gleichungssystems zu rechnen. Nur falls $\kappa\varepsilon << 1$ gilt, ergibt es Sinn, $\mathbf{x}$ überhaupt zu berechnen.

Sei $\tilde{\mathbf{x}}$ irgendeine Näherung von $\mathbf{x}$, dann ist das *Residuum*

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}} \tag{264}$$

der Unterschied zwischen linker und rechter Seite im Gleichungssystem. In der Praxis wird oft verglichen, ob beide Seiten des Residuums bis auf Rechengenauigkeit übereinstimmen, d.h.

$$\|\mathbf{r}\| = (\|\mathbf{b}\| + \|\mathbf{A}\tilde{\mathbf{x}}\|)O(eps) \tag{265}$$

Daraus folgt aber nicht, dass

$$\|\tilde{\mathbf{x}} - \mathbf{x}\| = \|\mathbf{x}\|O(eps) \tag{266}$$

Richtig ist stattdessen:

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}} = \mathbf{A}\mathbf{x} - \mathbf{A}\tilde{\mathbf{x}} \text{ also } \mathbf{x} - \tilde{\mathbf{x}} = \mathbf{A}^{-1}\mathbf{r} \tag{267}$$

und damit

$$\|\tilde{\mathbf{x}} - \mathbf{x}\| \leq \frac{\kappa\|\mathbf{r}\|}{\|\mathbf{A}\|} \tag{268}$$

Beispiel: sei $\kappa = 10^6$, dann kann ein beliebiges falsches $\tilde{\mathbf{x}}$ beim Einsetzen in $\mathbf{A}\mathbf{x} = \mathbf{b}$ eine Übereinstimmung mit $\mathbf{b}$ auf 6 Dezimalstellen ergeben.

Mit der Umordnung

$$\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b} - \mathbf{r} \tag{269}$$

kann das Residuum trotzdem verwendet werden. Damit ist $\tilde{\mathbf{x}}$ die exakte Lösung zu $\mathbf{A}$ und $\tilde{\mathbf{b}} = \mathbf{b}-\mathbf{r}$. Somit: ist das Residuum in der Größenordnung der Unsicherheit $\delta\mathbf{b}$ der rechten Seite, dann ist $\tilde{\mathbf{x}}$ akzeptabel.

Zwei Näherungen $\mathbf{x}^{(1)}$ und $\mathbf{x}^{(2)}$ können Fehler gleicher Größenordnung und Residuen sehr ungleicher Größenordnung haben:

$$\|\mathbf{x}^{(1)} - \mathbf{x}\| \approx \|\mathbf{x}^{(2)} - \mathbf{x}\| \text{ und } \|\mathbf{A}\mathbf{x}^{(1)} - \mathbf{b}\| << \|\mathbf{A}\mathbf{x}^{(2)} - \mathbf{b}\| \tag{270}$$

wobei der Unterschied bis zu einem Faktor κ sein kann. Bei $\mathbf{x}^{(1)}$ bestehen starke Korrelationen in den Fehlern der n Komponenten, während bei $\mathbf{x}^{(2)}$ die Fehler zwar gleich groß aber völlig unkorreliert sind. In Anwendungen mit nicht exakt bekannter rechter Seite $\mathbf{b}$ ist dann $\mathbf{x}^{(1)}$ wesentlich besser als $\mathbf{x}^{(2)}$ obwohl deren absolute Fehler durchaus vergleichbar sein können.

Betonung dieses Sachverhalts weil:

- die Gauß-Elimination mit Pivot-Suche (numerische Standard-Verfahren) produziert eine Näherung $\tilde{\mathbf{x}}$ des Typs $\mathbf{x}^{(1)}$ mit $\|\mathbf{r}\| = O(eps\|\mathbf{A}\|\|\tilde{\mathbf{x}}\|)$
- bei der Cramerschen Regel wird jede Komponente für sich berechnet; die Fehler sind zufällig und unkorreliert und es wird eine Näherung $\tilde{\mathbf{x}}$ des Typs $\mathbf{x}^{(2)}$ ermittelt
- wenn die Näherung $\tilde{\mathbf{x}}$ über $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ berechnet wird, werden die Komponenten von $\mathbf{x}$ ebenfalls unabhängig voneinander berechnet; selbst bei exaktem $\mathbf{A}^{-1}$ erhält man unkorrelierte Rundungsfehler der Größenordnung $O(eps\|\mathbf{A}^{-1}\|\|\mathbf{b}\|)$

4.4.2 Gauß-Eliminationsverfahren

Prinzip der Gauß-Elmination:

- löse eine der n Gleichungen nach einer Unbekannten auf
  $\rightarrow$ merke diese Gleichung zur späteren Verwendung
  $\rightarrow$ setze sie in die übrigen $n-1$ Gleichungen ein
- dadurch entsteht ein neues Gleichungssystem mit $n-1$ Gleichungen und $n-1$ Unbekannten
- nach weiteren $n-2$ solchen Schritten erhält man ein 1×1-System, das direkt gelöst wird
- gehe alle vorher gemerkten expliziten Gleichungen in Rückwärts-Reihenfolge durch und bestimme so nacheinander alle Unbekannten

Als *natürliche Reihenfolge* bezeichnet man, wenn im j-ten Eliminationsschritt die j-te Gleichung nach der j-ten Unbekannten gelöst wird.

Im ersten Eliminationsschritt wird die Unbekannte x_1 im Schema der Koeffizienten $\mathbf{A}, \mathbf{b}$ eliminiert:

- Multiplikation der Zeile 1 mit dem Faktor $l_{ij} = a_{i1}/a_{11}$
- Subtraktion von Zeile i, so dass eine Null in der Position $i, 1$ für $i = 2, \ldots n$ entsteht
- Danach stehen die Koeffizienten des reduzierten Gleichungssystems in den Zeilen $2, \ldots n$

  for $k = 1$ to n
  $\quad r_{1k} = a_{1k}$
  $y_1 = b_1$
  for $i = 2$ to n
  $\quad l_{i1} = a_{i1}/r_{11}$
  $\quad$ for $k = 2$ to n
  $\quad\quad a'_{ik} = a_{ik} - l_{i1}r_{1k}$
  $\quad b'_i = b_i - l_{i1}y_1$

- die Rest-Matrix ist somit

$$a'_{ik} = a_{ik} - a_{i1}a_{1k}/a_{11} \text{ für } i, k \in \{2, \ldots n\} \tag{271}$$

- die gemerkte erste Gleichung ist:

$$\sum_{k=1}^{n} r_{1k}x_k = y_1 \tag{272}$$

explizit

$$x_1 = \left(y_1 - \sum_{k=2}^{n} r_{1k} x_k \right) / r_{11} \tag{273}$$

Die Umbenennung von a_{1k} in r_{1k} und von b_1 in y wird im Computerprogramm nicht gemacht, sondern dient hier nur der Logik. In der Praxis werden die Werte *am Platz* gespeichert. Zudem wird die an der Postion a_{i1} erzeugte Null nicht gespeichert, sondern stattdessen dort der Wert l_{i1} für spätere mögliche Weiterverwendung (z.B. zur iterativen Nachverbesserung).

Die übrigen Schritte funktionieren analog, sodass sich insgesamt der folgende Algorithmus ergibt:

Algorithmus 4.5 (Gauß-Elimination in natürlicher Reihenfolge)
Eingabe: Matrix `A`, rechte Seite `b`
Ausgabe: Lösung `x` von $\mathbf{Ax}$=$\mathbf{b}$

```
for j=1 to n
  for k=j to n
    R[j][k]=A[j][k]
  y[j]=b[j]
  for i=i+1 to n
    L[i][j]=A[i][j]/R[j][j]
    for k=j+1 to n
      A[i][k]=A[i][k]-L[i][j]*R[j][k]
    b[i]=b[i]-L[i][j]*y[j]
for i=n downto 1
  s=0
  for j=i+1 to n
    s=s+R[i][j]*x[j]
  x[i]=(y[i]-s)/R[i][i]
```

Vorraussetzung für das Funktionieren ist, dass die sogenannten *Pivot-Elemente* r_{jj} alle ungleich Null sind.

Beachte, dass die Einträge der Matrix a_{ik} und der rechten Seite b_i im Lauf der Eliminationsschritte immer wieder umgerechnet werden. Die Historie der Elemente in Position (i, k) ist am Ende des Algorithmus:

$$r_{ik} = a_{ik} - \sum_{j=1}^{i-1} l_{ij} r_{jk} \text{ falls } i \leq k \tag{274}$$

$$l_{ik} = \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij} r_{jk} \right) / r_{kk} \text{ falls } i > k \tag{275}$$

$$y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j \tag{276}$$

Anzahl der arithmetischen Operationen in der Gauß-Elimination: je $\frac{(n-1)n(n+1)}{3} + \frac{(n-1)n}{2}$ Subtraktionen und Multiplikationen sowie $\frac{n(n+1)}{2}$ Divisionen

Die Gauß-Elimination kann mittels elementarer Zeilen-Operationen beschrieben werden:

$$\mathbf{N}(n, n-1, -l_{n,n-1}) \ldots \mathbf{N}(2, 1, -l_{2,1}) \mathbf{A} = \mathbf{R} \tag{277}$$

$$\mathbf{N}(n, n-1, -l_{n,n-1}) \ldots \mathbf{N}(2, 1, -l_{2,1})\mathbf{b} \quad = \quad \mathbf{y} \tag{278}$$
$$\tag{279}$$

und damit da $\mathbf{N}(i, j, \alpha)^{-1} = \mathbf{N}(i, j, -\alpha)$

$$\mathbf{b} = \mathbf{N}(2, 1, l_{2,1}) \ldots \mathbf{N}(n, n-1, l_{n,n-1})\mathbf{y} \tag{280}$$

Auf diese Weise gilt:

$$\mathbf{A} = \mathbf{LR} \text{ und } \mathbf{b} = \mathbf{Ly} \tag{281}$$

mit den Dreiecksmatrizen

$$\mathbf{L} = \begin{pmatrix} 1 & & & 0 \\ l_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \ldots & l_{n,n-1} & 1 \end{pmatrix}, \ \mathbf{R} = \begin{pmatrix} r_{11} & \ldots & \ldots & r_{1n} \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ 0 & & & r_{nn} \end{pmatrix} \tag{282}$$

Elementweise lauten die Gleichungen

$$a_{ik} = \sum_{j=1}^{\min(i,k)} l_{ij} r_{jk} \quad = \quad \begin{cases} \sum_{j=1}^{i-1} l_{ij} r_{jk} + 1 \cdot r_{ik} & \text{falls } i \leq k \\ \sum_{j=1}^{k-1} l_{ij} r_{jk} + l_{ik} \cdot r_{kk} & \text{falls } i > k \end{cases} \tag{283}$$

$$b_i = \quad \sum_{j=1}^{i-1} l_{ij} y_i + 1 \cdot y_i \tag{284}$$

was genau der Gauß-Elimination entspricht (wobei dort immer nach der einen, noch nicht berechneten Größe aufgelöst ist).

Damit ist die Gauß-Elimination in natürlicher Reihenfolge identisch mit

- der Dreieckszerlegung $\mathbf{A} = \mathbf{L} \cdot \mathbf{R}$
- der Vorwärtssubstition $\mathbf{Ly} = \mathbf{b}$ (ergibt $\mathbf{y}$) und der
- der Rückwärtssubstition $\mathbf{Rx} = \mathbf{y}$ (ergibt $\mathbf{x}$)

Der Unterschied ist nur in der Reihenfolge solcher Operationen, die nichts miteinander zu tun haben:

- die Gauß-Elimination formt die $n(n+1)$ Skalarprodukte simultan
- die Dreieckzerlegung mit Vorwärts- und Rückwärtssubstitution formt sie nacheinander

Dabei werden die gleichen Zwischenresultate und die gleichen Rundungsfehler gemacht. Beide Algorithmen sind äquivalent und geben auch unter Einfluß von Rundungsfehlern identische Resultate. Für die Rundungsfehleranalyse genügt es, einen der beiden Prozesse anzusehen.

4.4.3 Cholesky-Zerlegung

Wenn die Matrix $\mathbf{A}$ symmetrisch positiv definit ist, d.h.

$$\mathbf{A} = \mathbf{A}^T \text{ und } \mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \; \forall \; \mathbf{x} \neq \mathbf{0} \tag{285}$$

gibt es substantielle Einsparmöglichkeiten.

Anwendungen, bei denen sysmmetrisch positive Matrizen vorkommen sind z.B.:

- Minimierung einer Funktion: $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + \mathbf{c}$ in $\mathbb{R}^n$
- (selbstadjungierte) partielle Differentialgleichungen in sogenannter schwacher Form:

  $$\text{finde } u \in V \text{ sodass } a(u,v) = (f,v) \tag{286}$$

  (z.B. für die Laplace-Gleichung $\Delta u = f$: $a(u,v) = \int_\Omega \nabla u \nabla v \; d\Omega$) führt nach Diskretisierung mit Finiten Elementen zu

  $$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{287}$$

  mit $\mathbf{A}$ symmetrisch positiv definit
- lineare Ausgleichsprobleme mit der Methode der kleinsten Quadrate (siehe 4.5) führen zu

  $$\mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x} = \|\mathbf{A}\mathbf{x}\|_2^2 > 0 \Leftrightarrow \mathbf{A}\mathbf{x} \neq \mathbf{0} \Leftrightarrow \mathbf{x} \neq \mathbf{0} \tag{288}$$

Für $\mathbf{A}$ symmetrisch positiv definit zeigt die Wahl von $\mathbf{x} = \mathbf{e}_1 = (1, 0, \ldots, 0)^T$ sofort dass $a_{11} > 0$. Damit ist der erste Eliminationsschritt mit a_{11} als Pivot immer durchführbar. Die Restmatrix

$$a'_{ik} = a_{ik} - a_{i1}a_{1k}/a_{kk} \text{ für } i,k = 2,\ldots,n \tag{289}$$

bleibt symmetrisch positiv definit, denn mit der Zerlegung

$$\mathbf{A} = \begin{pmatrix} a_{11} & \mathbf{s}^T \\ \mathbf{s} & \mathbf{B} \end{pmatrix}, \; \mathbf{x} = \begin{pmatrix} \eta \\ \mathbf{y} \end{pmatrix} \text{ mit } \mathbf{B} \in \mathbb{R}^{n-1,n-1}, \; \mathbf{y} \in \mathbb{R}^{n-1} \tag{290}$$

gilt

$$\mathbf{A}\mathbf{x} = \begin{pmatrix} a_{11}\eta + \mathbf{s}^T\mathbf{y} & \\ \mathbf{s}\eta & \mathbf{B}\mathbf{y} \end{pmatrix} \tag{291}$$

Wähle $\mathbf{y} \neq 0$ beliebig, zudem $\eta = -\mathbf{s}^T\mathbf{y}/a_{11}$, dann ist auch $\mathbf{x} \neq \mathbf{0}$. Die erste Komponente von $\mathbf{A}\mathbf{x}$ verschwindet, sodass

$$\begin{aligned} \mathbf{0} < \mathbf{x}^T\mathbf{A}\mathbf{x} &= \mathbf{y}^T(\mathbf{s}\eta + \mathbf{B}\mathbf{y}) && (292)\\ &= \mathbf{y}^T(\mathbf{B} - \mathbf{s}\mathbf{s}^T/a_{11})\mathbf{y} && (293)\\ &= \mathbf{y}^T\mathbf{A}'\mathbf{y} && (294) \end{aligned}$$

mit der Restmatrix $\mathbf{A}' = \mathbf{B} - \mathbf{s}\mathbf{s}^T/a_{11}$ (s.o). Im nächsten Eliminationsschritt ist also die gleiche Situation, d.h. $\mathbf{A}'$ ist symmetrisch positiv definit, usw.

Da die Symmetrie erhalten bleibt, müssen von der Restmatrix immer nur die Elemente mit $i \leq k$ (oder $i \geq k$) berechnet werden $\Rightarrow$ spart fast 50% der Arbeit. Die Inhärente Symmetrie ist auch in der Dreieckszerlegung $\mathbf{A} = \mathbf{L}\mathbf{R}$ nutzbar durch Herausziehen des Faktors $\mathbf{D} = diag(r_{11}, \ldots, r_{nn})$ aus $\mathbf{R}$, also

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^T \tag{295}$$

Die Diagonalmatrix $\mathbf{D}$ enthält also alle Pivot-Elemente, die alle positiv sind. Somit ist

$$\mathbf{D}^{1/2} = diag(\sqrt{r_{11}}, \ldots, \sqrt{r_{nn}}) \tag{296}$$

wohdefiniert. Die *Cholesky-Zerlegung* ist damit

$$\mathbf{A} = \mathbf{C}\mathbf{C}^T \tag{297}$$

mit

$$\mathbf{C} = \mathbf{L}\mathbf{D}^{1/2} \tag{298}$$

bzw. $\mathbf{A} = \hat{\mathbf{C}}^T\hat{\mathbf{C}}$ mit $\hat{\mathbf{C}} = \mathbf{D}^{-1/2}\mathbf{R}$ wobei $\hat{\mathbf{C}} = \mathbf{C}^T$.

Anmerkung: die Cholesky-Zerlegung funktioniert auch für positiv definite komplexe Matrizen $\mathbf{A} \in \mathbb{C}^{n\times n}$ mit $\mathbf{A} = \mathbf{A}^H$.

4.4.4 Pivotsuche

Falls das Pivotelement $a_{kk} = 0$ (oder $|a_{kk}|$ zu klein) muss nach einem geeignetem Pivotelement gesucht werden. Man unterscheidet

- Zeilenpivotsuche: finde $\bar{j}$ sodass $|a_{k\bar{j}}| \geq |a_{kj}|$ für alle $j > k$ und vertausche Spalten j und $\bar{k}$
- Spaltenpivotsuche: finde $\bar{i}$ sodass $|a_{\bar{i}k}| \geq |a_{ik}|$ für alle $i > k$ und vertausche Zeilen k und $\bar{j}$
- Vollständige Pivotsuche: finde $\bar{i}$ und $\bar{j}$ sodass $|a_{\bar{i}\bar{j}}| \geq |a_{ij}|$ für alle $i, j > k$ und vertausche Zeilen j und $\bar{k}$ und Spalten k und $\bar{j}$

In Matrixnotation entspricht das Vertauschen von Zeilen und Spalten der Multiplikation mit einer Permutationsmatrix $\mathbf{P}(i, j)$ von links bzw. rechts. Das Produkt dieser Permutationen kann in einer Permutationsmatrix $\mathbf{P}$ zusammengefasst werden.

In der Praxis wird meist Spaltenpivotsuche eingesetzt, da vollständige Pivotsuche zu teuer und Zeilenpivotsuche komplexer zu programmieren ist (benötigt Vertauschung der Unbekannten).

Satz 4.6: Sei $\mathbf{A} \in \mathbb{R}^{n,n}$ nichtsingulär, dann gibt es eine Permutationsmatrix $\mathbf{P}$ mit

$$\mathbf{PA} = \mathbf{LR}$$

wobei $\mathbf{L}$ eine linke untere Dreiecksmatrix mit Einsen auf der Diagonale und $\mathbf{R}$ eine rechte obere Dreiecksmatrix ist.

Beweis: Da $\mathbf{A}$ nichtsingulär ist, existiert in der ersten Spalte wenigstens ein Element ungleich Null. Nach Spaltenvertauschung mit diesem Element und dem ersten Schritt der LR-Zerlegung bleibt die Restmatrix ebenfalls nichtsingulär und es kann wieder ein Element ungleich Null gefunden werden. Die Einzelpermationen können in $\mathbf{P}$ zusammengefasst werden. □

Auch aus numerischer Sicht ist die Pivotsuche sinnvoll. Je größer $|a_{kk}|$, desto kleiner ist $|l_{ij}|$ und desto besser sind die Chancen, dass Elemente in der nächsten Restmatrix nicht zu stark anwachsen. Umgekehrt lässt ein sehr kleines Pivotelement Elemente in $\mathbf{L}$ und $\mathbf{R}$ explodieren.

Beispiel (2×2-Matrix): Sei

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix},\ \mathbf{L} = \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix} \mathbf{R} = \begin{pmatrix} a & b \\ 0 & d - bc/a \end{pmatrix}$$

dann ist

$$|\mathbf{LR}| = \begin{pmatrix} |a| & |b| \\ |c| & |d| \end{pmatrix},\ |\mathbf{L}||\mathbf{R}| = \begin{pmatrix} |a| & |b| \\ |c| & \gamma|d| \end{pmatrix},$$

wobei $\gamma = |1 - \frac{bc}{ad}| + |\frac{bc}{ad}|$. Für $|bc| \leq |ad|$ ist $\gamma \leq 3$ während für $|bc| > |ad|$ γ beliebig groß werden kann. Das richtige Kriterium für die Pivotwahl wäre also, die Gleichungen zu vertauschen, wenn $|bc| > |ad|$.

Im Gegensatz dazu ist die Suche nach dem betragsgrößten Element abhängig von der Skalierung. Durch eine Skalierung $\mathbf{A} \to \mathbf{DA}$ bei Spaltenpivotsuche bzw. $\mathbf{A} \to \mathbf{D}_1\mathbf{AD}_2$ bei vollständiger Pivotsuche könnte eine beliebige Pivotfolge erzwungen werden. Vorraussetzung für das Suchkriterium nach dem betragsgrößten Element ist also eine vernünftige Skalierung der Matrix. Eine Ideale Skalierung mit vollständiger Pivotsuche würde ergeben

$$|\mathbf{L}||\mathbf{R}| = O(|\mathbf{LR}|) \tag{299}$$

und die Gauß-Elimination wäre numerisch stabil. Aber bisher gibt es keine Methode für eine automatische und preiswerte Konstruktion einer solchen vernünftigen Skalierung. Der Anwender ist hier zuständig und muß sich darum kümmern.

4.5 Lineare Ausgleichsrechnung

4.5.1 Normalengleichungen

Wir betrachten nun überbestimmte Gleichungssysteme, bei denen die Zahl der Gleichungen größer als die Zahl der Unbekannten ist:

$$\mathbf{Ax} = \mathbf{b} \text{ mit } \mathbf{A} = \mathbb{R}^{m,n},\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{b} \in \mathbb{R}^m,\ \text{wobei } m > n \tag{300}$$

wobei Ungenauigkeiten in den Eingabedaten $\mathbf{A}, \mathbf{b}$ vorliegen können. Dadurch ergeben sich Widersprüche:

$$\nexists\ \mathbf{x} \text{ das } \mathbf{Ax} = \mathbf{b} \text{ erfüllt} \tag{301}$$

$$\nexists\ \mathbf{x} \text{ sodass } \mathbf{r}(\mathbf{x}) = \mathbf{b} - \mathbf{Ax} = 0 \tag{302}$$

Der Ausweg ist nun, ein $\hat{\mathbf{x}}$ zu suchen, welches das Residuum möglichst klein macht, zum Beispiel in der Euklidischen Norm:

$$\text{Finde } \hat{\mathbf{x}} \text{ sodass } \|\mathbf{r}(\hat{\mathbf{x}})\|_2 \leq \|\mathbf{r}(\mathbf{x})\|_2\ \forall\ \mathbf{x} \in \mathbb{R}^n \tag{303}$$

Hierbei spricht man von einem *linearen Ausgleichsproblem*, wobei die Widersprüche nach der *Methode der kleinsten Quadrate* ausgeglichen werden. Dabei wird

$$\mathbf{r}^t(\mathbf{x})\mathbf{r}(\mathbf{x}) = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{b} + \mathbf{b}^T\mathbf{b} \tag{304}$$

minimiert. Die Ableitung nach $\mathbf{x}$ ergibt $2\mathbf{A}^T\mathbf{Ax} - 2\mathbf{A}^T\mathbf{b}$ und zu Null setzen ergibt die *Normalengleichungen*

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{b} \ \Leftrightarrow\ \mathbf{A}^T\hat{\mathbf{r}} = \mathbf{0} \tag{305}$$

Sie bilden ein notwendiges und hinreichendes Kriterium für die Minimierung, denn

$$\mathbf{r}(\mathbf{x}) = \hat{\mathbf{r}} + \mathbf{A}(\hat{\mathbf{x}} - \mathbf{x}) \text{ also } \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) = \hat{\mathbf{r}}^T\hat{\mathbf{r}} + \mathbf{0} + (\hat{\mathbf{x}} - \mathbf{x})^T\mathbf{A}^T\mathbf{A}(\hat{\mathbf{x}} - \mathbf{x}) \geq \hat{\mathbf{r}}^T\hat{\mathbf{r}} \tag{306}$$

und Gleichheit gilt für

$$\|\mathbf{A}(\hat{\mathbf{x}} - \mathbf{x})\|_2 = 0 \Leftrightarrow \mathbf{A}(\hat{\mathbf{x}} - \mathbf{x}) = 0 \Leftrightarrow \mathbf{r}(\mathbf{x}) = \hat{\mathbf{r}} \tag{307}$$

Satz 4.7: Das lineare Ausgleichsproblem

$$\mathbf{r}^t(\mathbf{x})\mathbf{r}(\mathbf{x}) \to \min$$

hat immer ein eindeutiges kleinstes Residuum $\hat{\mathbf{r}}$. Der zugehörige Minimierer $\hat{\mathbf{x}}$ ist eindeutig nur wenn die Spalten von $\mathbf{A}$ linear unabhängig sind, d.h. $Rang(\mathbf{A}) = n$. Notwendig und hinreichend für einen Minimierer sind die Normalengleichungen $\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{b}$.

Beweis: siehe oben □

Aufgrund der Normalengleichungen ist das immer eindeutige Residuum $\hat{\mathbf{r}}$ senkrecht zu allen Spalten von $\mathbf{A}$ bzw. den davon aufgespannten linearen Untervektorraum.

Bild Das kleinste Residuum steht senkrecht zu den Spalten von $\mathbf{A}$

Im folgenden seien die Spalten von $\mathbf{A}$ linear unabhängig, also

$$\mathbf{Ax} = \mathbf{0} \Rightarrow \mathbf{x} = \mathbf{0} \tag{308}$$

Damit ist auch der Minimierer $\hat{\mathbf{x}}$ eindeutig und die Koeffizientenmatrix $\mathbf{A}^T\mathbf{A}$ in den Normalengleichungen ist positiv-definit, nicht singulär.

4.5.2 Norm und Kondition

Zur Norm: Die Minimierung ist auch für andere Normen als $\|\cdot\|_2$ möglich

- $\|\mathbf{r}\| = \|\mathbf{r}\|_\mathbf{1}$
- $\|\mathbf{r}\| = \|\mathbf{r}\|_\infty$
- $\|\mathbf{r}\| = \mathbf{g_1 r_1^2} + \ldots + \mathbf{g_m r_m^2}$ wobei alle $g_i > 0$
- $\|\mathbf{r}\| = \mathbf{r^T M r}$ mit $\mathbf{M}$ positiv

Die ersten beiden Fälle führen zu nicht-quadratischen Minimierungsaufgaben, die durch sogenanntes *lineares Programmieren* gelöst werden. Sie sind jedoch wesentlich teurer und in den meisten Fällen inakzeptabel durch Zufälle oder Ausreisser in den Messwerten bestimmte Lösungen. Die letzten beiden Fälle sind in der Statistik wichtig, sie sind jedoch durch geeignete Skalierung (bzw. im letzten Fall durch Cholesky-Zerlegung) auf die Normalengleichungen rückführbar.

Zur Kondition: hier ist es nötig die Änderungen $\delta\mathbf{x}$ und $\delta\mathbf{r}$ für gegebene Änderungen $\delta\mathbf{A}$ und $\delta\mathbf{b}$ abzuschätzen. Es lässt sich zeigen (ohne Beweis):

$$\|\delta\mathbf{x}\| \leq \kappa(\alpha\|\hat{\mathbf{x}}\|_2 + \beta) + \kappa^2 \frac{\alpha\|\hat{\mathbf{r}}\|_2}{\|\mathbf{A}\|_\mathbf{2}} + O(\alpha^2) \tag{309}$$

$$\|\delta\mathbf{r}\|_2 \leq \|\delta\mathbf{A}\|\|\hat{\mathbf{x}}\|_2 + \|\delta\mathbf{b}\| + \kappa\alpha\|\hat{\mathbf{r}}\|_2 + O(\alpha^2) \tag{310}$$

wobei

$$\alpha = \|\delta\mathbf{A}\|_2/\|\mathbf{A}\|_2 \tag{311}$$

$$\beta = \|\delta\mathbf{b}\|_2/\|\mathbf{b}\|_2 \tag{312}$$

$$\tag{313}$$

und κ die Kondizionszahl von $\mathbf{A}$ ist. Entscheidend ist hier der $\kappa^2\|\hat{\mathbf{r}}\|_2$-Term. Das Ausgleichsproblem ist viel empfindlicher gegen Änderungen von $\mathbf{A}$ als das lineare Gleichungssystem. Dabei kommt es auf das Residuum an, also die Konsistenz auszugleichender Gleichungen. Für eine gute Lösung ist $\|\hat{\mathbf{r}}\|_2$ klein, für eine widersprechende Lösung groß.

4.5.3 Numerische Berechnung

Am billigsten ist die direkte Lösung der Normalengleichungen.

- bilden von $\mathbf{A}^T\mathbf{A}$ mit dessen Cholesky-Zerlegung kombinieren
- bilden von $\mathbf{A}^T\mathbf{b}$ mit der Vorwärts-Substitution kombinieren

Galt früher als ungenau, da $\mathbf{A}^T\mathbf{A}$ die Kondition κ^2 besitzt, was Rundungsfehler beim Bilden und Auflösen verstärkt. Obige Abschätzung hat jedoch auch schon einen κ^2-Term, deswegen ist dieses Vorgehen ok.

Empfehlenswert ist aber die Transformation von $(\mathbf{A}, \mathbf{b})$ mit ebenen Rotationen oder Spiegelungen von links auf obere Dreiecksform.

$$\begin{aligned} \mathbf{A} &\rightarrow \mathbf{Q}^T\mathbf{A} = \mathbf{R} && (314)\\ \mathbf{b} &\rightarrow \mathbf{Q}^T\mathbf{b} = \mathbf{c} && (315)\\ \mathbf{r}(\mathbf{x}) &\rightarrow \mathbf{Q}^T\mathbf{r}(bfx) && (316) \end{aligned}$$

Wegen $\|\mathbf{r}\|_2 = \|\mathbf{Q}^T\mathbf{r}\|_2$ ist das Resultat der Minimierung im reduzierten System:

$$(\mathbf{Q}^T\mathbf{r})_i = (\mathbf{Q}^T\mathbf{b} - \mathbf{Q}^T\mathbf{A}\mathbf{x})_i = \begin{cases} c_i - \sum_{j=1}^{n} r_{ij}x_j & \text{für } i = 1, \ldots, n \\ c_i & \text{für } i = n+1, \ldots, m \end{cases} \quad (317)$$

Die ersten n Komponenten lassen sich zu Null verkleinern, die letzten $m-n$ Komponenten lassen sich gar nicht beeinflussen. Man erhält $\hat{\mathbf{x}}$ durch Rückwarts-Substitution wie bei normalen linearen Gleichungssystemen.

Das gleiche Resultat erhält man duch Einsetzen der QR-Zerlegung von $\mathbf{A}$ in die Normalengleichungen:

$$\begin{aligned} \mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\hat{\mathbf{x}} = \mathbf{R}^T\mathbf{Q}^T\mathbf{b} &\Leftrightarrow \mathbf{R}^T\mathbf{R}\hat{\mathbf{x}} = \mathbf{R}^T\mathbf{c} \Leftrightarrow && (318)\\ \bar{\mathbf{R}}^T\bar{\mathbf{R}}\hat{\mathbf{x}} = \bar{\mathbf{R}}^T\bar{\mathbf{c}} &\Leftrightarrow \bar{\mathbf{R}}\hat{\mathbf{x}} = \bar{\mathbf{c}} && (319) \end{aligned}$$

wobei $\bar{\mathbf{R}}$ gleich $\mathbf{R}$ ohne die trivialen lezten $m-n$ Zeilen ist und $\bar{\mathbf{c}}$ die ersten n Komponenten von $\mathbf{c}$ enthält. Damit hat man eine strenge Analogie zur Gauß-Elimination bis auf

- mehr Zeilen als Spalten
- ersetze elementare Zeilenoperationen inklusive Pivotsuche durch ebene Rotationen oder Spiegelungen von links

5 Nichtlineare Gleichungen

Aufgabenstellung: bestimme die Nullstellen ξ einer Funktion f

$$f(\xi) = 0 \quad (320)$$

Beispiel: Nullstellen eines Polynoms $p(x) = a_0 + a_1x + \ldots + a_nx^n$

- geschlossene Lösung sind unpraktisch (Cardano-Formeln für $n = 4$)
- geschlossene Lösung unmöglich für $n > 4$ (Abel)

- deshalb sind hier Näherungsverfahren notwendig

Im allgemeinen ist f eine mehrdimensionale Funktion $f : E \to E$, z.B. $f : \mathbb{R}^n \to \mathbb{R}^n$

$$f(\mathbf{x}) = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ f_2(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{pmatrix} \tag{321}$$

und gesucht ist ein Vektor $\xi \in E$ mit

$$f(\xi) = \mathbf{0} \tag{322}$$

5.1 Eindimensionale Verfahren

Im folgenden beschränken wir uns auf relle Funktionen einer Variablen.

5.1.1 Bisektionsverfahren

Motivation für das *Bisektionsverfahren* ist der Zwischenwertsatz:

Ist f im Intervall $[a, b]$ stetig und es gilt $f(a)f(b) < 0$, so besitzt f dort mindestens eine Nullstelle ξ mit $f(\xi) = 0$.

Anmerkungen:

- $f(a)f(b) < 0$ bedeutet dass entweder $f(a) > 0$ und $f(b) < 0$ gilt oder $f(a) < 0$ und $f(b) > 0$
- wenn ξ z.B. eine doppelte Nullstelle ist, dann muss $f(a)f(b) < 0$ nicht gelten
- wenn $f(a)f(b) > 0$ ist, dann bedeutet das nicht, dass $[a, b]$ keine Nullstelle enthält

Sei nun $f(a)f(b) < 0$ vorrausgesetzt, dann geht das Bisektionsverfahren wie folgt vor:

Algorithmus 5.1 (Bisektionsverfahren)

```
setze x0=a, x1=b
solange x0-x1<eps
  setze x2=(x0+x1)/2
  falls f(x0)f(x2)<0 ersetze x1 durch x2
  sonst ersetze x0 durch x2
```

Bild Bisektionsverfahren

Das Bisektionsverfahren endet, sobald das Intervall $[x_0, x_1]$ die Länge ε unterschreitet. Dies ist nach n Unterteilungsschritten der Fall

$$\frac{b-a}{2^n} < \varepsilon \Leftrightarrow n > \log_2 \frac{b-a}{\varepsilon} \tag{323}$$

Insgesamt ist das Bisektionsverfahren von erster Ordnung konvergent.

5.1.2 Regula falsi

Das Regula-falsi-Verfahren funktioniert ähnlich wie das Bisektionsverfahren, nur dass zwischen den beiden Punkten $f(x_0)$ und $f(x_1)$ eine Sekante gelegt wird. Der Schnittpunkt x_2 mit der x-Achse wird dann als neue Näherung gewählt.

Bild Regula falsi

Es gilt:

$$x_2 = x_1 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} \cdot f(x_1) \tag{324}$$

Eigenschaften:

- bei stetigen Funktionen befindet sich der Schnittpunkt x_2 immer im Intervall $[x_0, x_1]$
- im Gegensatz zum Bisektionsverfahren gibt es Fälle, bei denen die Intervalllängen nicht gegen Null konvergieren, dennoch kann man den letzten Schnittpunkt als Näherung der Nullstelle verwenden
- dieses Verhalten kann durch geschickte Kombination mit dem Bisektionsverfahren vermieden werden (Dekker-Brent-Verfahren)
- solange f nicht strikt konvex oder konkav ist, erhält man superlineare Konvergenz

5.1.3 Newton-Verfahren

Im Newton-Verfahren wird nun statt der Sekante die Tangente gewählt und deren Schnittpunkt mit der x-Achse als nächste Iterierte. Betrachtet man die Taylor-Entwicklung der Funktion f um die Nullstelle ξ

$$0 = f(\xi) = f(x_0) + (\xi - x_0)f'(x_0) + \frac{1}{2}(\xi - x_0)^2 f''(x_0) + \ldots + \frac{1}{k!}(\xi - x_0)^k f^{(k)}(x_0 + \theta(\xi - x_0)),\ 0 < \theta < 1 \tag{325}$$

dann gilt näherungsweise

$$0 = f(x_0) + (\xi - x_0)f'(x_0) \tag{326}$$

und damit

$$\xi = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{327}$$

Das Newton-Verfahren ist demnach

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{328}$$

Bild Newton-Verfahren

Ganz analog lassen sich so auch Verfahren höherer Ordnung konstruieren, z.B. führt

$$0 = f(x_0) + (\xi - x_0)f'(x_0) + \frac{1}{2}(\xi - x_0)^2 f''(x_0) \tag{329}$$

und damit

$$\xi = x_0 - \frac{f'(x_0) \pm \sqrt{(f'(x_0))^2 - 2f(x_0)f''(x_0)}}{f''(x_0)} \tag{330}$$

zum Newton-Raphson-Verfahren zweiten Grades

$$x_{n+1} = x_n - \frac{f'(x_n) \pm \sqrt{(f'(x_n))^2 - 2f(x_n)f''(x_n)}}{f''(x_n)} \tag{331}$$

Bild Newton-Raphson-Verfahren

Allgemein approximiert man f durch ein Polynom vom Grad k und bestimmt die nächste Iterierte als Nullstelle dieses Polynoms.

5.1.4 Sekanten-Verfahren

Beim Sekanten-Verfahren wird die Ableitung $f'(x_n)$ im Newton-Verfahren durch den Differenzenquotient

$$f'(x) = \frac{f(x) - f(x-h)}{h} \tag{332}$$

approximiert, wobei $h = x_n - x_{n-1}$ gesetzt wird. Somit erhält man

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \cdot f(x_n) \tag{333}$$

Bild Sekantenverfahren

Eigenschaften:

- das Sekanten-Verfahren benötigt keine explizite Kenntnis der Ableitung f'
- insgesamt ist die Konvergenz des Sekantenverfahrens mit der des Newton-Verfahrens vergleichbar
- bei einfachen Nullstellen konvergiert das Sekantenverfahren mit Ordnung $1,618...$ (goldener Schnitt)
- bei mehrfachen Nullstellen oder wenn $f'(\xi) \approx 0$ gilt, dann verlangsamt sich die Konvergenz, da die Iterationsvorschrift dann die Form $x_{n+1} = x_n - \frac{0}{0} f(x_n)$ hat

5.2 Konvergenz von Iterationsverfahren

5.2.1 Konvergenzordnung

Sei eine *Iteration*

$$x_{i+1} = \phi(x_i),\ i = 0, 1, 2, \ldots \tag{334}$$

gegeben und ξ ein *Fixpunkt* von ϕ. Wenn für alle $x_0 \in U(\xi)$ in einer Umgebung U um ξ gilt

$$|x_{i+1} - \xi| \leq c|x_i - \xi|^p \tag{335}$$

wobei $p \geq 1$ und $c < 1$ für $p = 1$, dann heisst p die *Ordnung* des Iterationsverfahrens.

Die Ordnung einer Iteration kann bestimmt werden, wenn ϕ in der Umbgebung U von ξ ausrechend oft differenzierbar ist. Ist nun $x_i \in U(\xi)$ und $\phi^{(k)}(\xi) = 0$ für $k = 1, 2, \ldots p-1$ aber $\phi^{(p)}(\xi) \neq 0$, dann gilt:

$$x_{i+1} = \phi(x_i) = \phi(\xi) + \frac{(x_i - \xi)^p}{p!}\phi^{(p)}(\xi) + O(|x_i - \xi|)^{p+1} \tag{336}$$

und damit

$$\lim_{i\to\infty} \frac{x_{i+1} - \xi}{(x_i - \xi)^p} = \frac{\phi^{(p)}(\xi)}{p!} \tag{337}$$

Für $p = 2, 3, \ldots$ hat man ein Verfahren p-ter Ordnung. Für $p = 1$ erhält man ein Verfahren erster Ordnung (lineare Konvergenz), wenn zudem $|\phi'(\xi)| < 1$ gilt.

Wenn ϕ von der Ordnung p ist, dann ist ϕ *lokal konvergent*, d.h. es gibt ein $U(\xi)$ sodass für alle $x_0 \in U(\xi)$ gilt:

$$\lim_{i\to\infty} x_i = \xi \tag{338}$$

Ein Verfahren heisst *global konvergent*, wenn die Wahl $U(\xi) = \mathbb{R}$ möglich ist, damit erhält man Konvergenz für eine beliebige Wahl von x_0.

Beispiel: Sei ϕ differenzierbar in $U(\xi)$ und die Iterationsvorschrift gegeben durch

$$\phi(x) = \phi(\xi) + (x - \xi)\phi'(\xi) + O(|x - \xi|^2) \tag{339}$$

dann ist

$$\lim_{i\to\infty} \frac{x_{i+1} - \xi}{x_i - \xi} = \phi'(\xi) \tag{340}$$

also erhält man für $|\phi'(\xi)| < 1$ Konvergenz

Fall (a): $0 < \phi'(\xi) < 1$

Bild monotone Konvergenz

Fall (b): $-1 < \phi'(\xi) < 0$

Bild alternierende Konvergenz

5.2.2 Konvergenz des Newton-Verfahrens

Beim Newton-Verfahren gilt

$$\phi(x) = x - \frac{f(x)}{f'(x)} \tag{341}$$

Sei nun f genügend oft differenzierbar und ξ eine einfache Nullstelle, also $f'(\xi) \neq 0$ dann gilt

$$\phi(\xi) = \xi \tag{342}$$

$$\phi'(\xi) = 1 - \frac{(f'(\xi))^2 - f(\xi)f''(\xi)}{(f'(\xi))^2} = \frac{f(\xi)f''(\xi)}{(f'(\xi))^2} = 0 \tag{343}$$

$$\phi''(\xi) = \frac{f''(\xi)}{f'(\xi)} \tag{344}$$

und damit

$$\phi(x) = \phi(\xi) + (x-\xi)\phi'(\xi) + \frac{(x-\xi)^2}{2}\phi''(\xi) + O(|x-\xi|^3) \tag{345}$$

sowie

$$\lim_{i\to\infty} \frac{x_{i+1}-\xi}{(x_i-\xi)^2} = \frac{\phi''(\xi)}{2} \tag{346}$$

Man hat also ein Verfahren der Ordnung 2 mit (mindestens) quadratischer lokaler Konvergenz.

Wenn nun ξ eine m-fache Nullstelle mit $m > 1$ ist, also $f(\xi) = f'(\xi) = \ldots = f^{(m-1)}(\xi) = 0$ und $f^m(\xi) \neq 0$, dann kann man eine faktorisierende Darstellung von f und f' der Form

$$f(x) = (x-\xi)^m g(x) \tag{347}$$

$$f'(x) = m(x-\xi)^{m-1} g(x) + (x-\xi)^m g'(x) \tag{348}$$

mit einer differenzierbaren Funktion g mit $g(\xi) \neq 0$ angeben. Damit ist dann

$$\phi(x) = x - \frac{f(x)}{f'(x)} = x - \frac{(x-\xi)g(x)}{mg(x) + (x-\xi)g'(x)} \tag{349}$$

und

$$\phi'(\xi) = 1 - \frac{1}{m} \neq 0 \tag{350}$$

Für mehrfache Nullstellen ist das Newton-Verfahren nur linear monoton konvergent, man hat keine quadratische Konvergenz mehr.

Ausblick

- Iterative Lösung linearer Gleichungssysteme
  - Jacobi-Verfahren, Gauß-Seidel-Verfahren, SOR-Verfahren
  - Gradientenverfahren, konjugierte Gradienten
  - Mehrgitterverfahren
- Eigenwertprobleme
  - Gerschgorin-Kreise
  - QR-Algorithmus, verallg. Schur-Zerlegung
  - Powermethode, inverse Iteration
  - Lanczos-Verfahren
- Lösung gewöhnlicher Differentialgleichungen
  - Euler-Verfahren, Runge-Kutta-Verfahren (explizit, implizit)
  - Adams-Bashford, Adams-Moulton-Verfahren
- Lösung partieller Differentialgleichungen
  - Finite Differenzen
  - Finite Elemente
  - Finite Volumen
- Lösung von Integralgleichungen
- Mehrdimensionale Integration
  - Monte Carlo-Verfahren
  - Quasi-Monte Carlo-Verfahren
  - Dnngitter-Verfahren
- Mehrdimensionale Interpolation
- Mehrdimensionale Approximation

Literatur

[1] P. Deuflhard, A. Hohmann. Numerische Mathematik 1: Eine algorithmisch orientierte Einführung, 4. Auflage, Gruyter, 2008.

[2] J. Douglas Faires, R. Burden. Numerische Methoden: Näherungsverfahren und ihre praktische Anwendung, Spektrum Akademischer Verlag, 2000.

[3] R. Freund, R. Hoppe. Stoer/Bulirsch: Numerische Mathematik 1, 10. Auflage, Springer, 2007.

[4] G. Hämmerlin, K.-H. Hoffmann. Numerische Mathematik, 4. Auflage, Springer, 1994.

[5] M. Knorrenschild. Numerische Mathematik: Eine beispielorientierte Einführung, 3. Auflage, Hanser Fachbuch, 2008.

[6] R. Plato. Numerische Mathematik kompakt, 3. Auflage, Vieweg+Teubner, 2006.

[7] R. Schaback, H. Werner. Numerische Mathematik, 4. Auflage, Springer, 1993.

[8] H. Schwarz, N. Köckler. Numerische Mathematik, 6. Auflage, Vieweg+Teubner, 2006.