

# Numerik

Vorlesung TU Chemnitz



Arnd Meyer , Chemnitz

Rh 41/616, Tel.: 531 22520

email: [a.meyer@mathematik.tu-chemnitz.de](mailto:a.meyer@mathematik.tu-chemnitz.de)

# Inhaltsverzeichnis

<b>1 Fehler in numerischen Berechnungen</b>	<b>5</b>
1.1 Zahlendarstellung in Rechnern - Maschinenarithmetik . . . . .	6
1.2 Fehlerfortpflanzung . . . . .	9
<b>2 Lösung linearer Gleichungssysteme</b>	<b>11</b>
2.1 Direkte Verfahren . . . . .	11
2.1.1 Gaußscher Algorithmus . . . . .	11
2.1.2 Bemerkungen zum Gaußschen Algorithmus . . . . .	12
2.1.3 <i>LU</i> -Zerlegung bei schwach besetzten Matrizen . . . . .	17
2.1.4 Cholesky- und Crout-Zerlegung . . . . .	18
2.1.5 Spezialfall Tridiagonalmatrizen . . . . .	20
2.2 Fehlerbetrachtung bei linearen Gleichungssystemen . . . . .	22
2.3 Iterationsverfahren . . . . .	25
2.3.1 Elementare Iterationsverfahren . . . . .	25
2.3.2 Einzel- und Gesamtschrittverfahren . . . . .	26
2.3.3 Unter- und Überrelaxation . . . . .	28
2.3.4 Instationäre Iterationsverfahren . . . . .	29
2.3.5 Konvergenztheorie der Iterationsverfahren . . . . .	31
<b>3 Interpolation/Approximation von reellen Funktionen</b>	<b>37</b>
3.1 Grundaufgabe . . . . .	37
3.2 Grundlegende Theorie der Interpolationsaufgabe . . . . .	38
3.3 Interpolation mit Polynomen . . . . .	38
3.3.1 Lagrange-Interpolation . . . . .	39
3.3.2 Newton-Interpolation . . . . .	39
3.3.3 Interpolationsfehler/Konvergenz . . . . .	40
3.3.4 Hermite Interpolation . . . . .	42
3.4 Spline-Interpolation . . . . .	42
3.4.1 Kubische Splines . . . . .	43
3.4.2 B-Splines ** . . . . .	45
3.5 Alternative Interpolation . . . . .	46
3.6 Approximation . . . . .	47
3.6.1 Approximation im Mittel . . . . .	48

3.6.2	Gleichmässige oder Tschebyschew-Approximation . . . . .	50
<b>4</b>	<b>Nichtlineare Gleichungen</b>	<b>51</b>
4.1	Allgemeine Aufgabenstellung . . . . .	51
4.2	Nullstellenbestimmung mit Interpolationsideen . . . . .	51
4.3	Das Newton-Verfahren für $n > 1$ . . . . .	53
4.4	Fixpunktiteration . . . . .	54
4.5	Fixpunktiteration für $n > 1$ . . . . .	56
4.6	Besonderheiten bei Polynomgleichungen . . . . .	57
<b>5</b>	<b>Numerische Integration</b>	<b>62</b>
5.1	Vorbemerkungen . . . . .	62
5.2	Newton-Cotes-Quadratformeln . . . . .	63
5.3	Zusammengesetzte Newton-Cotes-Formeln . . . . .	66
5.4	ROMBERG-Integration . . . . .	67
5.5	Gauß-Integration . . . . .	68
5.6	Gauß-Integration in Mehrdimensionalen . . . . .	73
<b>6</b>	<b>Numerische Differentiation</b>	<b>76</b>
<b>7</b>	<b>Anfangswertaufgaben für gewöhnliche Differentialgleichungen</b>	<b>78</b>
7.1	Einfachste Grundideen / einfachste Verfahren . . . . .	79
7.2	Konsistenz . . . . .	81
7.3	Allgemeine Form: explizite Runge -Kutta - Verfahren . . . . .	83
7.4	Implizite Runge-Kutta-Formeln . . . . .	85
7.5	Stabilität - Vergleich explizit-implizit . . . . .	87
7.6	Steife Differentialgleichungen . . . . .	90
<b>8</b>	<b>Randwertprobleme - (RWP)</b>	<b>92</b>
8.1	Vorbemerkungen . . . . .	92
8.2	Differenzenverfahren . . . . .	93
8.3	Finite Elemente Methode - FEM . . . . .	95
<b>9</b>	<b>Eigenwertprobleme bei Matrizen</b>	<b>99</b>
9.1	Grundlagen der linearen Algebra . . . . .	99
9.2	Ungeeignete Verfahrensideen . . . . .	100
9.3	Iterative Berechnung von Eigenwerten bei symm. Tridiagonalmatrizen . . . . .	101
9.4	Berechnung von Eigenwerten und Eigenvektoren durch Transformation . . . . .	103
9.4.1	Spiegelungsmatrizen . . . . .	106
9.4.2	Nutzung der Spiegelungen zur QR- Zerlegung . . . . .	107
9.4.3	Transformation auf Hessenberg Gestalt . . . . .	108
9.4.4	Der Doppelschritt QR nach FRANCIS . . . . .	109

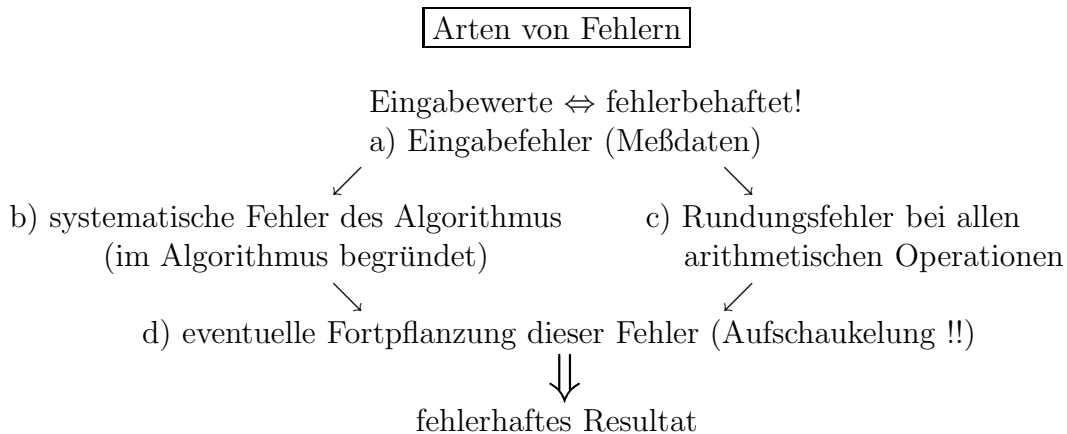
9.5	Vektoriteration . . . . .	110
9.6	Der Lanczos Algorithmus für große Eigenwertprobleme . . . . .	112

# Kapitel 1

## Fehler in numerischen Berechnungen

### Aufgabe der numerischen Mathematik:

Algorithmen anzugeben, mit denen effektiv und robust Berechnungen ausgeführt werden können. Fehler begrenzen die Resultatgäufigkeit und sind in der Regel unvermeidbar.



- a) Messfehler, Rundung auf Maschinengenauigkeit
- b) – Abbruch unendlicher Reihen
  - Annäherung komplizierter Funktionen durch einfachere
  - Diskretisierung
- c) typische Maschinenarithmetik
- d) Problem der aufeinanderfolgenden Schritte besonders zu untersuchen

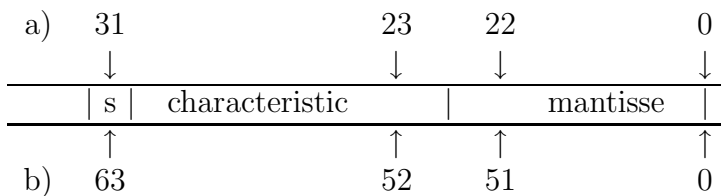
## 1.1 Zahlendarstellung in Rechnern - Maschinearithmetik

früher:  $\pm$  Mantisse  $\cdot \beta^{Exp}$  mit  $\beta = 2, \beta = 16$  (oder selten  $\beta = 10$  Robotron R300)

heute : moderne Gleitkomma-Koprozessoren- FPU benutzen IEEE- Datenformat  
Institute of **E**lectrical and **E**lectronics **E**ngineers

a) single precision: 4 Byte (1 Wort)

b) double precision: 8 Byte (1 Doppelwort)



$$0 \leq \text{mantisse} \leq .111\dots1_2$$

$$\text{Zahl} = (-1)^s \cdot 2^{Exp} \cdot \underbrace{(1.0 + \text{Mantisse})}_{1 \leq \text{Anteil} < 2}$$

Binärdarstellung solcher Zahlen fängt immer mit Bit 1 an  $\rightarrow$  keine Speicherung  
Exponent:

a) 8 Bit = 1 Byte Characteristic, schreiben hexadezimal 2 Hexadezimalziffern  
 $0\dots9ABCDEF$ ,  $F_H = 1111_2 = 15_{10}$

$$\text{Char} = 00_H \dots FF_H$$

$$\text{Sonderfall: Char} = 00_H \text{ und Char} = FF_H$$

$$\text{ansonsten: Char} = \text{Exp} - 7F_H$$

$$\text{Exp} = \text{Char} - 127_{10}$$

$$\text{Exp} \in [-126, 254 - 127 = 127] \iff \text{Char} \in [01_H, FE_H]$$

$$\text{Char} = 00_H \text{ und Mantisse} = 0\dots0_2 \implies \text{Zahl „0“}$$

$$\text{Char} = FF_H = 255 \implies \text{Mantisse} = 0\dots0_2 \pm \text{INF inity}$$

$$\implies \text{Mantisse} \neq 0\dots0_2 \text{ NaN (not a number)}$$

z.B.  $\frac{0}{0}, \frac{\infty}{\infty}, 0 \times \infty$  ergibt NaN(Not a Number)

- b) analog bei 11 Bit für Char (52 Bit Mantisse)  
 also von  $000_H$  bis  $7FF_H$  ( $0_{10} \dots 2047_{10}$ )  
 hier: Char  $=000_H \longrightarrow$  Zahl 0 (Mantisse = 0)  
 Char  $=7FF_H \longrightarrow \pm\infty$  bzw. *NaN*  
 sonst: Exp = Char  $-3FF_H$   
 $=$  Char  $-1023$   
 Exp  $\in [-1022, 2047 - 1023] \Leftrightarrow Char \in [001_H, 3FE_H]$

**Folgerung 1.1**

$fl(z)$  Annäherung der reellen Zahl  $z$  durch maschinen-interne Darstellung im IEEE-Datenformat  $\implies$

1.  $|z| < 2^{-126}$  (bzw.  $2^{-1022}$ )  $\implies fl(z) = 0$
2.  $|z| > 2^{128}$  (bzw.  $2^{1024}$ )  $\implies z$  nicht darstellbar
3. sonst  $fl(z) = z(1 + \varepsilon)$ ,  $|\varepsilon| < \varepsilon_{mach}$  „Maschinen-Epsilon“.

**Definition 1.2**

$\varepsilon_{mach}$  kleinste positive Zahl  $\varepsilon$  mit

$$fl(1 \oplus \varepsilon) > 1$$

↓

in der Maschinearithmetik ausgeführt

$$1 = 2^0 \cdot (1 + \text{Mantisse})$$

$$\text{Mantisse} = 0 \dots 01$$

a) 23 Bit

b) 52 Bit

$$\implies \varepsilon_{mach} = \underbrace{2^0(0. + [0 \dots 01]_2)}_{\substack{\text{a) } 2^{-23} \approx \frac{1}{8}10^{-6} \\ \text{b) } 2^{-52} \approx \frac{1}{4}10^{-15}}}$$

**Folgerung 1.3**

Bei jeder einzelnen arithmetischen Operation  $\oplus$  gilt

$$\underbrace{fl(a \oplus b)}_{\text{Maschinenergebnis}} = \underbrace{(a \oplus b)}_{\text{exaktes Ergebnis}} (1 + \varepsilon) \quad \text{mit} \quad |\varepsilon| \leq \varepsilon_{mach}$$

Maschinenergebnis      exaktes Ergebnis

Besonderheit: bei der Subtraktion

Stellenauslöschung

**Beispiel 1.4**

zur Vereinfachung betrachten wir Dezimalarithmetik mit 4 Mantissenstellen

a) 2 Zahlen unterschiedlicher Grösse

$$\begin{array}{r} .1000 \cdot 10^2 \\ - .1000 \cdot 10^{-3} \\ \hline .9999 \cdot 10^{+1} \end{array}$$

$$\begin{array}{r} .1000 \cdot 10^{+2} \\ - .000001 \cdot 10^{+2} \\ \hline .1000 \cdot 10^{+2} \end{array}$$

Fehler analog Folgerung 2

b) 2 Zahlen fast gleicher Größen

$$\begin{array}{r} .1234 \cdot 10^5 \\ - .1235 \cdot 10^5 \\ \hline -.0001 \cdot 10^5 = -.1000 \cdot 10^2 \end{array}$$

exaktes Ergebnis für diese eine Operation

aber: waren Summanden fehlerbehaftet durch vorherige Operation

→ z.B. nur Ziffern .123 verlässlich, in beiden Operanden

⇒ so erhalten wir ein völlig unsinniges Ergebnis.

**Beispiel 1.5**

Berechnung der beiden Wurzeln der quadratischen Gleichung

$$x^2 + px + q = 0$$

$$x_{1/2} = -\frac{p}{2} \Rightarrow \sqrt{\left(\frac{p}{2}\right)^2 - q}$$

↓

bei einem der VZ eventuelle Auslöschung

aber es gilt stets

1. stabil betragsgrösste Wurzel

$$x_1 = -\left(\frac{p}{2} + (\text{sign } p)\sqrt{\left(\frac{p}{2}\right)^2 - q}\right)$$

2. berechne betragskleinere Wurzel nach Vietaschen Wurzelsatz:

$$x_2 = \frac{q}{x_1} \quad \longrightarrow \quad \text{garantiert } fl(x_2) = x_2(1 + 2\varepsilon)$$

## 1.2 Fehlerfortpflanzung

### Beispiel 1.6

Seien  $I_0 \dots I_{20}$  zu berechnen mit

$$I_n = \frac{1}{e} \int_0^1 x^n e^x dx$$

#### 1. Algorithmus (instabil):

$I_0 = 1 - \frac{1}{e}$  mit  $\varepsilon_{mach}$  berechnet, mit partieller Integration folgt:

$$I_n = \frac{1}{e} x^n e^x \Big|_0^1 - \frac{1}{e} n \int_0^1 x^{n-1} e^x dx$$

$(u = x^n \quad v = e^x)$

$$\boxed{I_n = 1 - n I_{n-1}}$$

Rechnung ergibt:

Hauptgrund:

$$\begin{aligned} \tilde{I}_0 &= fl(I_0) = I_0 + \epsilon_0 \\ \tilde{I}_1 &= 1 - 1\tilde{I}_0 + \epsilon_1 = I_1 + \epsilon_0 + \epsilon_1 \\ \tilde{I}_2 &= 1 - 2\tilde{I}_1 = I_2 + 2\epsilon_0 + \dots \\ \tilde{I}_3 &= I_3 - 3 \cdot 2\epsilon_0 + \dots \\ &\vdots \\ \tilde{I}_n &= I_n \pm n! \epsilon_0 + \dots \end{aligned}$$

#### 2. Algorithmus:

$$I_{n-1} = \frac{1}{n}(1 - I_n)$$

Fehlerbetrachtung analog zu Algorithmus 1 aber umgekehrt

→ frühere Fehler werden stark gedämpft (wie  $\frac{1}{n!}$ ) ⇒ Startwert

z.B.  $\tilde{I}_{100}$  völlig beliebig → Rückrechnung bis  $I_{20}$  ergibt fast fehlerfreien Wert.

#### Theorie:

- absoluter Fehler der Grösse  $x$ :  $\Delta x$

$$\tilde{x} = x + \Delta x \quad (\text{genäherter Wert} = \text{exakter Wert} + \text{absoluten Fehler})$$

- relativer Fehler der Grösse  $x$ :  $\delta x$

$$\tilde{x} = x(1 + \delta x)$$

$$\delta x = \frac{\Delta x}{x}$$

- Vorwärtsanalyse des Fehlerverhaltens eines Rechenprozesses:

Eingabegrößen  $x_1, \dots, x_n$   
 mit Eingangsfehler  $\Delta x_1 \dots \Delta x_n$   
 bzw.  $\delta x_1 \dots \delta x_n$

Rechenvorschrift:

$$y = \varphi(x_1, \dots, x_n)$$

- a) betrachte Fortpflanzung der Eingabefehler
- b) betrachte zusätzliche Rundungsfehler in der Vorschrift  $\varphi$
- a) häufig reicht Betrachtung der 1. Ordnung in einer Taylorentwicklung:

$$\begin{aligned} \Delta y = \tilde{y} - y &= \varphi(x_1 + \Delta x_1, \dots, x_n + \Delta x_n) - \varphi(x_1, \dots, x_n) \\ &= \sum_{i=1}^n \frac{\partial \varphi(x_1, \dots, x_n)}{\partial x_i} \cdot \Delta x_i + \mathcal{O}(|\Delta x_i|^2) \end{aligned}$$

also Größen  $(\frac{\partial \varphi}{\partial x_i})$  Indikatoren für starke oder schwache Empfindlichkeit gegen Eingangsfehler

- Rückwärtsanalyse

Ergebnis: Resultat  $\tilde{y}$  ist für exakte Eingangsgrößen  $\tilde{x}_i + \varepsilon_i$  ( $i = 1, \dots, n$ ).  $\varepsilon_i$  nicht exakt bekannt, aber Abschätzungen der Art

$$|\varepsilon_i| < \varepsilon_{mach} \cdot f(n) \cdot \|x\|$$

sind möglich.

### Beispiel 1.7

Grundalgorithmen der linearen Algebra die auf Skalarprodukten basieren.

# Kapitel 2

## Lösung linearer Gleichungssysteme

### Aufgabenstellung:

geg.:  $A$  ( $n \times n$ )-Matrix

$$A = (a_{ij})_{i,j=1}^n, \quad b = (b_i)_{i=1}^n$$

ges.: Lösung von  $Ax = b$  Lösungsvektor  $x = (x_i)_{i=1}^n$

2 Möglichkeiten zur Lösung  $x$  eines Gleichungssystems zu gelangen

- direkte Verfahren  $\longrightarrow$  Gaußscher Algorithmus + verwandte Methoden
- Iterationsverfahren

## 2.1 Direkte Verfahren

### 2.1.1 Gaußscher Algorithmus

Grundidee: subtrahieren  $\frac{a_{i1}}{a_{11}}$ -fache der 1. Zeile von allen anderen

$$A^{(1)} = A \longrightarrow A^{(2)} = \begin{bmatrix} a_{11}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

und dieser Prozeß wird für die  $(n-1) \times (n-1)$ -Restmatrix fortgesetzt (rechte Seite kann sofort mit transformiert werden)

$$(A|b) = (A^{(1)}|b^{(1)}) \longrightarrow (A^{(2)}|b^{(2)}) \longrightarrow \dots \longrightarrow (A^{(n)}|b^{(n)})$$

wobei  $A^{(n)} = U = \begin{bmatrix} u_{11} & \cdots & u_{1j} \\ & \ddots & \vdots \\ & & u_{nn} \end{bmatrix}$  obere Dreiecksmatrix

Durch Zeilenlinearkombinationen der Ausgangsgleichung  $Ax = b$  hat sich  $x$  nicht geändert, also  $Ux = b^{(n)}$  leicht auflösbar:

Rückwärtselimination

$$x_n = \frac{b_n^{(n)}}{u_{nn}}$$

$$x_k = \left( b_k^{(n)} - \sum_{j>k} u_{kj} x_j \right) \frac{1}{u_{kk}}, \quad k = n-1, \dots, 1$$

## 2.1.2 Bemerkungen zum Gaußschen Algorithmus

### Bemerkung 2.1

$a_{kk}^{(k)} \neq 0$  notwendig wegen Division aber numerische Forderung:

„ $|a_{kk}^{(k)}|$  nicht klein“ ist Voraussetzung um ein stabiles Verfahren zu erhalten

$\implies$  **Rekursion:**

$$\text{Zeile } j := \text{Zeile } j - \left( \frac{a_{jk}^{(k)}}{a_{kk}^{(k)}} \right) * \text{Zeile } k$$

$$\text{nach 2.1} \longrightarrow \left( \frac{a_{jk}^{(k)}}{a_{kk}^{(k)}} \right) \text{ möglichst klein}$$

deshalb:

### Pivotisierung

d.h. suche ein genügend großes Element  $a_{\mu\nu}^{(k)}$  in der jeweiligen Restmatrix und vertausche Zeilen (Spalten)  $\mu \longleftrightarrow k$  ( $\nu \longleftrightarrow k$ ) in der Matrix  $(A^{(k)} | b^{(k)})$ ,

so daß  $a_{\mu\nu}^{(k)}$  zu  $a_{kk}^{(k)}$  in der vertauschten Matrix wird

(die Spaltenvertauschung zieht eine Umnummerierung der  $x_i$  nach sich)

deshalb:

**Varianten:** 1.) „Spaltenpivotsuche“

$$|a_{\mu k}^{(k)}| = \max_{i \geq k} |a_{ik}^{(k)}| \quad \text{gesucht}$$

$\longrightarrow$  dann Zeilenvertauschung  $\mu \longleftrightarrow k$

2.) „Zeilenpivotsuche“

$$|a_{k\nu}^{(k)}| = \max_{i \geq k} |a_{ki}^{(k)}| \quad \text{gesucht}$$

$\longrightarrow$  dann Spaltenvertauschung  $\nu \longleftrightarrow k$  und merken

3.) „vollständige Pivotsuche“ ( Mehraufwand !!)

$$|a_{\mu\nu}^{(k)}| = \max_{i,j \geq k} |a_{ij}^{(k)}| \quad \text{gesucht}$$

**Algorithmus 2.2** Gauß Algorithmus

Das Verfahren kann mathematisch exakt durch Matrixoperationen beschrieben werden. Wir betrachten der Einfachheit halber nur Spaltenpivotisierung

**1. Schritt**

Pivotisierung ergibt Index  $\mu_1 > 1$

$$P_1 := \left[ \begin{array}{c|ccc|c} 0 & & & 1 & \\ \hline & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ \hline 1 & & & 0 & \\ \hline & & & & 1 \end{array} \right] \begin{array}{l} \leftarrow 1 \\ \\ \\ \\ \leftarrow \mu_1 \end{array}$$

$$\begin{array}{ccc} \uparrow & & \uparrow \\ 1 & & \mu_1 \end{array}$$

$P_1$  ist Permutationsmatrix, die 1. und  $\mu_1$ -te Zeile vertauscht (bei  $\mu_1 > 1$ )

$$\implies (P_1 A^{(1)} | P_1 b^{(1)}) = (\tilde{A}^{(1)} | \tilde{b}^{(1)})$$

hat „Pivotzeile“ an Position 1,  $\tilde{a}_{11}^{(1)}$  ist betragsgrösstes Element

Subtraktion des  $\frac{\tilde{a}_{i1}^{(1)}}{\tilde{a}_{11}^{(1)}}$ -fachen der 1-ten Zeile von der i-ten ergibt

$$\tilde{L}_1^{-1} (\tilde{A}^{(1)} | \tilde{b}^{(1)}) = (A^{(2)} | b^{(2)})$$

$$\tilde{L}_1^{-1} = \left[ \begin{array}{cccc} 1 & & & \\ -\tilde{l}_{21} & \ddots & & \\ \vdots & & \ddots & \\ -\tilde{l}_{n1} & & & 1 \end{array} \right] \text{„Eliminationsmatrix“, } \tilde{l}_{i1} = \frac{\tilde{a}_{i1}}{\tilde{a}_{11}}$$

**2. Schritt**

Pivotisierung

$$P_2 := \left[ \begin{array}{c|ccc|c} 1 & & & & \\ \hline & 0 & & 1 & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ \hline & 1 & & 0 & \\ \hline & & & & 1 \end{array} \right] \begin{array}{l} \\ \leftarrow 2 \\ \\ \\ \\ \leftarrow \mu_k \end{array}$$

$$\begin{array}{ccc} \uparrow & & \uparrow \\ 2 & & \mu_k \end{array}$$

$$\tilde{L}_2^{-1} P_2(A^{(2)} | b^{(2)}) = (A^{(3)} | b^{(3)})$$

usw. schliesslich:

$$\tilde{L}_{n-1}^{-1} P_{n-1} \tilde{L}_{n-2}^{-1} P_{n-2} \cdot \dots \cdot \tilde{L}_1^{-1} P_1(A | b) = (U | b^{(n)})$$

( $P_k$  Permutationsmatrix, die k-te Zeile mit der  $\mu_k$ -ten Zeile vertauscht wobei ( $\mu_k \geq k$ ))

### Bemerkung 2.3

$$\tilde{L}_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \tilde{l}_{k+1k} & \ddots & \\ & & \vdots & & \ddots \\ & & \tilde{l}_{nk} & & 1 \end{bmatrix}$$

### Bemerkung 2.4

Um Speicherplatz zu sparen benutzt man eine geschickte Realisierung die den frei werdenden Speicherplatz ausnutzt.

#### Schritt 1:

$P_1(A|b)$  wird ausgeführt (d.h. Zeilen 1 und  $\mu_1$  vertauscht) jetzt: Multiplikation von  $\tilde{L}_1^{-1}(\tilde{A}^{(1)} | \tilde{b}^{(1)})$  ab Zeile/Spalte 2 wird ausgeführt, d.h.:

$$a_{ij}^{(2)} := \tilde{a}_{ij}^{(1)} - \tilde{l}_{i1} \tilde{a}_{1j}^{(1)} \quad \forall i > 1 \quad \forall j > 1$$

in der 1. Spalte (wo theoretisch Nullen entstehen)

speichert man  $\tilde{l}_{21} \dots \tilde{l}_{n1}$

#### Schritt 2:

neue Pivotisierung in Spalte 2 (ab  $a_{22}^{(2)}$ ) und  $P_2(A^{(2)}:b^{(2)})$  wird mit **gesamtem gespeichertem Feld** ausgeführt! (also auch  $\tilde{l}_{i1}$  mit vertauscht) danach wieder Eliminationsschritt und abspeichern von  $\tilde{l}_{i2}$  in 2. Spalte usw.

### Formelsatz

for  $k = 1, 2, \dots, n-1$  do

- Spaltenpivotisierung in k-ter Spalte ab  $a_{kk}^{(k)} \rightarrow \mu_k$
- Vertausche k-te und  $\mu_k$ -te Zeile (gesamte Zeile)

- for  $i = k + 1, \dots, n$  do

$$\tilde{l}_{ik} = \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}} \quad (\text{Speichern auf (i,k)-Position})$$

Subtrahiere das  $\tilde{l}_{ik}$ -fache der  $k$ -ten von  $i$ -ten Zeile:

$$\begin{aligned} a_{ij}^{(k+1)} &:= \tilde{a}_{ij}^{(k)} - \tilde{l}_{ik} \tilde{a}_{kj}^{(k)} \quad \forall j = k + 1, \dots, n \\ b_i^{(k+1)} &:= \tilde{b}_i^{(k)} - \tilde{l}_{ik} \tilde{b}_k^{(k)} \end{aligned}$$

(mit  $\tilde{l}_{ik}$  geschrieben!)

k-ter Schritt:

Matrix der Form

$$(T^{(k)} | b^{(k)}) = \left[ \begin{array}{cccccc|c} u_{11} & \cdots & \cdots & \cdots & \cdots & \cdots & u_{1n} & \tilde{b}_1^{(k)} \\ \lambda_{21} & u_{22} & & & & & \vdots & \vdots \\ \vdots & \lambda_{32} & \ddots & & & & \vdots & \vdots \\ \vdots & \vdots & \ddots & u_{k-1k} & \cdots & \cdots & u_{k-1n} & \tilde{b}_{k-1}^{(k)} \\ \vdots & \vdots & & \lambda_{k-1k-1} & \tilde{a}_{kk}^{(k)} & \cdots & \tilde{a}_{kn}^{(k)} & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \lambda_{n1} & \lambda_{n2} & \cdots & \lambda_{nk-1} & \tilde{a}_{nk}^{(k)} & \cdots & \tilde{a}_{nn}^{(k)} & \tilde{b}_n^{(k)} \end{array} \right]$$

( $\lambda_{ij}$  vertauschte  $\tilde{l}_{ij}$ )

- jetzt Pivotsuche aus  $\tilde{a}_{kk}^{(k)}$  bis  $\tilde{a}_{nk}^{(k)}$   $\longrightarrow$  Zeile  $\mu_k$
- vertauschen gesamte Zeile  $k$  und  $\mu_k$
- speichern neue  $\lambda_{ik}$  unterhalb von  $\tilde{a}_{kk}^{(k)}$

$$\lambda_{ik} := \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}}$$

- Subtraktion des  $\lambda_{ik}$ -fachen der Zeile  $(\tilde{a}_{k+1}^{(k)} \dots \tilde{a}_{kn}^{(k)} | \tilde{b}_k^{(k)})$  von allen anderen, ergibt:  $(T^{(k+1)} | b^{(k+1)})$ .

Dies führt auf ein allgemeine Formel:

$$a_{ij}^{(k+1)} := \tilde{a}_{ij}^{(k)} - \frac{\tilde{a}_{kj}^{(k)} \tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}}.$$

Am Ende ist  $(T^{(n)} | b^{(n)})$  auf dem Matrixfeld gespeichert, mit denen folgendes gilt:

$$T^{(n)} = \begin{bmatrix} u_{11} & \cdots & u_{1j} \\ & \ddots & \vdots \\ l_{ij} & & u_{nn} \end{bmatrix}$$

### Definition 2.5

$$L = \begin{bmatrix} 1 & & & \\ l_{21} & \ddots & & 0 \\ \vdots & & \ddots & \\ l_{n1} & \cdots & l_{nn-1} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & & u_{1j} \\ & \ddots & \\ 0 & & u_{nn} \end{bmatrix}$$

### Bemerkung 2.6

Definition 2.5  $\implies LU = PA$  wobei  $P = P_{n-1} \dots P_1$  Produkt der benutzten Permutationsmatrizen ist. Deshalb ist Gaußscher Algorithmus auch eine  $LU$ -Zerlegung ( $LR$ -Zerlegung) der Matrix  $PA$  (bei Zeilenpivotsuche oder vollständige Pivotsuche gilt  $LU = PAQ$  mit  $Q$  dem Produkt von Spaltenvertausch.) Besonders wichtig ist diese Vorgehensweise, wenn viele (zeitlich nacheinander auftretende) rechte Seiten vorhanden sind:

- 1.)  $LU = PA$  LU-Zerlegung
- 2.) für jede gegebene rechte Seite lösen wir

$$\begin{aligned} Ax &= b \\ \iff PAx &= Pb \\ LUx &= Pb \end{aligned}$$

mit Hilfe von:

- 2.1.)  $\underbrace{Ly = Pb = \tilde{b}}_{\text{Vorwärtselimination}}$  ergibt  $y$

$$\begin{aligned} y_1 &= \tilde{b}_1 \\ y_k &= \tilde{b}_k - \sum_{i < k} l_{ki} y_i \quad k = 2, \dots, n \end{aligned}$$



### 2.1.4 Cholesky- und Crout-Zerlegung

geg.:  $A = A^T$  positiv definite Matrix d.h.

- a)  $x^T Ax > 0 \quad \forall x \neq 0, x \in \mathbb{R}^n$   
 $\iff$  b) alle Eigenwerte positiv  
 $\iff$  c) alle Hauptminoren positiv dh.  $\det(a_{ij})_{i,j=1}^k > 0 \quad k = 1, \dots, n$

#### Satz 2.9

$\exists$  genau eine obere Dreiecksmatrix  $R$  mit  $A = R^T R$ .

**Beweis:** konstruktiv: Cholesky-Zerlegung betrachte

$$a_{jk} = e_j^T A e_k \quad \text{mit } e_j = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

$$= e_j^T R^T R e_k = (R e_j)^T (R e_k)$$

Skalarprodukt von  $k$ -ter und  $j$ -ter Spalte von  $R$

$$a_{jk} = \sum_{i=1}^{\min(j,k)} r_{ik} r_{ij}$$

Ordnen Elemente  $a_{jk}$  spaltenweise ( $j < k$ ) und betrachten jeweils obige Formel  
 $\longrightarrow$  definiert genau ein Element von  $R$

$$a_{11} = r_{11}^2 \quad r_{11} = \sqrt{a_{11}}$$

usw.

$$\left. \begin{aligned} r_{jk} &:= \frac{1}{r_{jj}} (a_{jk} - \sum_{i < j} r_{ik} r_{ij}) \quad \forall j < k \\ r_{kk} &:= (a_{kk} - \sum_{i < k} r_{ik}^2)^{1/2} \end{aligned} \right\} k = 1, \dots, n$$

□

#### Bemerkung 2.10

- alle Wurzeln sind definiert, da  $A$  positiv definit (induktiv zeigen)

- Spaltenskalarprodukte der Elemente von  $R \longrightarrow$  bei schwach besetzten Matrizen folgende Speicherung günstig: obere Hälfte von  $A \longleftrightarrow R$
- Anzahl notwendiger Operationen:  $n^3/6$  (halber Aufwand gegenüber Gauß)
- schwach besetzt, Bandbreite  $m \longrightarrow \approx \frac{nm^2}{2}$
- auch zeilenweise Berechnung der  $r_{kj}$  möglich

**Definition 2.11**

Eine Zerlegung der Form  $A = \tilde{R}^T D \tilde{R}$  mit  $D = \text{diag}(d_{11} \dots d_{nn})$

$$\tilde{R} = \begin{bmatrix} 1 & & \tilde{r}_{1j} \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$$

nennen wir die Crout Zerlegung .

Offenbar gilt:

$$R = D^{\frac{1}{2}} \tilde{R} \longrightarrow R^T R = \tilde{R}^T D^{\frac{1}{2}} D^{\frac{1}{2}} \tilde{R}$$

außerdem wenn wir  $L = \tilde{R}^T, U = D \tilde{R}$  setzen  $\implies A = LU$  (von  $LU$ -Zerlegung).

Die Realisierung erfolgt analog zu Cholesky: (Beachtung von  $u_{kj} = d_k \cdot \tilde{r}_{kj}$ )

$$\left. \begin{aligned} u_{jk} &:= a_{jk} - \sum_{i < j} \tilde{r}_{ij} u_{ik} \quad \forall j = 1, \dots, k-1 \\ \tilde{r}_{jk} &:= \frac{u_{jk}}{d_{jj}} \\ d_{kk} &:= a_{kk} - \sum_{i < k} \tilde{r}_{ik} u_{ik} \end{aligned} \right\} k = 1 \dots n$$

**Bemerkung 2.12**

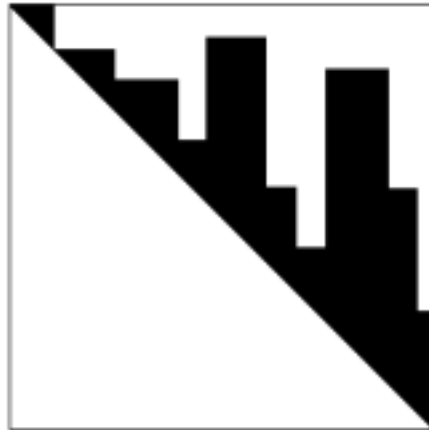
$u_{jk}$  für Spalte  $k$  nur auf einem Hilfsvektor zu speichern (nach Berechnung von  $d_{kk}$  unnötig.)

**Bemerkung 2.13**

Auf der ehemaligen Hauptdiagonale (Elemente  $a_{kk}$ ) wird am besten ( $1/d_{kk}$ ) gespeichert (da nur Multiplikation mit  $1/d_{kk}$  notwendig)  
Gleichungssystem:  $Ax = b : \tilde{R}^T y = b \longrightarrow z = D^{-1}y \longrightarrow \tilde{R}x = z$ .

**Bemerkung 2.14**

Wieder Spaltenskalarprodukte der oberen Hälfte der Matrixelemente ( $\tilde{R}, U$ ) notwendig  $\longrightarrow$  die „fill-in“- Positionen sind exakt bekannt, woraus sich das „Profil“, „sky-line“- der Matrix  $A$

**Bemerkung 2.15**

$A$  symmetrisch positiv definit  $\implies d_{kk} > 0 \forall k$ , die Crout-Zerlegung kann auch für indefinite Matrizen versucht werden (dann eben  $d_{kk} < 0$  für einige  $k$ )

Gefahr: kleine  $|d_{kk}| \implies$  wäre Pivotisierung nötig.

**2.1.5 Spezialfall Tridiagonalmatrizen**

d.h. Bandbreite 2 (oder 3 je nach Definition)

$$\begin{array}{lll} a_{11} \dots a_{nn} & \text{besetzt} & a_{ii} = \alpha_i \\ a_{12} \dots a_{n-1n} & \text{besetzt} & a_{i-1i} = \beta_{i-1} \quad i = 2, \dots, n \\ a_{21} \dots a_{nn-1} & \text{besetzt} & a_{ii-1} = \gamma_i \quad i = 2, \dots, n \end{array}$$

sonst  $a_{ij} = 0$

Gleichungssystem:

$$\begin{array}{rcl} \alpha_1 x_1 + \beta_1 x_2 & = & b_1 \\ \gamma_2 x_1 + \alpha_2 x_2 + \beta_2 x_3 & = & b_2 \\ & \vdots & \\ \gamma_{n-1} x_{n-2} + \alpha_{n-1} x_{n-1} + \beta_{n-1} x_n & = & b_{n-1} \\ \gamma_n x_{n-1} + \alpha_n x_n & = & b_n \end{array}$$

definiert man noch  $\gamma_1 = 0$  und  $\beta_n = 0$  so gilt:

$$\boxed{\gamma_i x_{i-1} + \alpha_i x_i + \beta_i x_{i+1} = b_i \forall i}$$

Ges.: einfacher Algorithmus, um die  $x_i$  zu bestimmen

Idee: drücken  $x_i$  durch  $x_{i+1}$  aus

1. Gleichung  $\alpha_1 x_1 + \beta_1 x_2 = b_1$

$$x_1 := \frac{b_1}{\alpha_1} - \frac{\beta_1}{\alpha_1} x_2$$

$$x_1 := B_1 - A_1 x_2, \quad B_1 = \frac{b_1}{\alpha_1}, \quad A_1 = \frac{\beta_1}{\alpha_1}$$

Einsetzen in 2. Gleichung

$$\gamma_2 x_1 + \alpha_2 x_2 + \beta_2 x_3 = b_2$$

$$\alpha_2 x_2 = (b_2 - \gamma_2 x_1 - \beta_2 x_3)$$

$$\alpha_2 x_2 = (b_2 - \gamma_2 B_1 + \gamma_2 A_1 x_2 - \beta_2 x_3)$$

$$(\alpha_2 - \gamma_2 A_1) x_2 = b_2 - \gamma_2 B_1 - \beta_2 x_3$$

$$x_2 := B_2 - A_2 x_3, \quad \text{mit}$$

$$B_2 = \frac{b_2 - \gamma_2 B_1}{\alpha_2 - \gamma_2 A_1}, \quad A_2 = \frac{\beta_2}{\alpha_2 - \gamma_2 A_1}$$

usw.: es gilt  $x_k = B_k - A_k x_{k+1}$   $k = 1, \dots, n-1$ , wobei

$$B_k = \frac{b_k - \gamma_k B_{k-1}}{\alpha_k - \gamma_k A_{k-1}}$$

$$A_k = \frac{\beta_k}{\alpha_k - \gamma_k A_{k-1}}$$

jetzt:  $x_{n-1} = B_{n-1} - A_{n-1} x_n$

letzte Gleichung

$$\gamma_n x_{n-1} + \alpha_n x_n = b_n$$

$$\gamma_n B_{n-1} - \gamma_n A_{n-1} x_n + \alpha_n x_n = b_n$$

$$x_n = \frac{b_n - \gamma_n B_{n-1}}{\alpha_n - \gamma_n A_{n-1}} = B_n$$

**Algorithmus 2.16 für Tridiagonalgleichungssystem**

<u>input:</u>	HD.: $\alpha_1 \dots \alpha_n$	<u>output:</u> $x_1 \dots x_n$ Lösung
	obere ND. $\beta_1 \dots \beta_{n-1}$	
	untere ND. $\gamma_2 \dots \gamma_n$	
	rechte Seite $b_1 \dots b_n$	

1. Start:  $B_0 = A_0 = \gamma_1 = 0$ 2. for  $k = 1, \dots, n$  do

$$\begin{cases} A_k & := \beta_k / (\alpha_k - \gamma_k A_{k-1}) \\ B_k & := (b_k - \gamma_k B_{k-1}) / (\alpha_k - \gamma_k A_{k-1}) \end{cases}$$

3.  $x_n := B_n$ 4. for  $k = n - 1$  down to 1

$$x_k := B_k - A_k x_{k+1}$$

**Bemerkung 2.17**

Dieser Algorithmus entspricht der Durchführung des GAUSSchen Algorithmus ohne Pivotisierung  $\implies$  also nicht immer stabil!

anwendbar:

bei diagonalüberwiegenden Matrizen  $|\alpha_i| \geq |\beta_i| + |\gamma_i|$  (auch bei symmetrischen, positiv definiten Matrizen)

## 2.2 Fehlerbetrachtung bei linearen Gleichungssystemen

Fortpflanzung von Fehlern in der rechten Seite nach Theorie aus Kapitel 1  $x_k = \varphi_k(b_1 \dots b_n)$ , wir bräuchten  $\frac{\partial \varphi_k}{\partial b_i} = (A^{-1})_{ki}$  dh. sehr viele Größen, diese sind unhandlich, deshalb betrachtet man Vektornormen für Fehler, dh.  $x \in \mathbb{R}^n$ :  $\|x\|$  eine Norm

**Beispiel 2.18**

$$\|x\|_p = (\sum |x_i|^p)^{1/p} \quad \begin{array}{ll} p = 1: & \|x\|_1 = \sum |x_i| \quad \text{Summennorm} \\ p = 2: & \|x\|_2 = \sqrt{\sum |x_i|^2} \quad \text{Euklidische V.-n.} \\ p = \infty: & \|x\|_\infty = \max_i |x_i| \quad \text{Maximumnorm} \end{array}$$

benötigen analog Matrixnormen insbesondere für eine Abschätzung der Art

$$\boxed{\|Ax\| \leq \|A\| \|x\|}$$

Eine Matrixnorm  $\|\cdot\|$  heißt verträglich mit einer Vektornorm  $\|\cdot\|$  wenn  $\forall x \in \mathbb{R}^n$  obige Ungleichung gilt. Wird  $\|A\| = \max_{x \neq 0} \|Ax\|/\|x\|$  definiert, so ist diese stets mit der benutzten Vektornorm verträglich.,

z.B.  $\|A\|_p = \max_{x \neq 0} \|Ax\|_p / \|x\|_p$

$$p = 1 \quad \|A\|_1 = \max_j \sum_i |a_{ij}| \quad \text{Spaltensummennorm}$$

$$p = 2 \quad \|A\|_2 = \rho(A^*A)^{1/2} \text{ (Spektralradius)} \quad \text{Spektralnorm}$$

$$p = \infty \quad \|A\|_\infty = \max_i \sum_j |a_{ij}| \quad \text{Zeilensummennorm}$$

$$\text{oft auch } \|A\|_F = \left(\sum_{ij} |a_{ij}|^2\right)^{1/2} = \text{tr}(A^*A)^{1/2} \quad \text{Frobeniusnorm}$$

$$(\text{tr} B = \sum b_{ii} = \sum \lambda_i(B))$$

### Bemerkung 2.19

Matrixnorm erfüllt  $\|AB\| \leq \|A\| \|B\|$

### Bemerkung 2.20

1. Fehlerentwicklung aus Fehler in der rechten Seite:

$\implies b$  hat absoluten Fehler  $\Delta b \in \mathbb{R}^n$

(relativer Fehler definiert Zahl  $\delta b = \|\Delta b\|/\|b\|$ ) und das führt zu den absoluten Fehler in der Lösung von  $\Delta x$   $A(x + \Delta x) = b + \Delta b$  da  $Ax = b$

$$\boxed{A \Delta x = \Delta b}$$

typisch für lineare Zusammenhänge von Input- und Output-Größen ist

$$\|\Delta x\| = \|A^{-1} \Delta b\| \leq \|A^{-1}\| \|\Delta b\|$$

(für eine mit Vektornorm verträgliche Matrixnorm)  
besser: relativer Fehler

**Definieren:**

$$\begin{aligned} \delta x &= \frac{\|\Delta x\|}{\|x\|} = \frac{\|A^{-1} \Delta b\|}{\|x\|} \cdot \frac{\|b\|}{\|b\|} \\ &= \frac{\|A^{-1} \Delta b\| \|Ax\|}{\|b\| \|x\|} \leq \|A^{-1}\| \|A\| \delta b \end{aligned}$$

Die Zahl  $\kappa(A) = \|A\| \|A^{-1}\|$  heißt Konditionszahl der Matrix  $A$  (von der jeweils gewählten Matrixnorm abhängig).

Oft wird die Spektralkondition :

$$\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\rho(A^*A)^{1/2}}{\lambda_{\min}(A^*A)^{1/2}} = \frac{\sigma_1(A)}{\sigma_n(A)} \quad \sigma_i(A) = \sqrt{\lambda_i(A^*A)}$$

benutzt.

2. Die Fehlerentwicklung bei der LU-Zerlegung ist schwierig zu untersuchen. Es gibt Rückwärtsabschätzungen, die die üblichen Aussagen liefern: Ergebnis  $L \cdot U$  ist exakt für  $A + \Delta A$  mit  $\|\Delta A\| \leq \epsilon_{mach} \|A\| \cdot f(n) \leftarrow$  (je nach Verfahren / verschiedene Pivotisierung). Somit ist nicht  $Ax = b$  sondern

$$LU\tilde{x} = (A + \Delta A)\tilde{x} = b \quad \text{mit } \tilde{x} = x + \Delta x$$

gelöst.

Abeschätzung des resultierenden Fehlers :

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

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

Voraussetzung:  $\|A^{-1} \Delta A\| < 1$

### Hilfssatz 2.21

$$\begin{aligned} \|F\| < 1 &\implies (I - F)^{-1} = \sum_{k=0}^{\infty} F^k \\ &\implies \|(I - F)^{-1}\| \leq \frac{1}{1 - \|F\|} \quad \text{Neumannsche Reihe} \end{aligned}$$

$$\begin{aligned} F = -A^{-1} \Delta A &\implies (I + A^{-1} \Delta A)^{-1} - I = \sum_{n=1}^{\infty} (-1)^n (A^{-1} \Delta A)^n \\ &= - \sum_{n=1}^{\infty} (-1)^{n-1} (A^{-1} \Delta A)^{n-1} A^{-1} \Delta A \\ &= -(I + A^{-1} \Delta A)^{-1} A^{-1} \Delta A \\ &= -(A + \Delta A)^{-1} \Delta A \\ (A + \Delta A)^{-1} - A^{-1} &= -(A + \Delta A)^{-1} \Delta A A^{-1} \end{aligned}$$

$$\begin{aligned}
\frac{\|\Delta x\|}{\|x\|} &= \frac{\|(A + \Delta A)^{-1} \Delta Ax\|}{\|x\|} \\
&\leq \|(A + \Delta A)^{-1} \Delta A\| \\
&\leq \frac{\|A^{-1} \Delta A\|}{1 - \|A^{-1} \Delta A\|} \\
\text{da } \frac{x}{1-x} &\text{ monoton wachsend für } x \in [0, 1) : \\
&\leq \frac{\|A^{-1}\| \|\Delta A\|}{1 - \|A^{-1}\| \|\Delta A\|} = \frac{\kappa(A) \delta_A}{1 - \kappa(A) \delta_A}
\end{aligned}$$

mit  $\delta_A = \frac{\|\Delta A\|}{\|A\|}$  als relativen Matrixfehler

## 2.3 Iterationsverfahren

### 2.3.1 Elementare Iterationsverfahren

Grundidee:

(1) Überführung der Gleichung

$$Ax^* = b \tag{1}$$

in eine Form  $x^* = Bx^* + c$  und Iteration  $x^{(k+1)} = Bx^{(k)} + c$ , die gegen die exakte Lösung  $x^*$  von konvergieren soll.

→ Aufspaltung  $A = C - N$  („convergent splitting“)  $C$  invertierbar

$$\begin{aligned}
Ax^* &= b \implies Cx^* - Nx^* = b \\
x^* &= C^{-1}(Nx^* + b) \\
x^* &= C^{-1}Nx^* + C^{-1}b \\
B &= C^{-1}N = I - C^{-1}A, \text{ wobei } c = C^{-1}b
\end{aligned}$$

(2) iterierfähige Gestalt  $\implies \boxed{x^{(k+1)} = C^{-1}Nx^{(k)} + c}$  (3)  
andere Möglichkeit der Darstellung von (3):

$$\begin{aligned}
x^{(k+1)} &= C^{-1}(C - A)x^{(k)} + c \\
&= x^{(k)} - C^{-1}(Ax^{(k)} - b) \\
&\quad Ax^{(k)} - b = r^{(k)} \text{ Residuum des Gleichungssystems} \\
x^{(k+1)} &= x^{(k)} - C^{-1}r^{(k)} \tag{4} \\
&\quad (\text{benötigen wir später für Verallgemeinerung})
\end{aligned}$$

Beschäftigen uns mit Konvergenz in Form (3):

$$\begin{array}{lcl} x^{(k+1)} & = & C^{-1}Nx^{(k)} + c \\ & = & Bx^{(k)} + c \\ \hline x^* & = & Bx^* + c \\ x^{(k+1)} - x^* & = & B(x^{(k)} - x^*) \end{array} \quad \begin{array}{l} + \\ - \\ - \\ - \end{array}$$

$B$  wird auch Fehlerübergangsoperator genannt, offenbar:

$\|B\| < 1$  (für irgendeine Matrixnorm, die mit der Vektornorm verträglich)  $\implies$  Fehler  $\|x^{(k)} - x^*\| \longrightarrow 0$

### Satz 2.22

Ist  $\|B\| < 1$ , so konvergiert das Iterationsverfahren (3) gegen die exakte Lösung  $x^*$ .

**Beweis:**

$$\begin{aligned} \|x^{(k)} - x^*\| &= \|B^k(x^{(0)} - x^*)\| \\ &\leq \|B\|^k \|x^{(0)} - x^*\| \end{aligned}$$

□

### Satz 2.23

Ist  $\rho(B) < 1 \iff$  das Iterationsverfahren (3) konvergiert gegen die exakte Lösung (wegen  $\rho(B) \leq \|B\| \forall$  Matrixnormen ist dieser Satz schärfer).

**Beweis:** Mit Hilfe der Eigenzerlegung von  $B$ .

## 2.3.2 Einzel- und Gesamtschrittverfahren

Sei  $A = L + D + U$ ,

$L$  strenge untere Dreiecksmatrix

$U$  strenge obere Dreiecksmatrix

$D = \text{diag}(A)$

### Definition 2.24

$$C = D, \quad N = -L - U$$

$\implies$  **Iterationsvorschrift**

$$x^{(k+1)} = D^{-1}(b - Lx^{(k)} - Ux^{(k)}) \quad (\text{Gesamtschrittverfahren, Jacobi-Iteration})$$

elementweise Betrachtung:

$$x^{(k)} = \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix}, \quad x^{(k+1)} = \begin{bmatrix} x_1^{(k+1)} \\ \vdots \\ x_n^{(k+1)} \end{bmatrix}$$

$$x_j^{(k+1)} = \frac{1}{d_{jj}} \left( b_j - \sum_{i < j} a_{ji} x_i^{(k)} - \sum_{i > j} a_{ji} x_i^{(k)} \right), \quad j = 1 \dots n$$

hieraus Idee: Einzelschrittverfahren (Gauß-Seidel-Iteration)

$x_1^{(k+1)} \dots x_{j-1}^{(k+1)}$  schon vorher berechnet, also in der obigen Formel benutzbar:

$$x_j^{(k+1)} = \frac{1}{d_{jj}} \left( b_j - \sum a_{ji} x_i^{(k+1)} - \sum a_{ji} x_i^{(k)} \right)$$

(dies ist die praktische Durchführung, gleiche Operationszahl wie GSV)  
theoretische Untersuchung:

$$\begin{aligned} x^{(k+1)} &= D^{-1}(b - Lx^{(k+1)} - Ux^{(k)}) \\ \implies (D + L)x^{k+1} &= b - Ux^{(k)} \\ x^{(k+1)} &= (D + L)^{-1}(b - Ux^{(k)}) \end{aligned}$$

d.h. es ist elementare Iteration mit

$$\underline{C = D + L, \quad N = -U}$$

Somit ist die Konvergenz beider Verfahren klar:

$$\begin{aligned} \rho(D^{-1}(L + U)) &= \rho(I - D^{-1}A) < 1 && \iff \text{GSV konvergent} \\ \rho((D + L)^{-1}U) &= \rho(I - (D + L)^{-1}A) < 1 && \iff \text{ESV konvergent} \end{aligned}$$

beides ist z.B. bei diagonal dominanten Matrizen erfüllt.

### Definition 2.25

A streng (schwach) diagonal dominant

$$\iff |a_{kk}| > (\geq) \sum_{j \neq k} |a_{kj}| \quad \forall k$$

( $\geq$ )  $\longrightarrow$  in mindestens einer Matrixzeile gilt  $>$ .

### 2.3.3 Unter- und Überrelaxation

Die Differenz  $x^{(k+1)} - x^{(k)}$  („Verbesserung“ der Näherung)

könnte man überhöhen, um mehr in Richtung der Lösung zu kommen:

$$\begin{aligned}\tilde{x}^{(k)} &= Bx^{(k)} + c \\ x^{(k+1)} &= x^{(k)} + \omega(\tilde{x}^{(k)} - x^{(k)})\end{aligned}\tag{3'}$$

$\omega > 1$  Überrelaxation  
 $\omega < 1$  Unterrelaxation (selten benutzt)  
 $\omega = 1$  elementares Iterationsverfahren

einfaches Aussehen in der Form (4):

$$\begin{aligned}\tilde{x}^{(k)} &= x^{(k)} - C^{-1}r^{(k)} \\ \implies x^{(k+1)} &= x^{(k)} - \omega C^{-1}r^{(k)}\end{aligned}\tag{4'}$$

#### Beispiel 2.26

SOR (sukzessive Überrelaxation) entsteht durch Definition  $C = D + \omega L$

$$\begin{aligned}\implies x^{(k+1)} &= x^{(k)} - \omega(D + \omega L)^{-1}(Ax^{(k)} - b) \\ &= \omega(D + \omega L)^{-1}b + (D + \omega L)^{-1}((1 - \omega)D - \omega U)x^{(k)} \\ x^{(k+1)} &= c(\omega) + B(\omega)x^{(k)}\end{aligned}$$

#### Bemerkung 2.27

In der praktischen Resalisierung ähnlich wie Einzelschrittverfahren:

$$x_j^{k+1} = \frac{\omega}{d_{jj}}(b_j - \sum_{i < j} a_{ji}x_i^{(k+1)} - \sum_{i > j} a_{ji}x_i^{(k)}) + (1 - \omega)x_j^{(k)}$$

#### Bemerkung 2.28

Die bisher betrachteten Iterationsverfahren werden nur angewendet in folgenden Situationen:

- Matrix  $A$  stark diagonal dominant (gut konditioniert),
- Matrix  $A$  schlecht konditioniert, Verfahren wird zur Dämpfung „hochfrequenter“ Fehleranteile verwendet (sogenannte Glätter) nicht aber zur Lösung des gegebenen Gleichungssystems. (später: Multigrid-Methode)

### 2.3.4 Instationäre Iterationsverfahren

#### Grundidee:

$x^{(k)}$  Näherung für Lösung  $x^*$ , konstruieren „Suchrichtung“  $q^{(k)}$   
 (Bsp.: Elementare Iterationsverfahren:  $q^{(k)} = C^{-1}r^{(k)}$ )  
 Definieren:  $x^{(k+1)} = x^{(k)} + \alpha_k q^{(k)}$ , so daß neuer Fehler  $z^{(k+1)} = x^{(k+1)} - x^*$   
 „minimal“ wird. „minimal“ heißt Fehler  $z^{(k+1)}$  bzgl. einer besonders zu definierenden Norm (übliche Euklidische Vektornorm ist ungeeignet:  $x^*$  unbekannt).

#### Definition 2.29

Skalarprodukt im  $\mathbb{R}^n$

$$\langle x, y \rangle = y^\top Fx \quad \forall x, y \in \mathbb{R}^n$$

wobei  $F = F^\top$  positiv definit Matrix ( $n \times n$ ).

(im komplexen  $\mathbb{C}^n$  :  $y^* Fx, F = F^* > 0$ )

Das Skalarprodukt  $\langle \cdot, \cdot \rangle$  induziert die Norm

$$\|x\| = \langle x, x \rangle^{\frac{1}{2}} \quad \forall x \in \mathbb{R}^n$$

$\alpha_k$  so berechnen, daß Fehler  $z^{(k+1)}$  minimale Norm hat:

$$\begin{aligned} \langle z^{(k+1)}, z^{(k+1)} \rangle &= \langle z^{(k)} + \alpha_k q^{(k)}, z^{(k)} + \alpha_k q^{(k)} \rangle \\ &= \|z^{(k)}\|^2 + 2\alpha_k \langle z^{(k)}, q^{(k)} \rangle + \alpha_k^2 \|q^{(k)}\|^2 \end{aligned}$$

$$\|z^{(k+1)}\| \text{ minimal} \iff \boxed{\alpha_k = -\frac{\langle z^{(k)}, q^{(k)} \rangle}{\langle q^{(k)}, q^{(k)} \rangle}}$$

Hieraus entstehen viele Varianten durch

1. Festlegung des konkreten  $q^{(k)}$
2. Festlegung des konkreten Skalarprodukts

$\langle z^{(k)}, q^{(k)} \rangle = q^{(k)\top} Fz^{(k)}$  muß berechenbar sein, z.B. bei  $F = ZA$

$$\begin{aligned} q^{(k)\top} ZAz^{(k)} &= q^{(k)\top} Z(Ax^{(k)} - \underbrace{Ax^*}_b) \\ &= q^{(k)\top} Zr^{(k)} \end{aligned}$$

aber zusätzlich:  $ZA$  muß symmetrisch positiv definit sein!

**Beispiel 2.30**  $F = A$  Koordinatenrelaxation

$$q^{(k)} = e_k \text{ (modulo } n),$$

$$\implies \boxed{\alpha_k = -\frac{e_k^\top r^{(k)}}{a_{kk}}}$$

**Beispiel 2.31**  $F = I$  Kaczmarz - Verfahren

$$q^{(k)} = A^\top e_k \text{ (modulo } n) \longrightarrow k\text{-te Zeile von } A$$

$$\implies \boxed{\alpha_k = -\frac{(A^\top e_k)^\top z^{(k)}}{\|q^{(k)}\|_2^2} = -\frac{e_k^\top r^{(k)}}{\|q^{(k)}\|_2^2}}$$

**Beispiel 2.32**  $F = A$  Gradientenverfahren

$$q^{(k)} = C^{-1}r^{(k)}$$

weitere Möglichkeiten für Wahl von  $F$  :  $F = A^\top A, F = A^\top C^{-1}A$

$$\implies \boxed{\alpha_k = -\frac{(q^{(k)})^\top r^{(k)}}{(q^{(k)})^\top A q^{(k)}}$$

heißt Gradientenverfahren (Verfahren des steilsten Abstiegs)

**Namensgebung:**

definieren:  $\mu(x) = \langle C^{(-1)}Ax, x \rangle - 2 \langle C^{(-1)}b, x \rangle$

mit  $\langle x, y \rangle = (Cx, y) = C^\top Cx$  für  $A = A^\top$  positiv definit,  $C = C^\top$  positiv definit

berechnen Gradient:

1. Richtungsableitung :

$$\frac{\partial \mu}{\partial s} = \lim_{\tau \rightarrow 0} \frac{\mu(x + \tau s) - \mu(x)}{\tau}$$

$$\|s\|^2 = \langle s, s \rangle = 1$$

2. suchen  $s$ , so daß  $\frac{\partial \mu}{\partial s} \rightarrow \max \quad \|s\| = 1$

$\implies$  Gradient  $\sim C^{-1}(Ax - b)$

**Beispiel 2.33** Verfahren der konjugierten Gradienten

$F = A$  ( $A = A^T$  positiv definit)  
 $q^{(k)} = w^{(k)} + \beta_{k-1}q^{(k-1)}$  mit  $w^{(k)} = C^{-1}r^{(k)}$   
 $\longrightarrow$ , so daß  $\langle q^{(k)}, q^{(k-1)} \rangle = 0$   
 und wieder  $\langle z^{k+1}, z^{k+1} \rangle \longrightarrow \min_{\alpha}$

• hieraus zeigt man induktiv:

$$\begin{aligned} \langle q^{(i)}, q^{(j)} \rangle &= 0 \quad \forall i \neq j \\ \langle z^{(k+1)}, q^{(i)} \rangle &= 0 \quad \forall i \leq k \end{aligned}$$

$\implies \{q^{(i)}\}$  Orthonormalsystem

$\implies$  bei  $k = n - 1$   $z^{(n)} \perp \text{span}(q^{(1)} \dots q^{(n-1)}) = \mathbb{R}^n \implies z^{(n)} \equiv 0$

$\longrightarrow$  endliches Verfahren

**Algorithmus 2.34**

(im einfachen Euklidischen Skalarprodukt aufgeschrieben )

Start:  $q^0 = w^0 = C^{-1}r^0$

$$\begin{aligned} \alpha_k &= -\frac{\langle q^{(k)}, r^{(k)} \rangle}{\langle Aq^{(k)}, q^{(k)} \rangle} \\ x^{(k+1)} &= x^{(k)} + \alpha_k q^{(k)} \\ r^{(k+1)} &= r^{(k)} - \alpha_k (Aq^{(k)}) \\ w^{(k+1)} &= C^{-1}r^{(k+1)} \\ \beta_k &= \frac{\langle w^{(k+1)}, r^{(k+1)} \rangle}{\langle w^{(k)}, r^{(k)} \rangle} \longrightarrow \text{hergeleitet aus obiger Forderung} \\ q^{(k+1)} &= w^{(k+1)} + \beta_k q^{(k)} \end{aligned}$$

Wird in der Regel in vorkonditionierter Form angewendet:

$C$  „Vorkonditionierung“ ebenfalls symmetrisch positiv definit

klassisch:  $C = I \longrightarrow$  HESTENESS/STIEFEL

**2.3.5 Konvergenztheorie der Iterationsverfahren**

**Satz 2.35**

Bei fast allen Iterationsverfahren gilt

$$z^{(k)} = p_k(\mathcal{A})z^{(0)}$$

mit:  $z^{(k)}$  Fehler im k-ten Schritt

$$z^{(k)} = x^{(k)} - x^*$$

$p_k(t)$  Polynom vom Grade  $k$  mit  $p_k(0) = 1$ .

$\mathcal{A} = C^{-1}A$  Iterationsoperator (bestimmt die Konvergenzgeschwindigkeit)

**Beweis:**

- stationäres Iterationsverfahren:

$$\begin{aligned}x^{(k+1)} &= x^{(k)} - C^{-1}r^{(k)} \\r^{(k)} &= Ax^{(k)} - b = Az^{(k)} \\z^{(k+1)} &= z^{(k)} - C^{-1}Az^{(k)} = (I - C^{-1}A)z^{(k)} \\z^{(k)} &= (I - C^{-1}A)^k z^{(0)}\end{aligned}$$

hier:  $p_k(t) = (1 - t)^k$ , bei Überrelaxation:  $p_k(t) = (1 - \omega t)^k$

- instationäres Iterationsverfahren  
Gradientenverfahren/steilster Anstieg:

$$\begin{aligned}x^{(k+1)} &= x^{(k)} + \alpha_k C^{-1}r^{(k)} \\z^{(k+1)} &= (I + \alpha_k \mathcal{A})z^{(k)} \\z^{(k)} &= (I + \alpha_{k-1} \mathcal{A})(I + \alpha_{k-2} \mathcal{A}) \cdot \dots \cdot (I + \alpha_0 \mathcal{A})z^{(0)} \\p_k(t) &= \prod_{j=0}^{k-1} (1 + \alpha_j t)\end{aligned}$$

- Verfahren der konjugierten Gradienten

$$\begin{aligned}x^{(k+1)} &= x^{(k)} + \alpha_k q^{(k)} \\ \iff z^{(k+1)} &= z^{(k)} + \alpha_k q^{(k)} & (1) \\ q^{(k+1)} &= w^{(k+1)} + \beta_k q^{(k)} & (2)\end{aligned}$$

wobei  $q^{(0)} = w^{(0)}$ ,  $w^{(k)} = C^{-1}r^{(k)} = C^{-1}Az^{(k)} = \mathcal{A}z^{(k)}$

Annahme:

$$\begin{aligned}z^{(k)} &= p_k(\mathcal{A})z^{(0)} \\ q^{(k)} &= \mathcal{A}q_k(\mathcal{A})z^{(0)}\end{aligned}$$

Zeigen Rekursion für beide Polynome

$$p_k(t), q_k(t) \text{ von Grade } k, p_k(0) = 1$$

Induktionsanfang:  $k=0$

$$\begin{array}{ccc} p_0(t) \equiv 1 & & q_0(t) \equiv 1 \\ \downarrow & & \downarrow \\ z^{(0)} = p_0(\mathcal{A})z^{(0)} & & q^{(0)} = w^{(0)} = \mathcal{A}z^{(0)} \quad \text{erfüllt} \end{array}$$

Induktion nach  $k$  aus (1), (2):

$$\begin{aligned} p_{k+1}(t) &= p_k(t) + \alpha_k \underbrace{tq_k(t)}_{\text{Grad } k+1} \\ p_{k+1}(0) &= p_k(0) = 1 \end{aligned}$$

$$\begin{aligned} & w^{(k+1)} + \beta_k q^{(k)} \\ & \downarrow \\ q^{k+1} = \mathcal{A}q_{k+1}(\mathcal{A})z^{(0)} &= \mathcal{A}p_{k+1}(\mathcal{A})z^{(0)} + \beta_k \mathcal{A}q_k(\mathcal{A})z^{(0)} \\ \implies q_{k+1}(t) &= p_{k+1}(t) + \beta_k q_k(t) \end{aligned}$$

□

Fazit: Stets gilt

$$\boxed{z^{(k)} = p_k(\mathcal{A})z^{(0)} \quad (p_k(0) = 1)}$$

### Konvergenztheorie

Zerlege  $z^{(0)}$  nach den Eigenvektoren von  $\mathcal{A}$  (setzen  $\mathcal{A}$  als diagonalisierbar voraus):

$$\begin{aligned} \mathcal{A}\varphi_i &= \lambda_i \varphi_i \quad z^{(0)} = \sum_{i=1}^n \gamma_i \varphi_i \\ \implies z^{(k)} &= p_k(\mathcal{A}) \sum_{i=1}^n \gamma_i \varphi_i = \underbrace{\sum_{i=1}^n \gamma_i p_k(\lambda_i) \varphi_i}_{\text{klein } \forall i \text{ (bzw. konvergiert gegen Null } k \rightarrow \infty)} \\ &\implies z^{(k)} \rightarrow 0 \end{aligned}$$

also ist Frage der Konvergenz  $z^{(k)} \rightarrow 0$  in ein Approximationsproblem überführt worden: Wie schnell konvergieren die Werte  $|p_k(\lambda_i)|$  gegen Null ( $k \rightarrow \infty$ )? (Beachte:  $p_k(0) = 1 \forall k$ ) Einschränkung: Sei  $\mathcal{A}$  selbstadjungiert bzgl.  $\langle \cdot, \cdot \rangle$  z.B.:  $C = C^\top$ ,  $A = A^\top +$  positiv definit und  $F = A$  ( $\implies \mathcal{A}$  hat reelle Eigenwerte)

**Bemerkung 2.36**

$$\|x\|^2 = \langle x, x \rangle = (Ax, x)$$

$\mathcal{A}$  ist selbstadjungiert bezüglich diesem Skalarprodukt

$$\begin{aligned} \langle \mathcal{A}x, x \rangle &= (AC^{-1}Ax, x) = (Ax, C^{-1}Ax) \\ &= \langle x, \mathcal{A}x \rangle \end{aligned}$$

$\implies$  Eigenvektoren sind ONB bezüglich diesen Skalarproduktes und alle Eigenwerte von  $\mathcal{A}$  sind reell

$$\begin{aligned} \|z^{(k)}\|^2 &= \left\| \sum \gamma_i p_k(\lambda_i) \varphi_i \right\|^2 = \sum \gamma_i^2 (p_k(\lambda_i))^2 \\ &\leq \max_i (p_k(\lambda_i))^2 \cdot \sum \gamma_i^2 = \rho_k^2 \cdot \|z^{(0)}\|^2 \end{aligned}$$

**Beispiel 2.37** Überrelaxation

$$p_k(t) = (1 - \omega t)^k$$

Habe  $\mathcal{A} = C^{-1}A$  nur reelle EW  $0 < \lambda_i \leq \dots \leq \lambda_n$

(z.B.:  $C = C^\top$  positiv definit,  $A = A^\top$ )

$|p_k(\lambda_i)| = |1 - \omega \lambda_i|^k \rightarrow \min_{\omega} \quad \omega$  einzig frei wählbare Größe

$$\begin{aligned} \omega_{opt} &= \frac{2}{\lambda_1 + \lambda_n} \\ \implies \rho_k &= \max_i |1 - \omega \lambda_i|^k = |1 - \omega \lambda_1|^k \\ &= \left| 1 - \frac{2\lambda_1}{\lambda_1 + \lambda_n} \right|^k = \left| \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right|^k \\ \rho_k &= \left| \frac{1 - \xi}{1 + \xi} \right|^k, \quad \xi = \frac{\lambda_1}{\lambda_n} \\ \implies \|z^{(k)}\| &\leq \eta^k \|z^{(0)}\| \\ \eta &\leq \frac{1 - \xi}{1 + \xi} \quad \text{Konvergenzquotient} \end{aligned}$$

(bei optimalem  $\omega = \frac{2}{\lambda_1 + \lambda_n}$ , im allgemeinen unbekannt)

**Beispiel 2.38** PCG

Wir wissen  $z^{(k)} = p_k(\mathcal{A})z^{(0)}$   $p_k(0) = 1$  Orthogonalitäten

$$\begin{aligned} & \langle z^{(k)}, q^{(i)} \rangle = 0 \quad \forall i < k \\ & z^{(k)} \perp \text{span}(q^{(0)}, \dots, q^{(k-1)}) \\ \iff & \|z^{(k)}\| = \min \|\tilde{p}_k(\mathcal{A})z^{(0)}\| \\ & \forall \tilde{p}_k(t) : \tilde{p}_k(0) = 1 \end{aligned}$$

d.h. PCG konstruiert optimales Polynom  $p_k(t)$

$$\begin{aligned} & \langle z^{(k)}, q^{(i)} \rangle = 0 \quad \forall i < k \\ & z^{(k)} \perp \text{span}(q^{(0)}, \dots, q^{(k-1)}) \iff \|z^{(k)}\| = \min \|\tilde{p}_k(\mathcal{A})z^{(0)}\| \\ & \forall \tilde{p}_k(t) \tilde{p}_k(0) = 1 \end{aligned}$$

d.h. PCG konstruiert „optimales Polynom“  $p_k(t)$

**Definition 2.39**

$p_k(t)$  optimal wenn  $\|p_k(\mathcal{A})z^{(0)}\| \leq \|\tilde{p}_k(\mathcal{A})z^{(0)}\| \forall \tilde{p}_k(0) = 1$

**Bemerkung 2.40**

$$\rho_k = \max_i |\tilde{p}_k(\lambda_i)|$$

Wenn  $\rho_k$  abgeschätzt werden kann, folgt eine Abschätzung für die Konvergenzgeschwindigkeit:

$$\|z^{(k)}\| \leq \rho_k \|z^{(0)}\|$$

$\rho_k$  ist im allgemeinen nicht angebbarda es von allen Eigenwerten abhängig ist

**Ausweg:** betrachten ein besonders einfach gewähltes Polynom  $\tilde{p}_n(A)$

$$\begin{aligned} \|z^{(k)}\| &= \|p_k(\mathcal{A})z^{(0)}\| \\ &\leq \|\tilde{p}_k(\mathcal{A})z^{(0)}\| \\ &\leq \max_i |\tilde{p}_k(\lambda_i)| \|z^{(0)}\| \quad \text{mit } \tilde{p}_k(0) = 1 \\ &\leq \max |\tilde{p}_k(t)| \|z^{(0)}\| \forall t \in [\lambda_1, \lambda_n] \\ &\quad \lambda_1 \text{ kleinster EW, } \lambda_n = \varrho(\mathcal{A}) \text{ größter EW} \end{aligned}$$

Aus der Approximationstheorie (später) kennt man die Lösung (bestes Polynom  $\tilde{p}_k$ ) als

$$\tilde{p}_k(t) = \frac{T_k(g(t))}{T_k(g(0))} \quad \begin{aligned} g(\lambda_1) &= -1 \\ g(\lambda_n) &= +1 \\ g(t) &= \frac{2t}{\lambda_n - \lambda_1} - \frac{\lambda_1 + \lambda_n}{\lambda_n - \lambda_1} \end{aligned}$$

$T_k$  Tschebyschew-Polynom  $k$ -ten Grades

$$\begin{aligned}T_k(\tau) &= \frac{1}{2}((\tau + \sqrt{\tau^2 - 1})^k + (\tau - \sqrt{\tau^2 - 1})^k) \\T_0 &= 1 \\T_1 &= \tau T_{k+1} = 2\tau T_k - T_{k-1} \\ \text{also } \implies & |T_k(\tau)| \leq 1 \quad \forall \tau \in [-1, 1]\end{aligned}$$

$$\rho_k \leq \frac{1}{|T_k(g(0))|} = \left[ T_k \left( \frac{1 + \xi}{1 - \xi} \right) \right]^{-1} \leq 2 \left( \frac{1 - \sqrt{\xi}}{1 + \sqrt{\xi}} \right)^k \quad \text{mit } \xi = \frac{\lambda_{\min}}{\lambda_{\max}}$$

# Kapitel 3

## Interpolation/Approximation von reellen Funktionen

### 3.1 Grundaufgabe

Aufgabe:

Finde eine Funktion  $g \in \mathcal{M}$ , die ein (vorgegebenes) Gütekriterium für gegebenes a), b) und c) erfüllt.

wobei:

- a) geg.: Funktion  $f$
- b) geg.:  $\mathcal{M}$  Menge von Funktionen
- c) geg.: Gütekriterium

Erklärungen

- a) eventuell ist die Funktion  $f$  nur durch diskrete Daten (etwa:  $f(x_i), i = 1, \dots, n$ ) gegeben
- b)  $\mathcal{M}$  ist oftmals ein endlich dimensionaler Funktionenraum (etwa:  $\Pi_n = \{ \text{aller Polynome bis zum Grad } n \}$ )
- c) Interpolation:  
die Funktion  $g$  muß an den Stellen  $\{x_i\}$  gewisse Werte annehmen (etwa :  $g(x_i) = f(x_i)$  oder  $g'(x_i)$  vorgegeben)

Approximation:

$g$  soll in einer bestimmten Norm minimal von  $f$  abweichen

Nutzen: Benutzen dann  $g(x)$  als Annäherung für  $f(x)$

- (a) um  $f(\bar{x})$  zu berechnen an Werten  $\bar{x} \neq x_i$  (Interpolation)  
 (b) um  $f(x) = 0$  zu lösen (Abschnitt 4)  
 (c) um  $\int_a^b f(x)dx$  näherungsweise zu berechnen (Abschnitt 5)

## 3.2 Grundlegende Theorie der Interpolationsaufgabe

geg.:  $x_0, \dots, x_n$  Stützstellen

$y_0, \dots, y_n$  Stützwerte  $y_i = f(x_i)$

geg.:  $\mathcal{M} = \text{span}(\varphi_0(x), \dots, \varphi_n(x))$

Basis eines endlich dimensionalen Funktionsraumes

ges.:  $g(x) \in \mathcal{M} : g(x) = \sum_{j=0}^n \alpha_j \varphi_j(x)$

mit  $g(x_i) = y_i$  als Lösung eines linearen Gleichungssystem  $(n+1) \times (n+1)$  :

$$\boxed{Ga = b}$$

mit  $a = (\alpha_0 \cdots \alpha_n)^T$ ,  $b = (y_0 \cdots y_n)^T$ ,  $G = (\varphi_j(x_i))_{i,j=0}^n$

### Definition 3.1

Wir schreiben für eine Funktion  $f$  die  $n$ -mal stetig differenzierbar auf ein Intervall  $[a, b]$  ist  $f \in C^{n+1}[a, b]$ .

### Satz 3.2

- Die Interpolationsaufgabe mit der Basis  $(\varphi_0, \dots, \varphi_n)$  ist eindeutig lösbar wenn die HAARSche Determinante  $\det G$  nicht verschwindet.
- Die Basis  $(\varphi_0, \dots, \varphi_n)$  erfüllt die HAARSche Bedingung, wenn  $\det G \neq 0$  für jede Wahl der Stützstellen  $\{x_i\}$ .

## 3.3 Interpolation mit Polynomen

allgemeine Aufgabenstellung

geg.: Stützstellen	$x_0 \cdots x_n$
Stützwerte	$f_0 \cdots f_n$ ( $f_k = f(x_k)$ )
eventuell weitere Vorgaben	z.B. $m_k = f'(x_k)$
$\mathcal{M} = \Pi_m$	
ges.: Polynom	$g \in \Pi_m : g(x_i) = f_i$

**einfachster Fall:** nur Funktionswerte vorgegeben

d.h.  $m = n$ ,  $\exists$  eindeutiges Polynom  $g(x) \in \Pi_n = \text{span}(1, x, \dots, x^n)$

$$G = (x_i^j)_{i,j=0}^n \Rightarrow \det G = \prod_{i < j} (x_i - x_j) \neq 0 \iff x_i \neq x_j$$

Vandermondsche Determinante

hier: Basis  $\varphi_k(x) = x^k$   $k = 0, \dots, n$  wird praktisch nicht verwendet sondern:

$$L_k(x) = \prod_{j \neq k} \frac{x - x_j}{x_k - x_j} \quad \text{bei Lagrange-Interpolation}$$

oder

$$N_k(x) = \prod_{j=0}^{n-1} (x - x_j) \quad \text{bei Newton-Interpolation}$$

$$N_0 \equiv 1$$

### 3.3.1 Lagrange-Interpolation

$$\text{geg.: } f_i = f(x_i) \quad i = 0, \dots, n$$

$$\mathcal{M} = \Pi_n$$

$$\text{ges.: } g(x) \in \Pi_n : g(x_i) = f_i$$

Lagrange-Interpolationspolynom wird direkt angegeben

$$g(x) = \sum_{k=0}^n f_k L_k(x)$$

wobei  $L_k(x)$  alles Polynome vom Grad  $n$  sind, die nur von den Stützstellen  $\{x_i\}$  nicht von Stützwerten  $f_i$  abhängen:

$$L_k(x) = \prod_{j \neq k} \frac{x - x_j}{x_k - x_j}$$

$\implies L_k(x_j) = \delta_{kj}$  also  $g(x_j) = f_j$  offenbar erfüllt

### 3.3.2 Newton-Interpolation

Aufgabe wie bei Lagrange-Interpolation

$$\text{geg.: } \mathcal{M} = \Pi_n, f_i = f(x_i)$$

somit gleiches Polynom  $g(x) \in \Pi_n$  als Lösung der Interpolationsaufgabe, aber andere Darstellung von  $g(x)$ :

$$g(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \dots + c_n(x - x_0) \cdot \dots \cdot (x - x_{n-1})$$

$c_i$  rekursiv berechenbar:

$$c_0 = g(x_0) = f_0$$

$$c_1 =: g(x_1) - c_0 = f_1 - f_0$$

$$c_1 = \frac{f_1 - f_0}{x_1 - x_0}$$

usw. oder rekursive Definition von Interpolationspolynomen  $p_k(x) \in \Pi_k$  auf den Stützstellen  $\{x_0, \dots, x_k\}$   $k = 0, 1, \dots, n$

$$\begin{aligned} p_0(x) &= c_0 = f_0 \\ p_1(x) &= c_0 + c_1(x - x_0) \\ &= p_0(x) + c_1(x - x_0) \text{ also mit } p_1(x_1) = f_1 \text{ folgt} \\ c_1 &= \frac{f_1 - f_0}{x_1 - x_0} = \frac{f_1 - p_0(x_1)}{x_1 - x_0} \end{aligned}$$

usw.

$$\begin{aligned} p_k(x) &= p_{k-1}(x) + c_k(x - x_{k-1}) \cdot \dots \cdot (x - x_0) \\ x &= x_k : c_k = \frac{f_k - p_{k-1}(x_k)}{\prod_{j=0}^{k-1} (x_k - x_j)} \end{aligned}$$

Ergebnis:

$$\begin{aligned} p_0(x) &= f_0 \\ p_k(x) &= p_{k-1}(x) + (f_k - p_{k-1}(x_k)) \cdot \prod_{j=0}^{k-1} \frac{x - x_j}{x_k - x_j}, \quad k = 1, \dots, n \\ g(x) &= p_n(x) \end{aligned}$$

### 3.3.3 Interpolationsfehler/Konvergenz

Die Lagrange und Newton-Interpolation liefern das gleiche Polynom  $g \in \mathcal{M} = \Pi_n$  in unterschiedlicher Darstellung

ges.: Abschätzung des Fehlers  
 $|g(\bar{x}) - f(\bar{x})|$  an einer Stelle  $\bar{x} \neq x_i$  eine solche Aussage ist nur möglich,

wenn Zusatzinformationen über  $f$  bekannt sind, also  $f \in F$  (Funktionenmenge mit gewissen Eigenschaften)

#### Satz 3.3

$\{x_i\} \subset [a, b]$  und  $f \in C^{n+1}[a, b]$  so gilt

$$r(x) = f(x) - g(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^n (x - x_i)$$

**Beweis:**

definiere Hilfsfunktion:  $h(x) = f(x) - g(x) - cw(x)$ ,

$$w(x) = \prod_{i=0}^n (x - x_i),$$

$$\implies h(x_i) = 0 \quad \forall i, \quad \text{für beliebiges } c,$$

$$\text{setzen: } c = \frac{f(\bar{x}) - g(\bar{x})}{w(\bar{x})} \implies h(\bar{x}) = 0 \quad \bar{x} \neq x_i$$

$h$  hat  $(n + 2)$  Nullstellen. **Satz von Rolle:**  $\implies$

$h'$  hat mindestens  $(n + 1)$  Nullstellen

$h''$  hat mindestens  $n$  Nullstellen

$\vdots$

$h^{(n+1)}$  hat mindestens 1 Nullstelle  $\tau$  in  $[a, b]$

$$h^{(n+1)}(x) = f^{(n+1)}(x) - 0 - c(n + 1)!$$

$$c = \frac{f^{(n+1)}(\tau)}{(n + 1)!} = \frac{f(\bar{x}) - g(\bar{x})}{w(\bar{x})}$$

Ist also  $|f^{(n+1)}(x)| \leq \|f^{(n+1)}\|_\infty = M$  beschränkt so gilt  $|r(x)| \leq \frac{M}{(n+1)!} |w(x)|$  □

Frage:  $n \rightarrow \infty$

**WEIERSTASSscher Approximationsatz:**  $f(x) \in C[a, b]$  kann beliebig genau durch Polynome angenähert werden.

aber: feste Vorgabe der Stützstellen  $x_0 \dots x_n$  ( $n \rightarrow \infty$ ) spielt hier besondere Rolle, müssen sogenannte Referenzfolge betrachten

$$X^{(n)} = \{x_0^{(n)} < x_1^{(n)} < \dots < x_n^{(n)}\} \subset [a, b]$$

Es gilt: Für jede Referenzfolge existiert eine Funktion  $f \in C[a, b]$  für die nicht

$$\begin{cases} \|f - g_n\|_\infty \rightarrow 0 & \text{gilt} \\ (g_n \text{ Interpolationspolynom zu Stützstellen } X^{(n)}) \end{cases}$$

### Beispiel 3.4

$f(x) = |x|$  ist eine solche Funktion für  $[a, b] = [-1, 1]$  und äquidistante Unterteilung.

### 3.3.4 Hermite Interpolation

geg.:  $f_i = f(x_i)$  und  $m_i = f'(x_i)$   $i = 0, \dots, n$

$2n + 2$  Daten  $\implies g \in \Pi_{2n+1}$

ges.:  $g(x_i) = f_i$  und  $g'(x_i) = m_i$  gesucht

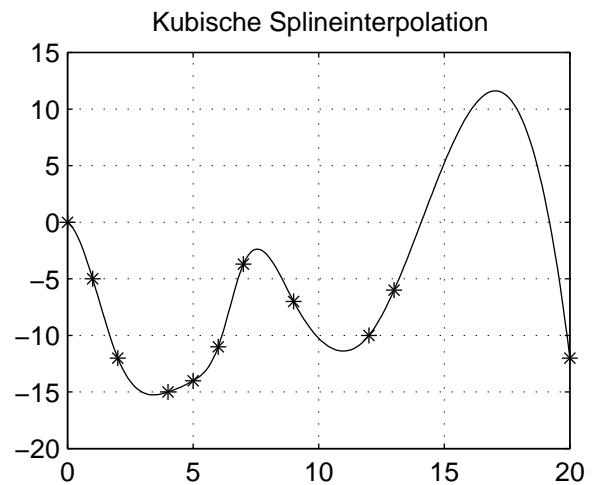
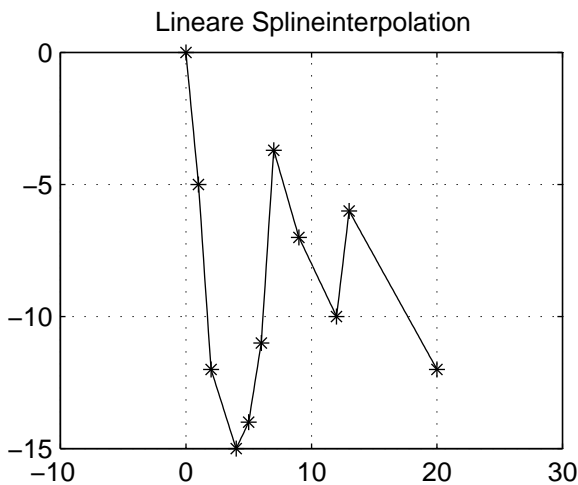
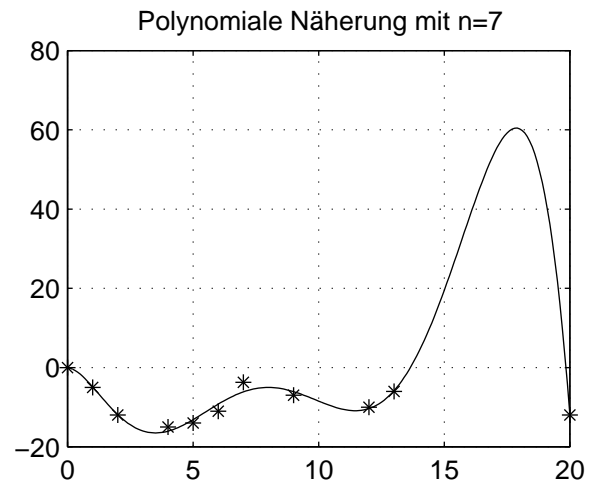
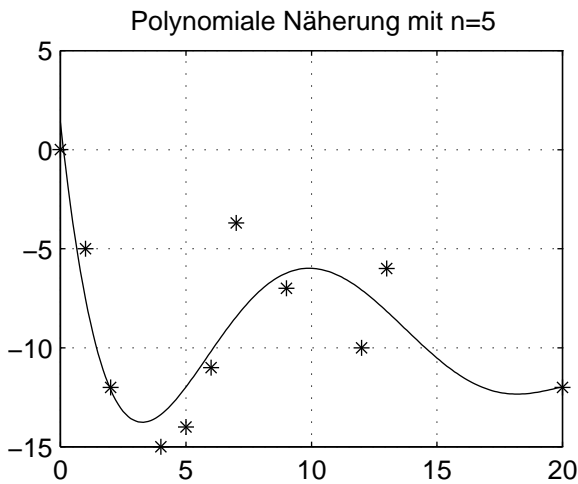
(Lösung: Hermite-Interpolationspolynom, wieder eindeutig bestimmt, falls  $x_i \neq x_j, i \neq j$ ; Darstellung ähnlich Lagrange-Interpolation möglich.)

## 3.4 Spline-Interpolation

Nachteil der Polynom-Interpolation:

Bei größerem  $n$  hat das Polynom  $g$  eventuell stärkere Schwankungen und kann eventuell noch stärker von  $f$  in einigen Zwischenpunkten abweichen als bei kleinerem Grad.

**Beispiel 3.5**



Alternative: auf Teilintervallen  $[x_{i-1}, x_i]$  mit Polynomen niedrigeren Grades arbeiten.

### 3.4.1 Kubische Splines

#### Definition 3.6

Unter einem kubischer Spline versteht man eine Funktion  $g(x) \in C^2[a, b]$  mit  $a = x_0 < x_1 < \dots < x_n = b$  Stützstellen die der Bedingung  $g(x_i) = f_i$  genügen, wobei  $g(x) \in \Pi_3$  auf  $[x_{i-1}, x_i] \forall i$ .

#### Betrachtung der Freiheiten/Bedingungen:

an den Stellen  $x_0 \dots x_n : g(x_i) = f_i$

an den Stellen  $x_1 \dots x_{n-1} : (n-1)$  Übergangsbedingungen:

$$\begin{aligned} g'(x_i - 0) &= g'(x_i + 0) \\ g''(x_i - 0) &= g''(x_i + 0) \end{aligned}$$

$g$  besteht aus  $n$  kubischen Polynomen:  $4n$  freie Parameter  
 $n$  mal rechte/linke Interpolationsbedingung  $g(x_i) = f_i$

$$\left. \begin{array}{l} 2n \\ +(2n-2) \text{ Übergangsbedingung} \end{array} \right\} 4n-2 \text{ Bedingungen}$$

Man darf noch 2 weitere Bedingungen stellen:

a)  $g''(x) = 0 \quad x = a, b$

b)  $g'(a) = m_a \quad g'(b) = m_b$  für gegebenes  $m_a, m_b$  (o.ä.)

Konstruktion: bezeichnen  $g''(x_i) = M_i, \quad i = 0, \dots, n$

setze  $h_i = x_i - x_{i-1} \quad i = 1, \dots, n$

auf  $[x_{i-1}, x_i]$  ist  $g(x) \in \Pi_3[x_{i-1}, x_i] \implies g''(x)$  linear

$$\begin{aligned} g''(x) &= M_{i-1} \frac{x_i - x}{h_i} + M_i \frac{x - x_{i-1}}{h_i} \quad \text{für } x \in [x_{i-1}, x_i] \\ g'(x) &= -\frac{M_{i-1}}{h_i} \frac{(x_i - x)^2}{2} + \frac{M_i}{h_i} \frac{(x - x_{i-1})^2}{2} + A_i \\ g(x) &= \frac{M_{i-1}}{h_i} \frac{(x_i - x)^3}{6} + \frac{M_i}{h_i} \frac{(x - x_{i-1})^3}{6} + A_i(x - x_{i-1}) + B_i \end{aligned}$$

mit gewissen Integrationskonstanten  $A_i, B_i$

Interpolationsforderungen:

$$g(x_i) = f_i, \quad g(x_{i-1}) = f_{i-1}$$

$$\begin{aligned} \implies f_{i-1} &= \frac{M_{i-1}h_i^2}{6} + B_i \\ f_i &= \frac{M_i h_i^2}{6} + A_i h_i + B_i \implies \\ B_i &= f_{i-1} - \frac{M_{i-1}h_i^2}{6} \\ A_i &= \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1}) \end{aligned}$$

Fazit: Jedes Polynom  $g(x) \in \Pi_3[x_{i-1}, x_i]$  ist linear von den Zahlen  $M_i$  abhängig. Bedingung  $g'(x_i - 0) = g'(x_i + 0)$  einarbeiten, um die Zahlen  $M_i$  zu bestimmen.

- für  $x \in [x_{i-1}, x_i]$ :

$$\begin{aligned} g'(x) &= \frac{-M_{i-1}}{2h_i}(x_i - x)^2 + \frac{M_i}{2h_i}(x - x_{i-1})^2 + A_i \\ g'(x_i - 0) &= \frac{M_i h_i}{2} + \frac{f_i - f_{i-1}}{h_i} - \frac{M_i h_i}{6} + \frac{M_{i-1} h_i}{6} \\ &= d_i + \frac{h_i}{6} M_{i-1} + \frac{h_i}{3} M_i \end{aligned}$$

mit

$$d_i = \frac{f_i - f_{i-1}}{h_i}$$

- für  $x \in [x_i, x_{i+1}]$

$$\begin{aligned} g'(x_i + 0) &= -\frac{M_i h_{i+1}}{2} + A_{i+1} \\ &= d_{i+1} - \frac{h_{i+1}}{3} M_i - \frac{h_{i+1}}{6} M_{i+1} \end{aligned}$$

$\implies$  also:  $g'(x_i - 0) = g'(x_i + 0) \iff$

$$\frac{h_i}{6} M_{i-1} + \frac{h_i + h_{i+1}}{3} M_i + \frac{h_{i+1}}{6} M_{i+1} = d_{i+1} - d_i \quad i = 1, \dots, n-1$$

$n-1$  Gleichungen für  $(n+1)$  Unbekannte  $M_i$  noch 2 frei wählbare Bedingungen hinzunehmen z. B.

$$\begin{aligned} g''(a) &= M_0 = 0 \\ g''(b) &= M_n = 0 \end{aligned} \implies (n-1) \text{ Gleichungen mit } (n-1) \text{ Unbekannten}$$

Gleichungssystem:

$$\begin{bmatrix} a_1 & b_2 & & & \\ c_2 & \ddots & \ddots & & \\ & \ddots & \ddots & b_{n-1} & \\ & & c_{n-1} & a_{n-1} & \end{bmatrix} \begin{bmatrix} M_1 \\ \vdots \\ \vdots \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} d_2 - d_1 \\ d_3 - d_2 \\ \vdots \\ d_n - d_{n-1} \end{bmatrix}$$

mit  $a_i = \frac{h_i + h_{i+1}}{3}$ ,  $b_i = \frac{h_i}{6} = c_i = \frac{h_i}{6}$ .

Diese Matrix ist stets streng diagonal dominant und somit regulär  
 $\implies M_1 \dots M_{n-1}$  stabil berechenbar.

Konvergenzverhalten:

Anders als bei Polynominterpolation konvergiert der Spline gegen  $f$  wenn  $n \rightarrow \infty$

### 3.4.2 B-Splines \*\*

**Definition 3.7** (*sehr allgemein*)

Seien  $M \in \mathbb{N}_0$  und  $u_0 < u_1 \leq \dots \leq u_{K-1} < u_K$ . Der Knotenvektor  $u = (u_j)_{j=0, \dots, K}$  definiert eine Zerlegung des Intervalls  $[u_0, u_K]$  in halboffene Intervalle  $[v_i, v_{i+1})$  ( $i = 0, \dots, \tilde{K} - 1$ ), wobei  $\{u_0, \dots, u_K\} = \{v_0, \dots, v_{\tilde{K}}\}$  mit  $v_0 < \dots < v_{\tilde{K}}$  und jedes der  $v_i$  genau  $l_i$ -mal in  $(u_j)_{j=0, \dots, K}$  vorkommt. Es gelte  $l_i \leq M + 1$  für alle  $1 \leq i \leq \tilde{K} - 1$ .

Der Splineraum  $S_{u, M}$  ist dann definiert durch

$$S_{u, M} := \{f : [u_0, u_K] \rightarrow \mathbb{R} : \text{Es existieren Polynome } p_0, \dots, p_{\tilde{K}-1} \\ \text{Grad} \leq M \text{ mit } f(x) = p_i(x) \text{ für } x \in [v_i, v_{i+1}) \text{ und für } M - l_i \geq 0 \\ \text{ist an der Stelle } v_i \text{ } M - l_i \text{-mal stetig differenzierbar}\}$$

Dabei ist mit 0-maliger Differenzierbarkeit die Stetigkeit gemeint.  
 $u$  wird als Knotenvektor,  $M$  als Grad des Splineraumes  $S_{u, M}$  und  $l_i$  als Vielfachheit des Knotens  $v_i$  bezeichnet.

**Definition 3.8**

$$B_{j,0,u}(x) = \begin{cases} 1 & \text{falls } u_j \leq x \leq u_{j+1} \\ 0 & \text{falls } x < u_j \text{ oder } x \geq u_{j+1} \end{cases}$$

für  $j = -M, -M + 1, \dots, K + M - 1, x \in \mathbb{R}$ , und

$$B_{i,l+1,u}(x) := \frac{x - u_j}{u_{j+l+1} - u_j} B_{j,l,u}(x) + \frac{u_{j+l+2} - x}{u_{j+l+2} - u_{j+1}} B_{j+1,l,u}(x)$$

für  $j = -M, -M + 1, \dots, K + M - l - 2, l = 0, 1, \dots, M - 1, x \in \mathbb{R}$ ,

**Definition 3.9**

Sei  $f : X \rightarrow \bar{\mathbb{R}}$  eine vorgegebene Funktion. Dann heißt die abgeschlossene Hülle der Menge  $\{x \in X \mid f(x) \neq 0\}$  der Träger der Funktion und man schreibt

$$\text{tr } f = \overline{\{x \in X \mid f(x) \neq 0\}}$$

**Bemerkung 3.10**

B-Splines haben einen endlichen Träger.

Durch Definition 3.8 sind die B-Splines  $B_{j,M,u}$  auf ganz  $\mathbb{R}$  definiert.

Durch Linearkombinationen der B-Splines entsteht dann ein Raum von auf ganz  $\mathbb{R}$  definierten Funktionen, welcher mit  $S_{u,M}$  bezeichnet wird.

$$S_{u,M} := \left\{ \sum_{j=-M}^{K-1} a_j B_{j,M,u} : a_j \in \mathbb{R} \forall j \right\}$$

**3.5 Alternative Interpolation**

- Rationale Interpolation

$$\mathcal{M} = \left\{ \text{rationale Funktionen } \frac{p_n(x)}{q_m(x)} \right\}$$

schwieriger zu handhaben

Vorteile bei Extrapolation: Pole exakt (gesucht Näherung  $f(\bar{x}) : \bar{x} \notin [a, b]$ )

- Trigonometrische Interpolation

$$\mathcal{M} = \text{span} (1, e^{ix}, e^{2ix}, \dots, e^{(n-1)ix})$$

für eine auf  $[0, 2\pi)$  periodische Funktion  $f$ ;

häufig wird auch  $x_k = \frac{k \cdot 2\pi i}{n}$  sein, so führt dieses Problem auf sehr einfache

**Lösung:**

setzen:  $\omega = e^{\frac{2\pi i}{n}}$  erste  $n$ -te Einheitswurzel

$$\omega(x) = e^{ix} \implies \mathcal{M} = \text{span} (\omega(x)^0, \omega^1(x), \dots, \omega^{n-1}(x))$$

Interpolation:

$$\begin{aligned} f_k &= \sum_{j=0}^{n-1} \beta_j \omega^j(x_k) \quad \forall k = 0, \dots, n-1 && \text{ges.: } \beta_j \\ \omega(x_k) &= e^{\frac{2\pi ki}{n}} = \omega^k \\ \implies f_k &= \sum_{j=0}^{n-1} \beta_j \omega^{kj} \quad \forall k = 0, \dots, n-1 \\ Fb &= f \quad b = (\beta_0 \dots \beta_{n-1})^T \\ f &= (f_0 \dots f_{n-1})^T \\ F &= (\omega^{jk})_{j,k=0}^{n-1} \quad \text{eventuell komplex} \end{aligned}$$

- a) symmetrisch aber komplex  
 b)  $F^*F = \bar{F}F = nI$  also  $(\frac{1}{\sqrt{n}}F)$  ist unitär.

bzw.  $F^*F = nI$   $F^{-1} = \frac{1}{n}F^* = \frac{1}{n}\bar{F}$  also Gleichungssystem  $Fb = f$  stets lösbar und  $b = \frac{1}{n}\bar{F}f$

$$\text{also } \beta_k = \frac{1}{n} \sum_{j=0}^{n-1} \bar{\omega}^{kj} f_j$$

$$\bar{\omega}^{kj} = e^{-\frac{kj2\pi i}{n}} = \cos\left(\frac{2\pi kj}{n}\right) - i \sin\left(\frac{2\pi kj}{n}\right)$$

d.h. es sind Summen der Art

$$\sum f_j \cos\left(\frac{2\pi kj}{n}\right) \text{ bzw. } \sum f_j \sin\left(\frac{2\pi kj}{n}\right)$$

notwendig  $\rightarrow$  übliche Summation aufwendig  $\mathcal{O}(n^2)$  und instabil

besonderer Algorithmus FFT „Fast Fourier Transformation“, schnelle Fourier Transformation beachtet die vielen Rekursionen dieser Zahlen

$$\omega^{jk} \rightarrow \begin{array}{l} 1. \text{ schnell dh. nur } \mathcal{O}(n \ln n) \text{ Operationen} \\ 2. \text{ stabil} \end{array}$$

- stets volle Summation  $Ff$  oder  $\bar{F}f$  zugleich,  $n$  muß eine Zweierpotenz sein (Verallg. auf anderen Zahlen möglich, aber nicht so effektiv)  
 COOLEY-TUKEY-Algorithmus

### 3.6 Approximation

- a) geg.: Funktion  $f$  (in der Regel stetig)  
 b) geg.:  $\mathcal{M}$  Funktionsmenge  
 (in der Regel  $\text{span}(\varphi_0(x), \dots, \varphi_n(x))$ )  
 c) Gütekriterium: eine Norm  $\|f - g\|$

Aufgabe: Finde  $g_f \in \mathcal{M}$  so daß  $\|f - g_f\| = \inf_{g \in \mathcal{M}} \|f - g\|$

$$\boxed{g_f \text{ heißt „beste Approximation“ von } f \text{ bezüglich } \|\cdot\|}$$

#### Bemerkung 3.11

betrachte nur lineare Approximationsprobleme wo

$$\mathcal{M} = \text{span}(\varphi_1 \dots \varphi_n) \text{ also } g = \sum \alpha_i \varphi_i \forall g \in \mathcal{M}$$

(auch  $\mathcal{M}$  mit nichtlinearen Parametern  $\alpha_j$  denkbar)

Normen:

$\|f(x)\|_{p,\omega} = \left(\int_a^b \omega(x)|f(x)|^p dx\right)^{\frac{1}{p}}$  werden benutzt (schreiben  $\|f\|_p = \|f\|_{p,1}$ )

bei  $\|\cdot\|_\infty = \max_{x \in [a,b]} |f(x)|$  „gleichmäßige Approximation“

bei  $\|\cdot\|_{2,\omega(x)}$  „Approximation im Mittel“

### 3.6.1 Approximation im Mittel

Betrachte

$$\langle f(x), g(x) \rangle_\omega = \int_a^b \omega(x) f(x) \bar{g}(x) dx$$

hierzu gehört die Norm  $\|f\|_{2,\omega} = \langle f, f \rangle_\omega^{\frac{1}{2}}$

$\omega(x)$  sei auf  $[a, b]$  erklärt und positiv (außer in endlich vielen Punkten)  $C[a, b]$  wird mit  $\langle \cdot, \cdot \rangle_\omega$  zum unitären Raum und kann zum Hilbertraum  $H_\omega$  vervollständigt werden.

Betrachte Raum  $\mathcal{M} = \text{span}(\varphi_0(x), \dots, \varphi_n(x)) \subset H_\omega$ , so kann die Basis stets als Orthonormalsystem (ONS) gewählt werden:

$$\langle \varphi_i(x), \varphi_j(x) \rangle_\omega = \delta_{ij} \cdot N_j \quad (\text{ONS} : \frac{\varphi_i}{\sqrt{N_i}})$$

#### Beispiel 3.12

a)  $\mathcal{M} = \Pi_n$  (Polynomapproximation)

$[a, b]$	$\omega(x)$	$\varphi_k(t)$	$N_k^{-1}$	Name
$[-1, 1]$	1	$\frac{1}{2^k k!} \frac{d^k}{dt^k} (t^2 - 1)^k$	$k + \frac{1}{2}$	LEGENDRE-Polynome
$[-1, 1]$	$(1 - t^2)^{-\frac{1}{2}}$	$\cos(k \arccos t)$	$\frac{1}{n}, \frac{2}{n} (k > 0)$	TSCHEBYSCHEW-Polynome
$[0, \infty)$	$e^{-t}$	$\sum_{j=0}^k \binom{k}{j} \frac{(-t)^j}{j!}$	1	LAGUERRE-Polynome
$(-\infty, \infty)$	$e^{-\frac{t^2}{2}}$	$(-1)^k e^{\frac{t^2}{2}} \frac{d^k}{dt^k} e^{-\frac{t^2}{2}}$	$\frac{1}{k! \sqrt{2\pi}}$	HERMITE-Polynome

b)  $\mathcal{M} = \text{span}(1, e^{\pm ix}, e^{\pm 2ix}, \dots, e^{\pm nix})$  ( $\omega(x) = 1$ , auf  $[-\pi, \pi]$ )  
 $N_k = 2\pi$

FOURIER-approximation bzw. im reellen:

$$\mathcal{M} = \text{span}(1, \sin x, \cos 2x, \dots, \cos 2x, \dots, \cos nx, \sin x, \sin 2x, \dots, \sin nx)$$

mit  $N_0 = 2\pi$   $N_k = \pi (k > 0)$

**Berechnung der besten Approximation**  $g_f = \sum_{i=0}^n \alpha_i \varphi_i(x)$

$$\begin{aligned} \|f - g\|^2 &= \|f\|^2 + \|g\|^2 - 2\langle f, g \rangle \\ &= \|f\|^2 + \sum |\alpha_i|^2 N_i - 2 \sum \alpha_i \langle f, \varphi_i \rangle \end{aligned}$$

$$\begin{aligned} D &= \text{diag}(N_0 \dots N_n) : a^T D a - 2b^T a + c \rightarrow \min_a a \in \mathbb{R}^{n+1} \\ a &= (\alpha_0 \dots \alpha_n)^T \quad \updownarrow \\ b &= (\langle f, \varphi_i \rangle)_{i=0}^n \quad D a = b \\ \alpha_i &= \frac{b_i}{N_i} = \frac{\langle f, \varphi_i \rangle}{N_i} \end{aligned}$$

**Satz 3.13**

Gilt  $\langle \varphi_i, \varphi_j \rangle = N_i \delta_{ij}$  für die Basis in  $\mathcal{M}$ , so ist

$$g_f = \sum_{i=0}^n \frac{\langle f, \varphi_i \rangle \varphi_i}{N_i}$$

die beste Approximation von  $f$  in  $\mathcal{M}$  bezüglich  $\|\cdot\| = \langle \cdot, \cdot \rangle^{\frac{1}{2}}$

**Bemerkung 3.14**

Die  $\frac{\langle f, \varphi_i \rangle}{\sqrt{N_i}}$  sind sogenannte FOURIER-Koeffizienten bezüglich der Basis  $\{\varphi_0 \dots \varphi_n\}$  (aus Verallgemeinerung von trigonometrischen Funktionen (Bsp. b))

$$g_f = \alpha_0 + \sum_{k=1}^n (\alpha_k \cos kt + \tilde{\alpha}_k \sin kt)$$

Abbruch der FOURIER-Reihe nach Gliedern  $\cos nt / \sin nt$ .

**Satz 3.15**

Aus  $\|f - g_f\|^2 \rightarrow \min_{g \in \mathcal{M}}$  folgt Orthogonalität des Fehlers zu  $\mathcal{M}$  :

$$\langle f - g_f, g \rangle = 0 \quad \forall g \in \mathcal{M}$$

**Beweis:**

$$\langle f - \sum \alpha_i \varphi_i, \varphi_k \rangle = \alpha_k N_k - \alpha_k N_k = 0 \quad \forall k \implies \text{für jede Funktion aus } \mathcal{M}$$

Abschätzen des Fehlers / Konvergenz gegen Null

$$\begin{aligned}\|f - g_f\|^2 &= \langle f - g_f, f \rangle - \langle f - g_f, g_f \rangle \\ &= \|f\|^2 - \langle g_f, f \rangle \geq 0 \\ \langle g_f, f \rangle &= \sum_{i=0}^n \frac{\langle f, \varphi_i \rangle^2}{N_i} = \sum_{i=0}^n \langle f, \frac{\varphi_i}{\sqrt{N_i}} \rangle^2 \leq \|f\|^2\end{aligned}$$

$n \rightarrow \infty$

BESSELSche Ungleichung

- a)  $\lim_{n \rightarrow \infty} \sum \langle f, \frac{\varphi_i}{\sqrt{N_i}} \rangle^2 = \|f\|^2$  für beliebige Funktion  $f$   
 $\implies$  Fehler geht gegen Null  $\forall f$   
 ein solches Basissystem  $\{\varphi_i\}_{i=0}^{\infty}$  heißt fundamental
- b)  $\lim \sum \langle f, \frac{\varphi_i}{\sqrt{N_i}} \rangle^2 \leq \|f\|^2 - d^2 \quad d > 0$   
 für eine Funktion  $f \implies f$  hat einen „Abstand“  $d$  vom Unterraum  $\text{span}(\varphi_0, \varphi_1, \dots)$

Viele orthogonale Funktionensysteme sind fundamental für  $f \in C[a, b]$ , so z. B.

- trigonometrische Funktion ( $\omega = 1, [-\pi, \pi]$ )
- LEGENDRE-Polynome ( $\omega = 1, [-1, 1]$ )
- TSCHEBYSCHEW-Polynome ( $\omega = (1 - t^2)^{-\frac{1}{2}}, [-1, 1]$ )

### 3.6.2 Gleichmässige oder Tschebyschew-Approximation

hier:  $\|f\|_{\infty} = \max_{x \in [a, b]} |f(x)|$

also:  $\|f - g_f\|_{\infty} \rightarrow \min_{g \in \mathcal{M}}$  heißt kleinstmögliche Abweichung der Funktion  $g$  von  $f$   
 (d.h. nur BANACH-Raum  $C[a, b]$ ,  $\|\cdot\|_{\infty}$  deshalb kein Skalarprodukt möglich).

Lösung  $g_f$  ist i.a. nicht explizit angebar, sondern muß iterativ bestimmt werden (mittels REMES-Algorithmus)

Ausnahme:

$f(x) = x^n, \mathcal{M} = \Pi_{n-1}[-1, 1]$

(also  $x^n$  durch Polynom höchstens  $(n - 1)$ -Grades approximieren)

Lösung:  $x^n - g_f(x)$  ist Tschebyschew-Polynom,  $n$ -ten Grades

$$x^n - g(x) = T_n(x)/2^{n-1}$$

# Kapitel 4

## Nichtlineare Gleichungen

### 4.1 Allgemeine Aufgabenstellung

A) geg.: Funktion  $f_k$  von  $n$  Veränderlichen  $x_1 \dots x_n$

ges.:  $f_k(x_1, \dots, x_n) = 0 \quad k = 1 \dots n$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad F = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \quad \text{wobei} \quad F(x) = 0$$

B) Verallgemeinerung: Anzahl der Funktionen  $\neq$  Anzahl der Unbekannten

$$\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix} = F, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad m > n$$

im allgemeinen  $F(x) = 0$  nicht lösbar aber  $\min_x \|F(x)\|$  möglich.

Sonderfall:  $n = 1 \quad f(x) = 0$  Nullstellen-Bestimmung der reellen Funktion  $f$

### 4.2 Nullstellenbestimmung mit Interpolationsideen

Idee:

nehmen einige Stützstellen  $x_0 \dots x_k$

berechnen  $f(x_0) \dots f(x_k)$

Konstruieren Interpolationspolynom  $g(x)$

→ neue Näherung für Nullstellen  $x^*$  ist Nullstelle  $x_{k+1}$  von  $g(x)$

**1. Sekantenverfahren** (*regula falsi*) $n = 1, g(x) \in \Pi_1:$ 

$$\text{Lagrange-Interpolation } g(x) = f_0 \frac{x - x_1}{x_0 - x_1} + f_1 \frac{x - x_0}{x_1 - x_0}$$

$$\text{Nullstelle } x_2 = \frac{f_0 x_1 - f_1 x_0}{f_0 - f_1}$$

Verfahren:

$$x_{k+1} = \frac{f(x_{k-1})x_k - f(x_k)x_{k-1}}{f(x_{k-1}) - f(x_k)}$$

günstig wenn:  $x_{k-1} < x^* < x_k$  ( $f(x_k) \cdot f(x_{k-1}) < 0$ ) giltDiejenige Stützstelle aus  $x_{k-1}, x_k$  jetzt auswählen um damit wieder $f(\tilde{x}_k) \cdot f(x_{k+1}) < 0$  zu erreichen.**2. Newton-Verfahren:**  $n = 1, g(x)$  aus Hermite Interpolation  $g \in \Pi_1$ 

$$x_0, f(x_0), f'(x_0) \implies g(x) = f(x_0) + (x - x_0)f'(x_0)$$

$$\text{Nullstelle } x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Verfahren:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

**3. Interpolation der Umkehrfunktion**

hier beliebig hoher Polynomgrad möglich:

geg.:  $x_0, \dots, x_k$  Stützstellen $y_0, \dots, y_k$  Funktionswerte  $y_j = f(x_j)$ Sei  $f$  monoton  $\implies \exists$  Umkehrfunktion  $x_j = f^{-1}(y_j)$  $g(x) \in \Pi_k$  Lagrange-Interpolationspolynom für  $f^{-1}$ Nutzen: $f(x^*) = 0 \iff x^* = f^{-1}(0)$  also nicht Nullstellen von  $g(x)$  benötigt, sondern

$$x^* = f^{-1}(0) \approx g(0) = x_{k+1}$$

#### 4. Andere Verfahren

$$n = 1, g \in \Pi_2$$

Hermite-Interpolationspolynom  $\rightarrow$  Newtonverfahren 2. Grades

$$n = 2, g \in \Pi_2$$

Lagrange-Interpolation  $\rightarrow$  MULLERsches Parallelverfahren

Verfahren mit noch höheren Polynomen sind ungebräuchlich weil neue Nullstelle schwer zu berechnen.

#### Bemerkung 4.1

$k = 1$  ( $x_0, x_1, y_0, y_1$  benutzt) ergibt  $x_2$  wie bei Sekantenverfahren, von nun an bessere Näherung, da stets alle  $x_i$  benutzt; vorteilhaft wird Newton-Interpolationspolynom benutzt

### 4.3 Das Newton-Verfahren für $n > 1$

(Verallgemeinerung obiger Form)

$$F = \begin{bmatrix} f_1(x_1 \dots x_n) \\ \vdots \\ f_k(x_1 \dots x_n) \end{bmatrix} \quad F(x) = 0$$

$f_k(x)$  hat Taylorentwicklung:

$$f(x) = f_k(x^{(j)}) + \nabla f_k^T(x - x^{(j)}) + \mathcal{O}(\|x - x^{(j)}\|^2)$$

$$\nabla f_k = \left( \frac{\partial f_k}{\partial x_1}, \dots, \frac{\partial f_k}{\partial x_n} \right)^T$$

Für den ganzen Vektor  $F$  aufgeschrieben:

$$F(x) = F(x^{(j)}) + \begin{bmatrix} \nabla f_1^T \\ \vdots \\ \nabla f_n^T \end{bmatrix} (x - x^{(j)}) + \mathcal{O}(\|x - x^{(j)}\|^2)$$

$$F'(x) = \left( \frac{\partial f_k}{\partial x_j} \right)_{k,j=1}^n \implies x - x^{(j)} = +(F'(x))^{-1} F(x) + \mathcal{O}(\|x - x^{(j)}\|^2)$$

Näherung von  $x^* \rightarrow x^{(k+1)}$ ,  $x \rightarrow x^{(k)}$  weglassen des Restgliedes

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

Verfahren:

$$\begin{array}{l} F(x^{(k)}) = (F'(x^{(k)}))(\Delta x^{(k)}) \\ x^{(k+1)} = x^{(k)} - \Delta x^{(k)} \end{array}$$

### Bemerkung 4.2

- lokale Konvergenz, falls  $F'(x^k) \forall k$  regulär
- Für jedes  $k$  ist ein Gleichungssystem zu lösen und die Systemmatrix ändert sich mit  $k \Rightarrow$  teuer

**verbessertes Newton Verfahren:**

$$\begin{array}{l} F'(x^0)(\Delta x^{(k)}) = -F(x^{(k)}) \\ x^{(k+1)} = x^{(k)} + \Delta x^{(k)} \end{array}$$

### Bemerkung 4.3

Jetzt liegt ein Gleichungssystem mit konstanter Systemmatrix vor.  
Diese sollte LU faktorisiert werden.

## 4.4 Fixpunktiteration

$f(x^*) = 0$  wird in iterierfähige Form gebracht  $\iff x^* = \varphi(x^*)$  wobei  $x^*$  Fixpunkt von  $\varphi$  heißt

Verfahren:

$$x_{k+1} := \varphi(x_k)$$

Konvergenzbetrachtung

### Definition 4.4

Eine Folge  $\{x_k\}$  mit Grenzwert  $x^*$  hat die Konvergenzordnung  $p$ , wenn

$$|x_{k+1} - x^*| \leq C|x_k - x^*|^p$$

gilt. (bei  $p = 1$  ist  $C < 1$  notwendig,  $p = 1$  heißt lineare Konvergenz).

### Satz 4.5

Gilt in einer Umgebung  $U_\varepsilon(x^*) : |\varphi'(x)| \leq q < 1$ , so konvergiert die Fixpunkt-Iteration für jeden Startwert  $x_0 \in U_\varepsilon(x^*)$  mindestens linear gegen  $x^*$ .

**Beweis:**

$$x_0 \in U_\varepsilon(x^*)$$

$$\begin{aligned} x_1 - x^* &= \varphi(x_0) - \varphi(x^*) \\ &= (x_0 - x^*)\varphi'(\xi) \end{aligned}$$

 $\xi$  zwischen  $x_0$  und  $x^* \implies \xi \in U_\varepsilon(x^*)$ 

$$|\varphi'(\xi)| \leq q < 1 \implies$$

$$|x_1 - x^*| \leq q|x_0 - x^*|$$

usw.

$$|x_k - x^*| \leq q^k|x_0 - x^*| \rightarrow 0$$

nach obiger Definition liegt lineare Konvergenz vor, wenn

$$|x_{k+1} - x^*| \leq q|x_k - x^*| \quad \forall k \quad q \text{ fest, } q < 1$$

gilt.

□

**Satz 4.6**Ist  $\varphi(x)$  in der Umgebung  $U_\varepsilon(x^*)$  genügend oft differenzierbar, mit

$$\begin{aligned} \varphi^{(k)}(x^*) &= 0 \quad k = 1, \dots, p-1 \\ \varphi^{(p)}(x^*) &\neq 0 \end{aligned}$$

so konvergiert die Fixpunktiteration mit der Ordnung  $p$  (falls überhaupt Konvergenz vorliegt).**Beweis:**

$$\begin{aligned} x_{k+1} - x^* &= \varphi(x_k) - \varphi(x^*) \\ &= \varphi^{(p)}(x^*) \frac{(x_k - x^*)^p}{p!} + \varphi^{(p+1)}(\xi) \frac{(x_k - x^*)^{p+1}}{(p+1)!} \\ \frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} &= \left| \frac{\varphi^{(p)}(x^*)}{p!} + \frac{\varphi^{(p+1)}(\xi)}{(p+1)!} (x_k - x^*) \right| \\ &\quad k \rightarrow \infty \end{aligned}$$

□

**Beispiel 4.7**

$$\begin{aligned} x - \cos x = 0 \quad x_{k+1} &= \cos x_k \\ \varphi(x) &= \cos x \\ \varphi'(x) &= -\sin x \\ \sin x^* &< 1 \text{ lineare Konvergenz.} \end{aligned}$$

**Beispiel 4.8**

Newtonverfahren mit  $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$

$$\varphi(x) = x - \frac{f(x)}{f'(x)} \quad \text{Sonderfall einer Fixpunktiteration}$$

$$\varphi'(x) = 1 - \frac{(f'(x))^2 - f''f}{(f')^2} = \frac{f''f}{(f')^2} = 0$$

$$\begin{aligned} \varphi''(x) &= \frac{(f'''f + f''f')(f')^2 - 2f'(f'')^2f}{(f')^4} \\ &= \frac{f'f'''f + f''(f')^2 - 2f^2f}{(f')^3} \\ &= \frac{f''(x)}{f'(x)} \neq 0 \end{aligned}$$

$\Rightarrow$  Ist  $f'(x^*) \neq 0$ , so konvergiert das Newtonverfahren mindestens quadratisch gegen  $x^*$  (nach Satz 4.6).

**4.5 Fixpunktiteration für  $n > 1$** 

$x = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ ,  $F(x) = 0$ ,  $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$   
betrachte eine nichtlineare Fixpunktgleichung

$$x^* = \Phi(x^*) \in \mathbb{R}^n$$

Iterationsverfahren:

$$\boxed{x^{(k+1)} = \Phi(x^{(k)})}$$

**Definition 4.9**

$\Phi$  heißt kontrahierende Abbildung, wenn

$$\|\Phi(x) - \Phi(y)\| \leq q\|x - y\| \quad \forall x, y \in U_\varepsilon(x^*) \quad \text{und} \quad q < 1$$

mit  $U_\varepsilon(x^*) = \{x \in \mathbb{R}^n : \|x - x^*\| < \varepsilon\}$ .

**Satz 4.10**

Ist  $\Phi$  in  $U_\varepsilon(x^*)$  kontrahierend, so konvergiert die Fixpunktiteration mindestens linear gegen  $x^*$ .

**Beweis** analog zu Satz 4.5

$$\begin{aligned} \text{lineare Konvergenz} & : \|x^{(k+1)} - x^*\| \leq q\|x^{(k)} - x^*\|, \quad q < 1 \\ \text{Konvergenzordnung } p & : \|x^{(k+1)} - x^*\| \leq C\|x^{(k)} - x^*\|^p \end{aligned}$$

**Bemerkung 4.11**

Wenn  $F(x) = Ax - b$  linear ist, so folgt aus der analogen Umformung aus Abschnitt 2

$$\begin{aligned} A &= C - N \\ F(x^*) &= 0 \iff x^* = Bx^* + c \\ \text{mit } B &= C^{-1}N \\ c &= C^{-1}b \\ \Phi(x) &= Bx + c \end{aligned}$$

$$\begin{aligned} &\Phi \text{ kontrahierend} \\ &\iff \\ &\|\Phi(x) - \Phi(y)\| \leq q\|x - y\| \\ &\|B(x - y)\| \leq q\|x - y\| \forall x, y \\ &\iff \\ &\|Bz\| \leq q\|z\| \quad \forall z \\ &\iff \\ &\|B\| \leq q < 1 \end{aligned}$$

Die Bedingung  $\|B\| \leq q < 1$  an Konvergenz des Iterationsverfahrens  $x^{(k+1)} = Bx^{(k)} + c$  ist ein Sonderfall von einer „kontrahierenden Abbildung“.

## 4.6 Besonderheiten bei Polynomgleichungen

Betrachte wieder den 1-dimensionalen Fall:

$$f(x^*) = 0 \quad x \in \mathbb{R} \text{ wobei } f(x) = \sum_{i=0}^n a_i x^i \text{ Polynom } n\text{-ten Grades.}$$

Möglichkeiten:

- Newtonverfahren
- Sekantenverfahren
- Parabelverfahren usw.

notwendig: effektive Berechnung von

$$f(x_k) = \sum_{i=0}^n a_i x_k^i$$

$\implies$  **Horner Schema** erfordert  $n$  wesentliche Operationen!

$$f(x) = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + a_n x))) \dots$$

einfacher Algorithmus

$$\begin{aligned} f &:= a_n \\ \text{for } i &= n - 1 \text{ down to } 0 \text{ do} \\ f &:= f * x_0 + a_i \end{aligned}$$

**Bemerkung 4.12**

Alle angegebenen Iterationsverfahren haben nur lokale Konvergenz (d.h. Startwert  $x_0 \in U_\varepsilon(x^*)$ ), i.a. ist diese Startumgebung  $U_\varepsilon(x^0)$  unbekannt.

**Satz 4.13**

Hat das Polynom  $f \in \Pi_n$  nur reelle Nullstellen  $x_i^*$ , so konvergiert das Newton-Verfahren für alle reellen Startwerte  $x_0$  mit  $x_0 > \max_i x_i^*$  gegen die größte Nullstelle.

**Beweis:** (klar am Bild)  
Sei o.B.d.A.

$$\lim_{n \rightarrow \infty} = -\infty \quad \text{Sei } x_0 < \min_i x_i^* := x^*$$

man zeigt

$$\begin{aligned} (1) \quad &x_{n+1} > x_n \\ (2) \quad &x_{n+1} < x^* \end{aligned}$$

$\implies \{x_n\}$  ist monoton wachsende beschränkte Folge

$\implies \{x_n\} \rightarrow \tilde{x} \leq x^*$

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ \tilde{x} &= \tilde{x} - \frac{f(\tilde{x})}{f'(\tilde{x})} \end{aligned}$$

$\tilde{x} \leq x^* \implies$  ist Nullstelle  $x = x^*$

□

$$f(x) = (x - x_0)f_1(x)$$

Herleitung der Koeffizienten von  $f_1(x) = \sum_{i=0}^{n-1} b_i x^i$

$$\begin{aligned}
f(x) &= (x - x_0)f_1(x) + f(x_0) \\
&= (x - x_0) \sum_{i=0}^{n-1} b_i x^{i+1} - \sum_{i=0}^{n-1} b_i x^i x_0 + f(x_0) \\
\sum_{i=0}^n a_i x^i &= b_{n-1} x^n + \sum_{i=0}^{n-1} (b_{i-1} - b_i x_0) x^i - b_0 x_0 + f(x_0)
\end{aligned}$$

Koeffizientenvergleich ergibt

$$\begin{aligned}
x^n &: b_{n-1} &= a_n \\
x^i &: b_i &= b_{i-1} - b_i x_0 \quad i = 1, \dots, n-1 \\
&& b_{i-1} &= a_i + b_i x_0 \\
x^0 &: a_0 &= -b_0 x_0 + f(x_0) \\
&& f(x_0) &= a_0 + b_0 x_0
\end{aligned}$$

#### verbesserter Algorithmus

$$\begin{aligned}
b_{n-1} &= a_n \\
\text{for } i &= n-1 \text{ down to } 0 \text{ do} \\
b_{i-1} &:= b_i x_0 + a_i \implies b_{-1} = f(x_0) \\
f_1(x) &= \sum_{i=0}^{n-1} b_i x^i, \\
f(x) &= f_1(x)(x - x_0) + b_{-1}
\end{aligned}$$

#### **Bemerkung 4.14**

Aus dem Satz 4.13 folgt, daß eine Konstruktion einer kleinsten/größten Nullstelle  $x_0$  möglich ist. Weitere Nullstellen erhält man durch Betrachtung des reduzierten Polynoms. Dessen Koeffizienten erhält man nach dem HORNER - Schema.

#### **Bemerkung 4.15**

Konstruktion von  $x_0$  somit aus irgendeiner Abschätzung der Nullstellen von Polynomen möglich, z. B.

$$\begin{aligned}
|x_i^*| &\leq \max_{1 \leq i \leq n-1} \left\{ \left| \frac{a_n}{a_0} \right|, 1 + \left| \frac{a_i}{a_0} \right| \right\} \\
|x_i^*| &\leq \max \left\{ 1, \sum_{i=1}^n \left| \frac{a_i}{a_0} \right| \right\} \\
|x_i^*| &\leq \max \left\{ \left| \frac{a_n}{a_{n-1}} \right|, 2 \left| \frac{a_{n-1}}{a_{n-2}} \right|, \dots, 2 \left| \frac{a_1}{a_0} \right| \right\}
\end{aligned}$$

### Das Verfahren von BAIRSTOW

Bei reellem Startwert  $x_0$  sind alle iterierten  $x_k$  reell, somit auf diese Weise keine komplexen Nullstellen berechenbar.

#### Ausweg:

Iteration für (konjugiert komplexe) Paare von Nullstellen, also  $x^2 - rx - q$  ist Teiler von  $f(x) \iff f(x)$  hat die beiden Nullstellen von  $(x^2 - rx - q)$ .

#### **Ansatz:**

$$f(x) = f_1(x)(x^2 - rx - q) + \underbrace{Ax + b}_{\text{Rest}}$$

$$\implies \begin{cases} A(r, q) = 0 \\ B(r, q) = 0 \end{cases} \text{ Nichlineares Gleichungssystem mit 2 Unbekannten}$$

Rest bei Division von  $f(x)$  durch  $x^2 - rx - q$  soll Null werden.

Benutzung des Newtonverfahrens für 2 gekoppelte Gleichungen mit 2 Unbekannten:

$$\begin{bmatrix} r^{k+1} \\ q^{k+1} \end{bmatrix} = \begin{bmatrix} r^k \\ q^k \end{bmatrix} - D^{-1} \begin{bmatrix} A(r^k, q^k) \\ B(r^k, q^k) \end{bmatrix}$$

$$D = \begin{bmatrix} \frac{\partial A}{\partial r} & \frac{\partial A}{\partial q} \\ \frac{\partial B}{\partial r} & \frac{\partial B}{\partial q} \end{bmatrix} \Big|_{(r^k, q^k)}$$

Geg.:  $r = r^k, q = q^k$

Dann ist zur Bestimmung von  $r^{(k+1)}, q^{(k+1)}$  die Berechnung von  $A, B, A_{,r}, A_{,q}, B_{,r}, B_{,q}$  notwendig. Die Berechnung der Ableitungswerte ist mit einer weiteren Division von  $f_1(x)$  durch  $x^2 - rx - q$  möglich:

$$f_1(x) = f_2(x)(x^2 - rx - q) + \tilde{A}x + \tilde{B}$$

$$A_{,q} = \tilde{A}, B_{,q} = \tilde{B}, A_{,r} = r\tilde{A} + \tilde{B}, B_{,r} = q\tilde{A}$$

Die Division von  $f(x)$  (bzw.  $F_i(x)$ ) durch  $x^2 - rx - q$  wird besten mit einem Horner-ähnlichen Schema durchgeführt.

Herleitung:

$$\begin{aligned}
 f(x) = \sum_{i=1}^n a_i x^i &= (x^2 - rx - q)f_1(x) + Ax + B \\
 &= (x^2 - rx - q) \sum_{i=0}^{n-2} b_i x^i + Ax + B \\
 &= \sum_{i=0}^{n-2} b_i x^{i+2} - r \sum_{i=0}^{n-2} b_i x^{i+1} - q \sum_{i=0}^{n-2} b_i x^i + Ax + B \\
 &= \sum_{i=2}^n b_{i-2} x^{i+2} - r \sum_{i=0}^{n-1} b_{i-1} x^i - q \sum_{i=0}^{n-2} b_i x^i + Ax + B \\
 &= \sum_{i=2}^{n-2} (b_{i-2} - rb_{i-1} - qb_i) x^i + b_{n-2} x^n \\
 &\quad + (b_{n-3} - rb_{n-2}) x^{n-1} + (A - rb_1 - qb_1) x + (B - qb_0)
 \end{aligned}$$

Koeffizientenvergleich:

$$\begin{aligned}
 a_n &= b_{n-2} \quad i = n - 2, \dots, 2 \\
 a_{n-1} &= b_{n-3} - rb_{n-2} \\
 a_i &= b_{i-2} - rb_{i-1} - qb_i \\
 a_1 &= A - rb_0 - qb_1 \\
 a_0 &= B - qb_0
 \end{aligned}$$

# Kapitel 5

## Numerische Integration

### 5.1 Vorbemerkungen

**Ziel:** näherungsweise Berechnung von  $\int_a^b f(x)dx$  für möglichst beliebige Funktionen  $f$

**Bemerkung 5.1**

$f \in C[a, b]$ , so ist  $I(f) = \int_a^b f dx$  ein stetiges beschränktes lineares Funktional von  $f$  über  $C[a, b]$ .  $I(f)$  wird wie folgt durch stetige beschränkte lineare Funktionen approximiert:

$$(5.1) \quad Q_n(f) = \sum_{i=0}^n \omega_i f(x_i)$$

mit den reellen Integrationsgewichten  $\omega_i$  und den Stützstellen

$$x_0 = a < x_1 < \dots < x_n = b)$$

Man bezeichnet 5.1 als „Quadraturformel“

**Satz 5.2**

Mit wachsenden  $n$  werden die Koeffizienten  $\omega_i$  betragsmäßig immer größer. Andererseits gilt

$$b - a = \sum_{i=0}^n \omega_i$$

**Bemerkung 5.3**

$Q_n(f)$  benutzt nur endlich viele Funktionswerte  $\Rightarrow$  nicht alle Funktionen werden gleich gut integriert (von einer Quadraturformel)

**Ziel:** Konstruktion von Quadraturformeln

## 5.2 Newton-Cotes-Quadratformeln

Stützstellen  $a = x_0 < x_1 \cdots < x_n = b$  werden willkürlich festgelegt, ersetzen  $I(f)$  durch Integration über Lagrange - Interpolationspolynom

$$\begin{aligned} Q_n(f) &= \int_a^b \sum_{i=0}^n f(x_i) L_i(x) dx \\ &= \sum_{i=0}^n f(x_i) \int_a^b L_i(x) dx \\ \Rightarrow \omega_i &= \int_a^b L_i(x) dx \text{ Integrationsgewichte} \end{aligned}$$

### Bemerkung 5.4

Die Integrationsgewichte  $\omega_i$  hängen nur von  $x_i$  ab und nicht von  $f$  !!!!

Insbesondere: äquidistante Stützstellen

setzen

$$x = a + sh \quad \text{mit} \quad s \in [0, n]$$

$$dx = h ds$$

$$x_k = a + ih \quad \text{mit} \quad h = \frac{b-a}{n} \quad \text{dann folgt}$$

$$\begin{aligned} L_i(x) &= \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x_i - x_j}{x_i - x_j} = \prod_{j \neq i} \frac{a + sh - a - jh}{a + ih - a - jh} \\ &= \prod_{j \neq i} \frac{sh - jh}{ih - jh} = \prod_{j \neq i} \frac{s - j}{i - j} \end{aligned}$$

$$\omega_i = \int_0^n h \prod_{j \neq i} \frac{s - j}{i - j} ds.$$

(feste Zahlen, nur abhängig von n)

**Beispiel 5.5**

$$\underline{n = 1} \quad h = b - a$$

$$\omega_0 = \omega_1 = h \int_0^1 \frac{s-0}{1-0} ds = h \int_0^1 \frac{s-1}{-1} ds = \frac{h}{2}$$

$$Q_1(f) = \frac{b-a}{2} (f(x_0) + f(x_1)) \quad \text{Trapezregel}$$

$$\underline{n = 2} \quad h = \frac{b-a}{2}$$

$$\omega_0 = \omega_2 = \frac{b-a}{6} \quad \omega_1 = 2\frac{b-a}{3}$$

$$Q_2(f) = (b-a) \left( \frac{1}{6} f(a) + \frac{2}{3} f\left(\frac{a+b}{2}\right) + \frac{1}{6} f(b) \right) \quad \text{Simpson - Regel}$$

$$\underline{n = 3}$$

$$\omega_0 = \omega_3 = \frac{b-a}{8} \quad \omega_1 = \omega_2 = \frac{3}{8}(b-a) \quad \frac{3}{8} - \text{Regel}$$

$$\underline{n = 4}$$

$$\omega_0 = \omega_4 = \frac{7}{90}(b-a) \quad \omega_1 = \omega_3 = \frac{16}{45}(b-a)$$

$$\omega_2 = \frac{6}{45}(b-a)$$

usw. bis  $n=6$  danach werden einige der Integrationsgewichte  $\omega_i$  negativ, was zur Instabilität führt

**Bemerkung 5.6**

aus der Konstruktion folgt, daß

$$Q_n(f) = \int_a^b f(x) dx \text{ gilt, wenn } f \text{ ein Polynom vom Grade } \leq n \text{ ist.}$$

Spezialfall: bei  $f(x) \equiv 1 \in \Pi_n \quad \int_a^b f dx = b - a$

$$Q_n(f) = \sum_{i=0}^n \omega_i = b - a \quad \text{Kontrolle}$$

**Bemerkung 5.7**

Definiere noch  $Q_0(f) = (b-a)f\left(\frac{a+b}{2}\right)$  als Mittelpunktsregel

**Fehlerfunktional**

$$R_n(f) = I(f) - Q_n(f)$$

ergibt sich aus der Fehlerbetrachtung der Polynom-Interpolation bei der Funktion  $f$ , die  $(n + 1)$ -mal stetig differenzierbar sei.

$$\begin{aligned}
 R_n(f) &= \int_a^b (f(x) - P_n(x)) dx \\
 &= \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi(x)) \prod_{i=0}^n (x - x_i) dx \\
 &\quad x = a + sh \\
 &= \frac{h^{n+2}}{(n+1)!} \int_0^n f^{(n+1)}(\xi(s)) \prod_{i=0}^n (s - i) ds \\
 &\quad (P_n \text{ das Lagrange-Interpolationspolynom von } f)
 \end{aligned}$$

### Beispiel 5.8

$$\begin{aligned}
 n = 1 \quad |R_1(f)| &= \frac{h^3}{2!} \left| \int_0^1 f''(\xi(s)) s(s-1) ds \right| \\
 \text{MWS Integralrechnung : } |R_1(f)| &= \frac{h^3}{2} |f''(\xi_0)| \int_0^1 s(s-1) ds \\
 \Rightarrow |R_1(f)| &\leq \frac{(b-a)^3}{12} \max_{x \in [a,b]} |f''(x)|
 \end{aligned}$$

### Beispiel 5.9 Fehler bei

$$\begin{aligned}
 \text{Trapezregel: } f \in C^2[a, b] &\rightarrow |R_1(f)| \leq \frac{h^3}{12} \|f''\|_\infty \quad h = (b-a) \\
 \text{Simpson-Regel: } f \in C^4[a, b] &\rightarrow |R_2(f)| \leq \frac{h^5}{90} \|f^{(4)}\|_\infty \quad h = \frac{(b-a)}{2} \\
 \frac{3}{8} \text{ Regel: } f \in C^4[a, b] &\rightarrow |R_3(f)| \leq \frac{3h^5}{80} \|f^{(4)}\|_\infty \quad h = \frac{(b-a)}{3}
 \end{aligned}$$

### Bemerkung 5.10

$$\begin{aligned}
 Q_0 &\text{ integriert exakt falls } f \in \Pi_1 \\
 Q_1 &\text{ integriert exakt falls } f \in \Pi_1 \\
 Q_2 &\text{ integriert exakt falls } f \in \Pi_3 \\
 Q_3 &\text{ integriert exakt falls } f \in \Pi_3
 \end{aligned}$$

### Bemerkung 5.11

$n \rightarrow \infty$  i.a. keine Konvergenz  $Q_n(f)$  gegen  $I(f)$

### 5.3 Zusammengesetzte Newton-Cotes-Formeln

Ausweg für höhere Genauigkeit:

$$\text{unterteilen } [a, b] = \bigcup_{i=1}^n [x_{i-1}, x_i]$$

und benutzen auf jeden Teilintervalle eine Newton-Cotes-Formel mit niedrigeren Grad.

#### Beispiel 5.12

Zusammengesetzte Trapezregel:

$$Q_n^{(1)}(f) = \frac{b-a}{2n} (f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n))$$

$$x_n = a + kh$$

ergibt den Fehler

$$\begin{aligned} |R_n^{(1)}(f)| &= |I(f) - Q_n^{(1)}(f)| \leq \frac{1}{12} \left(\frac{b-a}{n}\right)^3 \left| \sum_i f''(\xi_i) \right| \\ &\leq \frac{(b-a)^3}{12} \frac{1}{n^2} \|f''\|_\infty \longrightarrow 0 \quad \text{mit } n \longrightarrow \infty \end{aligned}$$

#### Beispiel 5.13

Zusammengesetzte (verallgemeinerte) SIMPSON-Regel:

Sei  $n$  gerade,  $h = \frac{b-a}{n}$ , auf  $[x_0, x_2], [x_2, x_4], \dots, [x_{n-2}, x_n]$  wird

SIMPSON-Regel angewandt:

$$Q_n^{(2)}(f) = \frac{b-a}{3n} (f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n))$$

Fehlerfunktional

$$\begin{aligned} |R_n^{(2)}(f)| &\leq \sum \left(\frac{b-a}{n}\right)^5 \frac{1}{90} |f^{(4)}(\xi_k)| \\ &\leq \frac{n}{2} \|f^{(4)}\|_\infty \frac{(b-a)^5}{n^5} \cdot \frac{1}{90} = \frac{(b-a)^5}{n^4 \cdot 180} \|f^{(4)}\|_\infty \end{aligned}$$

## 5.4 ROMBERG-Integration

betrachte stets äquidistante Stützstellen  $x_k = a + kh$  ( $h = \frac{b-a}{n}$ ) so ist zusammengesetzte Trapezregel (oder zusammengesetzte SIMPSON-Regel) eine Folge  $\{Q_n^{(1)}(f)\}_{n=1}^\infty$  von Näherungen für  $I(f)$ . In der Berechnung gibt es eine besonders günstige

Teilfolge:

$$\left\{ Q_1^{(1)}, Q_2^{(1)}, Q_4^{(1)}, Q_8^{(1)} \dots \right\}$$

da jeweils nur  $2^{k-1}$  neue Stützstellen hinzukommen.

### Algorithmus 5.14

$$Q_{m/2}^{(1)}(f) = \frac{b-a}{n} (f(x_0) + 2f(x_2) + 2f(x_4) + \dots + f(x_{m/2}))$$

$Q_n^{(1)}(f) = \frac{1}{2} Q_{n/2}^{(1)}(f) + \frac{b-a}{n} \sum_{k \text{ unger.}} f(x_k)$
<p>Start : <math>Q_1^{(1)}(f) = Q_1(f) = \frac{b-a}{2} (f(a) + f(b))</math></p>

#### Weiter Verbesserung:

aus der Folge  $\{Q_{2^k}^{(1)}\}_{k \geq 0}$  wird durch sogenannte RICHARDSON - Extrapolation eine schneller konvergente Folge gebildet:

$$Q_n^{(2)} = Q_n^{(1)} + \frac{1}{3} (Q_n^{(1)} - Q_{n/2}^{(1)})$$

ist identisch mit Zusammengesetzter SIMPSON-Regel (somit schneller konvergent), Fortsetzung diese Prozesses:

$Q_n^{(k+1)} = Q_n^{(k)} + \frac{1}{4^k - 1} (Q_n^{(k)} - Q_{n/2}^{(k)}) \quad n = 1, 2, 4, 6 \dots,$
---

ROMBERG-Quadratur

#### Schema:

$Q_1^{(1)}$						
	$\ddots$					
$Q_2^{(1)}$	$\dots$	$Q_2^{(2)}$				
	$\ddots$		$\ddots$			
$Q_4^{(1)}$	$\dots$	$Q_4^{(2)}$	$\dots$	$Q_4^{(3)}$		
	$\ddots$		$\ddots$	$\ddots$		
$Q_8^{(1)}$	$\dots$	$Q_8^{(2)}$	$\dots$	$Q_8^{(3)}$	$\dots$	$Q_8^{(4)}$
$\downarrow$		$\downarrow$		$\downarrow$		$\downarrow$
$I(f)$		$I(f)$		$I(f)$		$I(f)$

**Bemerkung 5.15**

Die Konvergenzgeschwindigkeit nimmt mit steigenden  $n$  und auch steigenden  $k$  zu.

**5.5 Gauß-Integration**

Wir wissen das für Newton-Cotes-Formel gilt:

$$Q_n(f) = I(f) \quad \forall f \in \Pi_n$$

dabei waren Stützstellen  $\{x_i\}$  fest vorgegeben.

jetzt: Stützstellen  $x_0 \cdots x_n$  zusätzlich frei wählbar, so daß

$$Q_n^G(f) = \sum_{i=0}^n \omega_i f(x_i) = I(f) \quad \forall f \in \Pi_{2n+1} \quad (2n+2 \text{ Freiheitsgrade})$$

**Beispiel 5.16**

$$n=1 \rightarrow Q_1^G(f) = \omega_0 f(x_0) + \omega_1 f(x_2)$$

betrachte  $[a, b] = [-1, 1]$  soll exakt für Polynom bis 3. Grad sein:

$$\begin{aligned} \Pi_0: \quad \omega_0 + \omega_1 &= 2 &= \int_{-1}^1 dx \\ \Pi_1: \quad \omega_0 x_0 + \omega_1 x_1 &= 0 &= \int_{-1}^1 x dx \quad \text{da Funktion ungerade} \\ \Pi_2: \quad \omega_0 x_0^2 + \omega_1 x_1^2 &= \frac{2}{3} &= \int_{-1}^1 x^2 dx \\ \Pi_3: \quad \omega_0 x_0^3 + \omega_1 x_1^3 &= 0 &= \int_{-1}^1 x^3 dx \end{aligned}$$

4 nichtlineare Gleichungen mit 4 Unbekannten

$$\text{Lösung:} \quad \omega_0 = \omega_1 \quad x_0 = x_1 = \frac{1}{3}\sqrt{3}$$

Gaußsche Quadratformel:  $n = 1; [a, b] = [-1, 1]$

$$\boxed{Q_1^G(f) = f\left(-\frac{1}{3}\sqrt{3}\right) + f\left(\frac{1}{3}\sqrt{3}\right)}$$

(Größere Genauigkeit als SIMPSON-Regel mit 3 Stützstellen.)

$n > 1$  Nichtlineares Gleichungssystem schwer lösbar, anderer Weg:

$\{x_i\}$  sind Nullstellen orthogonaler Polynome betrachte Gewichtsfunktion  $\omega(x)$

mit Skalarprodukt wie früher  $\langle f, g \rangle = \int_a^b \omega(x) f(x) g(x) dx$

jetzt näherungsweise Berechnung von  $I(f) = \int_a^b \omega(x) f(x) dx$  durch

Gauß-Integrationsformel

$$Q_n^G(f) = \sum_{i=0}^n \omega_i f(x_i)$$

wobei wir  $\omega_i$  und  $x_i$  suchen so daß,

$$I(f(x)) = Q_n^G(f(x)) \quad \forall f \in \Pi_{2n+1}$$

**Vorbemerkung:** Sind  $x_i$  bestimmt, dann ist

$$\omega_k = Q_n^G(L_k(x)) = I(L_k(x))$$

mit  $L_k$  die Lagrangepolynome über die Stützstellen  $x_i$

**Satz 5.17**

Mit  $w(x) = \prod_{i=0}^n (x - x_i)$  gilt:

$$Q_n^G(f) \text{ ist exakt} \iff \int_a^b \omega(x) w(x) z(x) dx = 0$$

für alle  $f \in \Pi_{2n+1}$                       für alle  $z(x) \in \Pi_n$

**Beweis**

(a) „  $\implies$ “     $w(x) \in \Pi_{n+1}, z(x) \in \Pi_n : \quad w(x)z(x) \in \Pi_{2n+1}$

$$\begin{aligned} \implies Q_n^G(w(x)z(x)) &= I(w(x)z(x)) \\ \sum \omega_i w(x_i) z(x_i) &= 0 \quad \forall z(x) \end{aligned}$$

(b) „  $\impliedby$ “     $f(x) \in \Pi_{2n+1}$  beliebig

$$\begin{aligned} f(x) &= w(x)z(x) + y(x) \\ w &\in \Pi_{n+1}, z \in \Pi_n : y \in \Pi_n \text{ Rest} \end{aligned}$$

$$I(f) = I(w \cdot z + y) = I(w \cdot z) + I(y)$$

$$I(f) = I(y)$$

$y \in \Pi_n, y = p_n(x)$  Lagrange auf  $(n+1)$  Stützstellen  
auf  $\{x_0 \cdots x_n\}$

$$I(f) = I(y) = Q_n^G(y) = \sum \omega_i y(x_i)$$

$$\sum \omega_i f(x_i) = Q_n^G(f)$$

□

**Bemerkung 5.18**

Also Konstruktion  $x_0, \dots, x_n$  durch andere Interpretation des Satzes:  
 $w(x)$  Polynom  $(n+1)$ -Grades mit Nullstelle  $x_0 \cdots x_n$  und ist orthogonal zu allen Polynomen  $n$ -ten Grades.

also: Sind  $p_0, p_1, \dots, p_{n+1}$  die orthogonalen Polynome zum Gewicht  $\omega(x)$  und zu  $[a, b]$ , so sind  $\{x_0, \dots, x_n\}$  als Nullstellen des Polynoms  $p_{n+1}$  zu wählen.

**Beispiel 5.19**

$[a, b] = [-1, 1]$ ,  $\omega(x) \equiv 1$  LEGENDRE-Polynome (mit  $c = \text{const}$ )

$$\left( p_k(x) = c \cdot \frac{d^k}{dx^k} (x^2 - 1)^k \right)$$

$n = 0$ :  $x_0 = 0$ ,  $\omega_0 = 2 \longrightarrow$  MITTELpunktregel

$n = 1$ :  $p_2(x) = ((x^2 - 1)^2)'' = (2(x^2 - 1)2x)' = (4x^3 - 4x)' = 12x^2 - 4$

$$x^2 - \frac{1}{3} = 0 \quad x_0 = \pm \frac{1}{3}\sqrt{3}$$

$$-x_0 = x_1 = \frac{1}{3}\sqrt{3} \quad \omega_0 = \omega_1 = 1$$

$n = 2$ :  $-x_0^2 x_2 = \sqrt{\frac{3}{5}} x_1 = 0$

$$\omega_0 = \omega_2 = \frac{5}{9} \quad \omega_1 = \frac{8}{9}$$

usw. (tabellarisch erfaßt)

Fehlerfunktional bei Gauß - Integration

Ist  $f \in C^{2n}[a, b]$ , so gilt

$$\int_a^b \omega(x) f(x) dx - Q_n^G(f) = \frac{f^{(2n)}(\xi)}{(2n)!} \langle p_n, p_n \rangle_\omega$$

wobei  $p_n$  das  $n$ -te orthogonale Polynom ist bzgl.  $\langle \cdot, \cdot \rangle_\omega$  mit höchsten Koeffizient 1 ( $p_n^2 x^n + \dots$ )

Fehler  $\rightarrow 0$  für  $n \rightarrow \infty$

(aber Nachteil: ständig neue Stützstellen)

## Nullstellen Orthogonaler Polynome

## Beispiel 5.20

$$\omega(x) = (1 - x^2)^{-\frac{1}{2}} \quad [a, b] = [-1, 1]$$

bei diesen Tschebyscheff-Polynomen sind Nullstellen bekannt

$$p_{n+1}(x) = \cos(n+1)(\arccos x) = 0 \quad \forall |x| < 1$$

$$(n+1) \arccos x_i = \frac{\pi}{2} + k\pi$$

$$x_k = \cos\left(\frac{\pi}{2n+2} + \frac{k\pi}{n+1}\right) = \cos\left(\frac{2k+1}{2n+2}\pi\right)$$

$$k = 0 \dots n$$

Die Gewichte sind hierbei unabhängig von Stützstellen:  $\omega_k = \frac{\pi}{n+1} \forall k$   
allgemeiner Fall:

Nullstellen  $x_i$  des Polynoms  $p_{n+1}^{(x)}$  sind stabil als Eigenwerte einer Tridiagonalmatrix berechenbar.

**Satz 5.21**

Die orthogonalen Polynome haben folgende 3-gliedrige Rekursionsformel:

$$p_{-1}(x) = 0$$

$$p_0(x) = 1$$

$$p_{n+1}(x) = (x - \alpha_n)p_n(x) - \beta_n^2 p_{n-1}(x) \quad n = 0, 1, \dots$$

**Beweis:** Zeigen, daß  $\langle p_i, p_j \rangle = 0 \quad \forall i \neq j$  dabei wird die Wahl von  $\alpha_n, \beta_n$  mit angegeben.

1.  $n=0$

$$p_1(x) = (x - \alpha_0)p_0(x) = xp_0(x) - \alpha_0 p_0(x)$$

$$\langle p_1, p_0 \rangle = 0 = \langle xp_0, p_0 \rangle - \alpha_0 \langle p_0, p_0 \rangle$$

$$\text{also } \alpha_0 = \frac{\langle xp_0, p_0 \rangle}{\langle p_0, p_0 \rangle}$$

2.  $n > 0$

$$\text{sei } \langle p_i, p_j \rangle = 0 \quad i \neq j, i, j \leq n$$

$$p_{n+1}(x) = xp_n(x) - \alpha_n p_n(x) - \beta_n^2 p_{n-1}(x)$$

$$\alpha_n = \langle xp_n(x), p_n(x) \rangle / \gamma_n^2, \quad \gamma_n^2 = \langle p_n(x), p_n(x) \rangle$$

$$\langle p_{n+1}, p_{n-1} \rangle = 0 = \langle xp_n, p_{n-1} \rangle - \beta_n^2 \gamma_{n-1}^2$$

$$\begin{aligned}
\beta_n^2 &= \langle p_{n-1}, xp_n \rangle / \gamma_{n-1}^2 \\
&= \langle p_n, xp_{n-1} \rangle / \gamma_{n-1}^2 \\
&= \langle p_n, p_n + \alpha_{n-1}p_{n-1} + \beta_{n-1}p_{n-2} \rangle / \gamma_{n-1}^2 \\
\beta_n^2 &= \gamma_n^2 / \gamma_{n-1}^2 \\
j &< n - 1 \\
\langle p_{n+1}, p_j \rangle &= 0 \quad \text{automatisch, da:} \\
&= \langle xp_n, p_j \rangle - \langle \alpha_n p_n, p_j \rangle - \beta_n^2 \langle p_{n-1}, p_j \rangle \\
&= \langle p_n, xp_j \rangle \\
\langle p_n, xp_j \rangle &= \langle p_n, p_{j+1} + \alpha_j p_j + \beta_j^2 p_{j-1} \rangle = 0 \\
&\quad (\text{wegen: } j < n - 1 \text{ heißt } j + 1 < n)
\end{aligned}$$

**Bemerkung 5.22**

Wichtigste Voraussetzung ist Symmetrie des Operators  $\mathcal{A}p(x) = xp(x)$   
bzgl.  $\langle p, q \rangle = \int \omega(x)p(x)q(x)dx$

$$\langle \mathcal{A}p, q \rangle = \langle p, \mathcal{A}q \rangle$$

**Satz 5.23**

Die Tridiagonalmatrix

$$\tilde{T} = \begin{bmatrix} \alpha_0 & \beta_1^2 & & & \\ 1 & \ddots & \ddots & & \beta_n^2 \\ & \ddots & \ddots & & \\ & & & 1 & \alpha_n \end{bmatrix} \quad \text{bzw. } T = \begin{bmatrix} \alpha_0 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_n & \\ & & & \beta_n & \alpha_n \end{bmatrix}$$

hat die Eigenwerte  $x_0, \dots, x_n$  (Nullstellen von  $p_{n+1}$ )

**Beweis:** Definiere  $q_{k+1}(x) = \det(xI - T_k) \in \Pi_{k+1}$  für alle  $k = 0, 1, \dots, n$

$$T_k = \begin{bmatrix} \alpha_0 & \beta_1^2 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & & \beta_k^2 \\ & & & 1 & \alpha_k \end{bmatrix}$$

Eigenwerte von  $T_k = \tilde{T}$  sind Nullstellen von  $q_{k+1}(x)$

zeigen:  $q_k = p_k \quad \forall k \quad q_0 = p_0 = 1$

$$q_1 = x - \alpha_0 = p_1$$

$q_{k+1} = \det(xI - T_k)$  entwickelt nach letzter Zeile und dann nach letzter Spalte ergibt gleiche Rekursion wie für  $p_{k+1}$ .

Mit  $T = D\tilde{T}D^{-1}$

$$D = \text{diag} (1, \beta_1, \beta_1\beta_2, \beta_1\beta_2\beta_3, \dots)$$

haben also  $T$  und  $\tilde{T}$  gleiche Eigenwerte.

$T$  symmetrische Tridiagonalmatrix, davon besonders stabil Eigenwerte berechenbar.

Gewinnung der  $(\alpha_n, \beta_n)$  i.a. ähnlich schwierig

□

**Beispiel 5.24** *Tschebyscheff-Polynom:*

$$\begin{aligned} T_0 &= 1 & T_1(x) &= x \\ T_{k+1} &= 2xT_k - T_{k-1} \end{aligned}$$

$$\boxed{p_n(x) = \frac{1}{2^{n-1}} T_n(x)}$$

$$\begin{aligned} p_{k+1}(x) &= \frac{1}{2^k} T_{k+1} = \frac{1}{2^k} (2xT_k - T_{k-1}) \\ &= \frac{1}{2^k} (2x2^{k-1}p_k(x) - 2^{k-2}p_{k-1}(x)) \\ &= xp_k(x) - \frac{1}{4}p_{k-1}(x) \\ \rightarrow \alpha_n &= 0 \forall k, \quad \beta_n^2 = \frac{1}{4} \forall k \end{aligned}$$

## 5.6 Gauß-Integration in Mehrdimensionalen

Die Verallgemeinerung der „Mittelpunktsregel“ im Mehrdimensionalen

$$f(\vec{x}) \quad \text{skalare Funktion von Ort } \vec{X} \in \mathbb{R}^d \quad (d = 1, 2, 3)$$

gesuchte Näherung für

$$I(f) = \int_{\Omega} f(\vec{x}) d\Omega \quad \Omega \subset \mathbb{R}^d \quad \text{Gebiet}$$

$d = 2$	Flächenelement
$d = 3$	Volumenelement

einfachster Fall: 1-Punkt Gauß - Integration

$$\begin{aligned} I(f) &\approx Q_1^G(f) = \omega f(\vec{x}_0) = \text{mes}\Omega f(\vec{x}_0) \\ \text{mes}\Omega &= \int_{\Omega} d\Omega \quad \text{und ist } \vec{x}_0 \text{ der Schwerpunkt von } \Omega \\ &\rightarrow Q_1^G \text{ exakt für lineare Funktion } f \end{aligned}$$

$$\vec{x}_0 = \frac{1}{\text{mes } \Omega} \int_{\Omega} \vec{x} d\Omega$$

bessere Genauigkeit:

a)  $\Omega$  achsenparalleles Rechteck ( $d = 2$ )  $\Omega = [a, b] \times [c, d]$

$$\int_{\Omega} f(x) d\Omega = \int_a^b \int_c^d f(x, y) dx dy$$

→ zwei eindimensionale Integrale mit Gaußintegration

b)  $d = 2$ ,  $\Omega$  Dreieck ( $d = 3$ ,  $\Omega$  Tetraeder)

$$I(f) \approx \sum_{i=0}^n \omega_i f(\vec{x}_i)$$

unter Benutzung der Gaußpunkte  $\vec{x}_i \in \Omega$

übliche Vorgehensweise:

Transformation des gegebenen Dreiecks auf sogenanntes Masterdreieck  $\hat{\Omega}$

$$\begin{aligned} \vec{x} &= A\hat{x} + b \\ \text{Eckpunkte } x_E^{(1)} &= A \begin{pmatrix} 0 \\ 0 \end{pmatrix} + b \\ x_E^{(2)} &= A \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \\ x_E^{(3)} &= A \begin{pmatrix} 0 \\ 1 \end{pmatrix} + b \\ \text{mit } A &= \begin{bmatrix} x_E^{(2)} - x_E^{(1)} & x_E^{(3)} - x_E^{(1)} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \int_{\Omega} f(\vec{x}) &= \int_{\hat{\Omega}} f(A\hat{x} + b) \cdot |det A| d\hat{\Omega} \\ &= \int_0^1 \int_0^{1-\hat{x}_1} \tilde{f}(\hat{x}_1, \hat{x}_2) d\hat{x}_2 d\hat{x}_1 \\ &\approx \sum_{i=0}^n \omega_i \tilde{f}(\hat{x}^{(i)}) \end{aligned}$$

dabei sind die Gaußpunkte  $\hat{x}^{(i)} \in \hat{\Omega}$  und Gewichte  $\omega_i$  tabelliert zu finden

**Beispiel 5.25** ( $n=0$ )

$$\hat{x}^{(0)} = \left(\frac{1}{3}\right) \omega_0 = \frac{1}{2} \quad \text{exakt für lineare Funktionen}$$

**Beispiel 5.26** ( $n=1$ )

$$\hat{x}^{(i)} \text{ Seitenmitten } \left(0, \frac{1}{2}\right)^\top \quad \left(\frac{1}{2}, 0\right)^\top \quad \left(\frac{1}{2}, \frac{1}{2}\right)^\top \quad \omega_i = \frac{1}{6}$$

**oder**

$$\hat{x}^{(1)} = \left(\frac{1}{6}, \frac{1}{6}\right)^\top \quad \hat{x}^{(2)} = \left(\frac{4}{6}, \frac{1}{6}\right)^\top \quad \hat{x}^{(3)} = \left(\frac{1}{6}, \frac{4}{6}\right)^\top \quad \omega_i = \frac{1}{6}$$

# Kapitel 6

## Numerische Differentiation

geg.:  $f(x)$  Stelle  $\bar{x}$   
ges.:  $f'(\bar{x})$  näherungsweise  
insbesondere:  $f$  nur aus diskreten Werten  $x_0 < x_1 < \dots < x_n$  (äquidistant)  
mit  $f_k = f(x_k)$  gegeben

Wichtigster Spezialfall:

$x_k = \bar{x}$  benutzen hierzu Daten in der Umgebung von  $x_k$  ( $x_{k-1}$  und  $x_{k+1}$ )  
und setzen  $x_{k+1} - x_k = h = x_k - x_{k-1}$

**Vorwärtige Differenz:**

$$\boxed{\frac{f(x_{k+1}) - f(x_k)}{h} = f'(x_k) + \mathcal{O}(h)}$$

da nach Taylor:  $f(x_k + h) = f(x_k) + hf'(x_k) + \frac{h^2}{2}f''(x_k) + \dots$

**rückwärtige Differenz**

$$\boxed{\frac{f(x_k) - f(x_{k-1}))}{h} = f'(x_k) + \mathcal{O}(h)}$$

da nach Taylor:  $f(x_k - h) = f(x_k) - hf'(x_k) + \frac{h^2}{2}f''(x_k) + \mathcal{O}(h^3)$

**zentrale Differenz:**

$$\boxed{\frac{f(x_{k+1}) - f(x_{k-1}))}{2h} = f'(x_k) + \mathcal{O}(h^2)}$$

da nach Taylor:  $f(x_k + h) - f(x_k - h) = 2hf'(x_k) + \mathcal{O}(h^3)$

**Approximation von  $f''(x_k)$ :**

1. zentrale Differenz  $\left|_{x_k + \frac{h}{2}} f'(x_k + \frac{h}{2}) = \frac{f(x_{k+1}) - f(x_k)}{h} + \mathcal{O}(h^2)\right.$

2. zentrale Differenz  $\left|_{x_k - \frac{h}{2}} \quad f'(x_k - \frac{h}{2}) = \frac{f(x_k) - f(x_{k-1}))}{h} + \mathcal{O}(h^2)\right.$

3. zentrale Differenz  $\left|_{x_k} \quad f''(x_k) = \frac{f'(x_k + \frac{h}{2}) - f'(x_k - \frac{h}{2}))}{h} + \mathcal{O}(h^2)\right.$

einsetzen ergibt  $f''(x_k) = \frac{\frac{f(x_{k+1}) - f(x_k)}{h} - \frac{f(x_k) - f(x_{k-1}))}{h}}{h}$

$$\boxed{f''(x_k) = \frac{-2f(x_k) + f(x_{k+1}) + f(x_{k-1}))}{h^2}} + \mathcal{O}(h^2)$$

# Kapitel 7

## Anfangswertaufgaben für gewöhnliche Differentialgleichungen

Betrachten hier Differentialgleichungen von einer Veränderlichen (meist Zeit  $t$ )

Ges.: Vektorfunktion  $y(t_0) \in \mathbb{R}^n$

Geg.: Anfangswert  $y(t_0)$       üblich  $t_0 = 0$

allgemein  $\dot{y}(t) = \frac{d}{dt}y(t) = f(t, y(t))$  Vektorfunktion

Spezialfall: lineare DGL  $\dot{y} + Ay = g(t)$       mit  $A \in \mathbb{R}^{n,n}$

**Anwendungen:**

- Mehrkörperdynamik/Bewegung von Körpern in Kraftfeldern
- Entwicklung von Populativen
- chemische Reaktion
- bei elektrischen Schaltkreisen

allgemeine Differentialgleichung der Ordnung  $K$ :

$$y^{(K)} = f(t, y, \dot{y}, \dots, y^{(k-1)}) \quad y^{(j)} = \frac{d^j}{dt^j}y$$

können durch Substitution  $z_j(t) = y^{(j-1)}(t)$   $j = 1, \dots, K$  auf größeres System 1. Ordnung zurückgeführt werden:

$$\begin{aligned} \dot{z}_k &= f(t, z_1, z_2, \dots, z_k) \\ \dot{z}_{k-1} &= z_k \\ \dot{z}_{k-2} &= z_{k-1} \\ &\vdots \\ \dot{z}_1 &= z_2 \end{aligned} \quad \Leftrightarrow \quad \dot{z} = \hat{f}(t, z) \quad z \in \mathbb{R}^{k \cdot n}$$

$$y = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix} \in \mathbb{R}^{kn} \quad \dot{y} = F(t, y)$$

Deshalb ist Betrachtung von  $\dot{y}(t) = f(t, y(t))$  ausreichend.

Beachte: i.a.  $y, f \in \mathbb{R}^n$  (Spezialfall  $n = 1$  enthalten)

Vorgehen: gesuchte Funktion  $y(t) \forall t > 0$  (bei gegebenen  $y(0)$ )

numerisch: bestimmen  $y^{(k)} \approx y(t_k)$  für diskrete Zeitpunkte  $0 < t_1 < t_2 \dots$  und  $y^{(0)} = y(0)$

## 7.1 Einfachste Grundideen / einfachste Verfahren

$y^{(0)} = y(0)$  gegeben

Sei  $y^{(k)} \approx y(t_k)$  bestimmt

bezeichne  $t_{k+1} = t_k + \tau$  mit  $\tau$  als Zeitschrittweite

ges.:  $y^{(k+1)} \approx y(t_k + \tau)$  unter Benutzung von  $y^{(k)}$  und  $\tau$  ( $\rightarrow$  Einschrittformel)

$$\dot{y} = f(t, y(t))$$

**1. Möglichkeit:** Approximation  $\dot{y}(t)$ : Sei  $t = t_k$

a) betrachte  $t = t_k$  und benutzen vorwärtige Differenz

$$\text{Approximation } \frac{y^{(k+1)} - y^{(k)}}{\tau} = f(t_k, y^{(k)})$$

$$\implies \text{Verfahren } \boxed{y^{(k+1)} = y^{(k)} + \tau f(t_k, y^{(k)})}$$

explizit Euler (alt: Eulersches Polygonzugverfahren)

b) betrachte  $t = t_k + \tau$  und benutzen rückwärtige Differenz

$$\begin{aligned} \dot{y}(t_k + \tau) &= f(t_k + \tau, y(t_k + \tau)) \\ &\downarrow \text{Approximation} \\ \frac{y^{(k+1)} - y^{(k)}}{\tau} &= f(t_{k+1}, y^{(k+1)}) \end{aligned}$$

$$\implies \text{Verfahren } \boxed{y^{(k+1)} = y^{(k)} + \tau f(t_{k+1}, y^{(k+1)})}$$

implizit Euler

c) betrachten  $t = t_k + \frac{\tau}{2}$ , und benutzen zentrale Differenz

$$\begin{aligned} \dot{y}(t_k + \frac{\tau}{2}) &= (f_k + \frac{\tau}{2}, y(t_k + \frac{\tau}{2})) \\ &\downarrow \text{Approximation} \\ \frac{y^{(k)} - y^{(k)}}{\tau} &\approx f(t_k + \frac{\tau}{2}, y(t_k + \frac{\tau}{2})) \\ y(t_k + \frac{\tau}{2}) &\approx \frac{y^{(k+1)} + y^{(k)}}{2} \end{aligned}$$

$$\implies \text{Verfahren } \boxed{y^{(k+1)} = y^{(k)} + \tau f(t_k + \frac{\tau}{2}, \frac{1}{2}(y^{(k)} + y^{(k+1)}))}$$

d) analog zu c) aber  $y(t_k + \frac{\tau}{2})$  mit explizit Euler

$$\implies \text{Verfahren } \boxed{y^{(k+1)} = y^{(k)} + \tau f(t_k + \frac{\tau}{2}, y^{(k)} + \frac{\tau}{2} f(t_k, y^{(k)}))}$$

explizit, verbessertes (modifiziertes) Polygonzugverfahren

andere Schreibweise:

$$\left| \begin{array}{l} k^{(1)} = f(t_k, y^{(k)}) \\ k^{(2)} = f(t_k + \frac{\tau}{2}, y^{(k)} + \frac{\tau}{2} k^{(1)}) \\ y^{(k+1)} = y^{(k)} + \tau k^{(2)} \end{array} \right. \quad \text{Predictor-Korrektor-Schritt}$$

## 2. Möglichkeit:

$\dot{y}(t) = f(t, y(t)) \quad t = t_k$  bei gegeben  $y^{(k)}$

$$\int_{t_k}^{t_{k+1}} \dot{y}(t) dt = y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t)) dt$$

Approximation des Integrals mit Integrationsformeln

$y(t_{k+1})$  mittels Newton-Cotes Formeln

a) Rechteckregel  $\int_a^b g(x) dx \approx (b-a)g(\xi)$  wählen  $\xi = t_k$

$$y^{(k+1)} = y^{(k)} + \tau f(t_k, y^{(k)})$$

a) explizit EULER

b) Rechteckregel mit  $\xi = t_{k+1}$

$$y^{(k+1)} = y^{(k)} + \tau f(t_{k+1}, y^{(k+1)})$$

implizit EULER

c) Mittelpunktsregel  $\xi = \frac{a+b}{2}$

$$y^{(k+1)} = y^{(k)} + \tau f\left(t_k + \frac{\tau}{2}, y\left(t_k + \frac{\tau}{2}\right)\right)$$

wieder wie vorher  $y^{(k+1)} = y^{(k)} + \tau f\left(t_k + \frac{\tau}{2}, \frac{1}{2}(y^{(k)} + y^{(k+1)})\right)$

d) Trapezregel

$$y^{(k+1)} = y^{(k)} + \frac{\tau}{2}(f(t_k, y^{(k)}) + f(t_k + \tau, y^{(k+1)}))$$

entweder implizit so, oder wieder Prediktor

$$\begin{aligned} k^{(1)} &= f(t_k, y^{(k)}) \\ k^{(2)} &= f(t_k + \tau, y^{(k)} + \tau k^{(1)}) \\ y^{(k+1)} &= y^{(k)} + \frac{\tau}{2}(k^{(1)} + k^{(2)}) \end{aligned} \quad \text{Prediktor-Korrektor-Verfahren}$$

EULER-HEUN-Verfahren

usw. mit weiteren Newton-Cotes-Formeln entstehen viele weitere Möglichkeiten

⇒ Bewertung dieser Verfahren? allg. Konstruktionsprinzip?

## 7.2 Konsistenz

alle expliziten Formeln lassen sich als

$$\begin{cases} k := g(t_k, y^{(k)}, \tau) \\ y^{(k+1)} = y^{(k)} + \tau k \end{cases} \quad (1)$$

schreiben mit einer gewissen „Zuwachsfunktion“  $g(t_k, y^{(k)}, \tau)$

**Beispiel 7.1**

a) explizit EULER:  $g(t, y, \tau) = f(t, y)$

d) modifiziertes Polygonzugverfahren:  $g(t, y, \tau) = k^{(2)} = f(t + \frac{\tau}{2}, y + \frac{\tau}{2}k^{(1)})$

e) EULER-HEUN:

$$\begin{aligned} g(t, y, \tau) &= \frac{1}{2}(k^{(1)} + k^{(2)}) \\ &= \frac{\tau}{2}(f(t, y) + f(t + \tau, y + \tau f(t, y))) \end{aligned}$$

**Definition 7.2** (Konsistenz)

Die explizite Vorschrift (1) heißt konsistent, wenn  $g(t, y, \tau)$  definiert ist  $\forall 0 < \tau < \tau_{\max} \forall (t, y) \in D_f(f(t, y))$  und wenn

$$\lim_{\tau \rightarrow 0} g(t, y, \tau) = f(t, y)$$

**Beispiel 7.3**

bei a), d), e) offenbar erfüllt

Die Güte der Konsistenz beschreibt die Konsistenzordnung betrachten entsprechend (1) die Funktion von  $(t, y, \tau)$

$$R(y) = \frac{y(t+\tau) - y(t)}{\tau} - g(t, y, \tau)$$

durch Taylorentwicklung an der Stelle  $t$  in  $y(t + \tau)$  und in  $g(t, y, \tau)$  ergibt sich für  $r(y) = \tau^p \cdot d(t) + \mathcal{O}(\tau^{p+1})$  beliebiger Vektor unabhängig von  $\tau$ .

**Definition 7.4**

Das Verfahren zur Lösung von  $\dot{y} = f(t, y)$  hat die Konsistenzordnung  $p$ , wenn die Taylorentwicklung für  $R(y)$  an der Stelle  $t$  als niedrigsten Term

$$\tau^p d(t)$$

ergibt.

**Beispiel 7.5**

a)

$$\begin{aligned} R(y) &= \frac{y(t + \tau) - y(t)}{\tau} - f(t, y) \\ &= \frac{y(t) + \tau \dot{y} + \tau^2 \frac{1}{2} \ddot{y} - y(t) - f(t, y)}{\tau} \\ &= \dot{y}(t) + \frac{\tau}{2} \ddot{y}(t) + \dots - f(t, y) \\ &= \frac{\tau}{2} \ddot{y}(t) + \mathcal{O}(\tau^2) \quad p = 1 \end{aligned}$$

e) EULER-HEUN

$$\frac{y(t+\tau) - y(t)}{\tau} = \dot{y}(t) + \frac{\tau}{2}\ddot{y}(t) + \mathcal{O}(\tau^2) \quad (*)$$

$$\begin{aligned} g(t, y, \tau) &= \frac{1}{2}(f(t, y) + f(t + \tau, y + \tau f(t, y))) \\ &= \frac{1}{2}(f + f + \tau \underbrace{f_t}_{\in \mathbb{R}^n} + \tau \underbrace{f_y}_{(n \times n)} \underbrace{f}_{\in \mathbb{R}^n}) + \mathcal{O}(\tau^2) \quad (**) \end{aligned}$$

$$\begin{aligned} (*) - (**) \quad \ddot{y} &= \frac{d}{dt}\dot{y} = \frac{d}{dt}f(t, y) = f_t - f_y f \\ R(y) &= (\dot{y} - f) + \frac{\tau}{2}(\ddot{y} - f_t - f_y f) + \mathcal{O}(\tau^2) \\ &\text{Konsistenzordnung } p = 2 \end{aligned}$$

analog für d)

### 7.3 Allgemeine Form: explizite Runge -Kutta - Verfahren

Herleitung von Verfahren höherer Konsistenzordnung nicht über Quadraturformeln, sondern mit Taylorabgleich  $R(y) = \tau^p d(y) + \dots$

allgemeine Form

$$\text{Prediktor} \left\{ \begin{array}{l} k^{(1)} = f(t_k, y^{(k)}) \\ k^{(2)} = f(t_k + a_2\tau, y^{(k)} + b_{21}\tau k^{(1)}) \\ k^{(3)} = f(t_k + a_3\tau, y^{(k)} + (b_{31}k^{(1)} + b_{32}k^{(2)})) \\ \vdots \\ k^{(s)} = f(t_k + a_s\tau, y^{(k)} + \sum_{i < s} b_{si}k^{(i)}) \end{array} \right.$$

schließlich

$$\text{Korrektor} \left\{ \begin{array}{l} k = g(t_k, y^{(k)}, \tau) = \sum_{i=1}^s c_i k^{(i)} \\ (\text{also: } y^{(k+1)} = y^{(k)} + \tau \sum_{i=1}^s c_i k^{(i)}) \end{array} \right.$$

d.h. das jeweilige Verfahren ist durch die Konstanten

$$\begin{array}{c|cccc} 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_{21} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_s & b_{s1} & \cdots & b_{s,s-1} & 0 \\ \hline & c_1 & \cdots & c_{s-1} & c_s \end{array} \Leftrightarrow \begin{array}{c|c} a & B \\ \hline & c^T \end{array}$$

$c, a \in \mathbb{R}^s (a_1 = 0)$  und  $B \in \mathbb{R}^{s,s}$  strenge untere Dreiecksmatrix bestimmt, wobei (aus Konsistenzgründen):

$$\begin{array}{l} a_2 = b_{21} \\ a_3 = b_{31} + b_{32} \end{array} \text{ usw. } \left( \begin{array}{l} Be = a \\ c^T e = 1 \end{array} \right)$$

$$e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^s$$

### Beispiel 7.6

a) explizit Euler:

$$\frac{0 \mid 0}{\mid 1}$$

$$s = 1 \quad p = 1$$

d) modifiziertes Polygonzugverfahren

$$\frac{0 \mid \frac{1}{2}}{\frac{1}{2} \mid \frac{1}{2}} \\ \hline \mid 0 \quad 1$$

$$s = 2 \quad p = 2$$

e) EULER-HEUN Verfahren

$$\frac{0 \mid \mid}{1 \mid 1} \\ \hline \mid \frac{1}{2} \quad \frac{1}{2}$$

$$s = 2 \quad p = 2$$

Um  $p = 4$  zu erreichen ist  $s = 4$  notwendig, aber die Konstante nicht eindeutig bestimmt  $\rightarrow$  mehrere explizite Runge-Kutta-Verfahren

### Bemerkung 7.7

Im allgemeinen  $p = n \not\Rightarrow s = n$

### Beispiel 7.8

mit  $s = 4, p = 4$  einfachstes (besonders günstig für Handrechnung) ist **klassisches Runge-Kutta-Verfahren**

0					$k^{(1)} = f(t_k, y^{(k)})$
$\frac{1}{2}$	$\frac{1}{2}$				$k^{(2)} = f(t_k + \frac{\tau}{2}, y^{(k)} + \frac{\tau}{2}k^{(1)})$
$\frac{1}{2}$	0	$\frac{1}{2}$			$k^{(3)} = f(t_k + \frac{\tau}{2}, y^{(k)} + \frac{\tau}{2}k^{(2)})$
1	0	0	1		$k^{(4)} = f(t_k + \tau, y^{(k)} + \tau k^{(3)})$
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$	$y^{(k+1)} = y^{(k)} + \tau(\frac{k^{(1)}}{6} + \frac{k^{(2)}}{3} + \frac{k^{(3)}}{3} + \frac{k^{(4)}}{6})$

es gibt mindestens 4 weitere bekannte Formeln

- $\frac{3}{8}$ -Formel
- vierstufige Englund - Formel
- Gill-Modifikation der Runge-Kutta-Formel
- Kunthmann Formel

### Bemerkung 7.9

Um  $p = 5$  zu erreichen, ist  $s = 6$  notwendig (ebenfalls wieder mindestens 4 Formeln bekannt).

## 7.4 Implizite Runge-Kutta-Formeln

Definition der Konsistenzordnung analog zu expliziten Formeln nur ist jetzt:

$$R(y) = \frac{y(t+\tau) - y(t)}{\tau} - g(t, y, \tau) \text{ und } g(t, y, \tau) \text{ enthält } y(t + \tau)$$

### Beispiel 7.10 implizit Euler

$$\begin{aligned} y^{(k+1)} &= y^{(k)} + \tau f(t_k + \tau, y^{(k+1)}) \\ g(t, y, \tau) &= f(t + \tau, y(t + \tau)) \end{aligned}$$

$$\begin{aligned}
R(y) &= \frac{y(t+\tau) - y(t)}{\tau} - f(t+\tau, y(t+\tau)) \\
&\stackrel{\text{Taylorentwicklung}}{\downarrow} \\
&= \dot{y}(t) + \frac{\tau}{2}\ddot{y} + \dots - f(t+\tau, y + \tau\dot{y} + \frac{\tau^2}{2}\ddot{y} + \dots) \\
&= \dot{y}(t) + \frac{\tau}{2}\ddot{y} + \dots - [f + \tau f_t(t, y)] + \tau f_t(t, y) + (f_y)(\tau\dot{y}) \\
&= \tau(-\frac{1}{2}\ddot{y}) + \mathcal{O}(\tau^2)
\end{aligned}$$

also Konsistenzordnung auch  $p = 1$

Allgemein ist bei impliziten Verfahren an mindestens einer Stelle ein nichtlineares Gleichungssystem zur Definition eines Predictors  $k^{(j)}$  zu lösen, d.h. gleiche Formeln wie beim expliziten Runge-Kutta-Verfahren nur jetzt eventuell  $b_{jj} \neq 0$  in der Matrix  $B$  (oder gar  $B$  keine Dreiecksmatrix mehr).

### **$s$ -stufiges implizites Runge-Kutta-Verfahren**

#### 1 Gleichungssystem

$$\begin{cases} k^{(i)} = f(t_k + a_i\tau, y^{(k)} + \tau \sum_{j=1}^s b_{ij}k^{(j)}) & i = 1 \dots s \\ k = g(t, y, \tau) = \sum_{i=1}^s c_i k^{(i)} \\ y^{(k+1)} = y^{(k)} + k \end{cases}$$

$$s^2 + 2s \text{ Koeffizienten: } \frac{a \mid B}{c^\top} \quad \left( \begin{array}{l} Be = a \\ c^\top e = 1 \end{array} \right)$$

Verfahren

- explizit, wenn  $B(s \times s)$  strenge untere Dreiecksmatrix
- diagonal implizit, wenn  $B$  untere  $\Delta$ -Matrix  
( $\rightarrow s$  einzelne nichtlineare Gleichungssysteme für jedes  $k^{(i)}$ )
- $B$  volle Matrix  $\rightarrow$  ein nichtlineares Gleichungssystem  $\forall k^{(i)}$

### **Beispiel 7.11**

b) implizit EULER

$$\frac{1 \mid 1}{\mid 1}$$

$$\begin{aligned}
y^{(k+1)} &= y^{(k)} + \tau k^{(1)} \\
&= y^{(k)} + \tau f(t_k + \tau, y^{(k+1)})
\end{aligned}$$

c) implizite Mittelpunkregel

$$y^{k+1} = y^{(k)} + \tau f\left(t_k + \frac{\tau}{2}, \frac{1}{2}(y^{(k)} + y^{(k+1)})\right)$$

$$k^{(1)} = \frac{y^{(k+1)} - y^{(k)}}{\tau}$$

also:

$$k^{(1)} = f\left(t_k + \frac{\tau}{2}, y^{(k)} + \frac{\tau}{2}k^{(1)}\right)$$

$$y^{(k+1)} = y^{(k)} + \tau k^{(1)}$$

$$\frac{\frac{1}{2} \mid \frac{1}{2}}{\frac{1}{2} \mid 1} \quad \text{aber} \quad p = 2 \quad !$$

e) aus Trapezregel

$$y^{(k+1)} = y^{(k)} + \frac{\tau}{2}(f(t_k, y^{(k)}) + f(t_k + \tau, y^{(k+1)}))$$

$$k^{(1)} = f(t_k, y^{(k)})$$

$$k^{(2)} = f\left(t_k + \tau, y^{(k)} + \frac{\tau}{2}k^{(1)} + \frac{\tau}{2}k^{(2)}\right)$$

$$y^{(k+1)} = y^{(k)} + \frac{\tau}{2}(k^{(1)} + k^{(2)})$$

$$s = 2 \quad \frac{0 \mid 0 \quad 0}{1 \mid \frac{1}{2} \quad \frac{1}{2}} \quad \text{ebenfalls} \quad p = 2$$

## 7.5 Stabilität - Vergleich explizit-implizit

Betrifft jetzt die Fortpflanzung eines Fehlers  $y^{(k)} - y(t_k)$  in weiteren gehen wir von einem linearen Differentialgleichungssystem aus

$$\left\{ \begin{array}{l} \dot{y} + Ay = 0 \\ y(0) = y^{(0)} \text{ geg.} \\ f(t, y) = -Ay \\ A \text{ bzgl. } t \text{ konstant und symmetrisch positiv definite Matrix} \end{array} \right.$$

- einfachste Verfahren:
  - a) explizit EULER
  - b) implizit EULER
  - c) CRANK-NICOLSON  
Mittelpunktregel/Trapezregel

$$\text{a) } y^{(k+1)} = y^{(k)} + \tau f(t_k, y^{(k)}) = y^{(k)} - \tau A y^{(k)}$$

$$\boxed{y^{(k+1)} = (I - \tau A)y^{(k)}}$$

b)

$$\begin{aligned} y^{(k+1)} &= y^{(k)} + \tau f(t_k + \tau, y^{(k+1)}) \\ &= y^{(k)} - \tau A y^{(k+1)} \end{aligned}$$

also

$$\boxed{(I + \tau A)y^{(k+1)} = y^{(k)}} \Leftrightarrow \boxed{y^{(k+1)} = (I + \tau A)^{-1}y^{(k)}}$$

c)

$$\begin{aligned} y^{(k+1)} &= y^{(k)} + \tau f\left(t_k + \frac{\tau}{2}, \frac{1}{2}(y^{(k)} + y^{(k+1)})\right) \\ y^{(k+1)} &= y^{(k)} - \tau \frac{1}{2} A (y^{(k)} + y^{(k+1)}) \end{aligned}$$

$$\boxed{(I + \frac{\tau}{2}A)y^{(k+1)} = (I - \frac{\tau}{2}A)y^{(k)}} \Leftrightarrow \boxed{y^{(k+1)} = (I + \frac{\tau}{2})^{-1}(I + \frac{\tau}{2})y^{(k)}}$$

Verallgemeinerung:

$$\begin{array}{ccc} \dot{y} & + & Ay & = & 0 \\ \swarrow & & \searrow & & \\ \frac{y^{(k+1)} - y^{(k)}}{\tau} & & \alpha Ay^{(k)} + (1 - \alpha)Ay^{(k+1)} & & \end{array}$$

$$\dot{y} + Ay = 0 \rightarrow \frac{y^{(k+1)} - y^{(k)}}{\tau} + \alpha Ay^{(k)} + (1 - \alpha)Ay^{(k+1)} = 0$$

$$\boxed{(I + \tau(1 - \alpha)A)y^{(k+1)} = (I - \tau\alpha A)y^{(k)}} \quad \alpha \in [0, 1]$$

$$\begin{array}{lll} \alpha = 1 & \text{explizit EULER} & (p = 1) \\ \alpha = 0 & \text{implizit EULER} & (p = 1) \\ \alpha = \frac{1}{2} & \text{CRANK-NICOLSON} & (p = 2) \end{array}$$

Somit haben die expliziten und impliziten 1-Schrittverfahren für  $\dot{y} + Ay = 0$  letztlich die Gestalt

$$y^{(k+1)} := M(\tau)y^{(k)}$$

**Stabilität**

- sehr vielschichtiger Begriff

1. Es gibt verschiedene Stabilitätsbegriffe für die gegebene DGL

Bspw.  $\dot{y} + Ay = 0$   $0(y) = y^{(0)}$  hat Lösung  $y(t) = e^{-tA}y^{(0)}$

$$\text{mit } e^{-tA} = \sum_{k=0}^{\infty} \frac{(-tA)^k}{k!}$$

da  $A$  positiv definit sind Eigenwerte  $\lambda_i > 0 \implies$  Dämpfung.

2. Stabilität des Runge - Kutta - Verfahren muß diese Eigenschaft der DGL mitmachen

$\rightarrow$  Fehler wird in weiteren Schritten nicht aufschaukelt, sondern gedämpft  
(dazu recht groß Theorie vorhanden)

**Beispiel 7.12**

$$y^{(k+1)} = M(\tau)y^{(k)}$$

$\varrho(M(\tau)) > 1 \implies$  Fehleraufschaukelung  
 $\implies$  Formel unbrauchbar

$\varrho(M(\tau)) < 1 \implies$  Fehler gedämpft  
 $\implies$  Formel „stabil“

- a) explizit EULER

$$\begin{aligned} M(\tau) &= I - \tau A \\ \lambda_i(M) &= 1 - \tau \lambda_i(A) \\ |1 - \tau \rho(A)| &< |\lambda_i(M)| < 1 \\ -(1 - \tau \rho(A)) &< 1, \tau > 0 \\ &\iff \boxed{\tau < \frac{2}{\varrho(A)}} \end{aligned}$$

diese explizite Formel ist bedingt stabil bei genügend kleinem  $\tau$ .

- b) implizit EULER

$$\begin{aligned} M(\tau) &= (I + \tau A)^{-1} \\ \lambda_i(M) &= \frac{1}{1 + \tau \lambda_i} \in (0, 1) \end{aligned}$$

$\implies$  unbedingt stabil

c) CRANK-NICOLSON bzw. allgemeiner Fall

$$\begin{aligned}
 M(\tau) &= (I + (1 - \alpha)\tau A)^{-1}(I - \alpha\tau A) \\
 \text{hat EW } \lambda_i(M(\tau)) &= \left| \frac{(1 - \alpha)\tau\lambda_i}{1 + (1 - \alpha)\tau\lambda_i} \right| < 1 \\
 \Leftrightarrow |1 - \alpha\tau\lambda_i| &< 1 + (1 - \alpha)\tau\lambda_i \\
 1 - \alpha\tau\lambda_i &< 1 + (1 - \alpha)\tau\lambda_i \quad \text{gilt immer} \\
 -(1 - \alpha\tau\lambda_i) &< 1 + (1 - \alpha)\tau\lambda_i \\
 \rightarrow (2\alpha - 1)\tau\lambda_i &< 0
 \end{aligned}$$

- a)  $(2\alpha - 1) \leq 0$   
für  $\alpha \leq \frac{1}{2} \rightarrow$  Formel immer stabil
- b)  $(2\alpha - 1) \geq 0$   
 $\tau < \frac{2}{(2\alpha - 1)\rho(A)} \rightarrow$  bedingt stabil

typische Erkenntnis:

- explizite Formeln bedingt stabil und  $|\tau| <$  Schranke
- implizite Formeln unbedingt stabil aber oft größere  $\tau$  wählbar

## 7.6 Steife Differentialgleichungen

betrifft wieder  $\dot{y} + Ay = 0$  mit  $y(0) = y^{(0)}$

Lösung:  $y(t) = e^{-tA}y^{(0)}$

betrachten Eigenproblem für A

$$Aq^{(i)} = \lambda_i q^{(i)} \quad \text{mit } \lambda_i \text{ EW und } q^{(i)} \text{ EV}$$

$$\text{sei } A = A^T > 0 \rightarrow \text{o.B.d.A.} \quad (q_i, q_j) = \delta_{ij}$$

$$\text{zerlegen } y^{(0)} = \sum_{i=1}^n \alpha_i q^{(i)}$$

$$\Rightarrow y(t) = \sum \underbrace{(\alpha_i e^{-t\lambda_i})}_{\text{Koordinaten von } y(t) \text{ in Eigenbasis}} q_i$$

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_k$$

- (a) bei großen Eigenwerten wesentliche Vorgänge in  $(0, \tau_0)$   $\tau_0 \ll 1$
- (b) bei kleinen Eigenwerten langsame Annäherung an stationären Zustand

allgemeiner:

A hat auch komplexe Eigenwerte die Zeitabhängig sind  
Lösung ist hier Überlagerung von

- (a) starke Oszillation in kleinen Zeiten
- (b) langsam annähern an stationären Zustand

**Definition 7.13**

Differentialgleichungen mit stark unterschiedlichen Eigenwerten heißen steife Differentialgleichungen.

**Bemerkung 7.14**

Lösung solcher steifer DGLs erfordert Umschalten zwischen expliziten und impliziten Verfahren und einer Schrittweitensteuerung.

- kleine Zeiten  $\rightarrow$  kleines  $\tau$  und explizites Verfahren
- später  $\rightarrow$   $\tau$  vergrößern und Wechsel zu impliziten Verfahren

# Kapitel 8

## Randwertprobleme - (RWP)

### 8.1 Vorbemerkungen

Besondere Bedeutung haben Randwertprobleme bei partiellen Differentialgleichungen vor allem für gesuchte Funktionen im 2- bzw. 3-dimensionalen

#### Beispiel 8.1

Wärmeverteilung in  $\Omega \subset \mathbb{R}^2$  wenn am Rand Temperatur oder Flüsse gegeben sind.

$$\begin{aligned} \frac{\partial u}{\partial w} &= 0 \text{ an allen anderen Rändern} \\ -\Delta u &= f(\vec{x}) \text{ in } \Omega \text{ bei Temperaturquellen und -senken} \\ \text{mit } \Delta &= \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial u^2} \end{aligned}$$

betrachten analog zu Kapitel 7 nur Funktionen von einer reellen Veränderlichen  $x$ :

ges.:  $u(x)$  mit

$$u''(x) = f(x)|_{[a,b]}$$

geg.:  $u(a) = u_a$   $u(b) = u_b$

allgemeiner:

$$\alpha(x)u''(x) + \beta(x)u'(x) + \gamma(x)u(x) = f(x)$$

$$u(a) = u_a$$

$$u(b) = u_b$$

(oder auch  $u'(a), u'(b)$  vorgegeben)

unter gewissen Einschränkungen von  $\alpha(x), \beta(x), \gamma(x)$

z.B.

$$\alpha(x) \leq \alpha_0 < 0 \forall x$$

$$\gamma(x) \geq 0$$

bei solchen linearen Differentialgleichungen im RWP spielt die Diskretisierung zentrale Rolle.

## 8.2 Differenzenverfahren

**Grundidee:** Geben diskrete Punkte

$$a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$$

vor, an denen  $u_k \approx u(x_k)$  zu berechnen sind. ( $u_0 = u_a, u_n = u_b$  gegeben)  
betrachten o.E.d.A.: äquidistante Diskretisierungspunkte

$$x_k = a + kh \quad h = \frac{b-a}{n}$$

**Differenzenverfahren** sind Näherungsverfahren zur Lösung von RWP, die durch Annäherung von  $u''(x_k)$  entstehen.

### Beispiel 8.2

betrachte

$$\begin{cases} -u''(x) = f(x) & |_{[a,b]} \\ u(a) = u_a & u(b) = u_b \end{cases}$$

aus Kapitel 6:

$$u''(x) = \frac{u(x_k-h) - 2u(x_k) + u(x_k+h)}{h^2} + \mathcal{O}(h^2)$$

andererseits soll  $-u''(x_k) = f(x_k) \quad k = 1, \dots, n-1$   
ersetzen  $u(x_k)$  durch gesuchte Näherung  $u_k$

$$\implies -u_{k-1} + 2u_k - u_{k+1} = h^2 f(x_k) \quad k = 1, \dots, n-1$$

$\implies (n-1)$  Gleichungen mit  $(n-1)$  Unbekannten  $u_1 \dots u_{n-1}$  da  $u_0, u_n$  gegeben

#### Frage:

Wird bei  $k \rightarrow 0 \quad u_k \rightarrow u(x_k)$  konvergieren?

Typische Untersuchung: einsetzen der exakten Lösung  $u(x_k)$  in das Näherungsproblem

(a)  $-u_{k-1} + 2u_k - u_{k+1} = h^2 f(x_k)$  gilt exakt  $\forall k$

(b)  $-u(x_{k-1}) + 2u(x_k) - u(x_{k+1}) = h^2 f(x_k)$

aus Taylorentwicklung folgt

$$u(x_{k+1}) = u(x_k) + hu'(x_k) + \frac{h^2}{2}u''(x_k) + \frac{h^3}{3!}u'''(x_k) + \frac{h^4}{4!}u^{(iv)} + \mathcal{O}(h^5)$$

$$u(x_{k-1}) = u(x_k) - hu'(x_k) + \frac{h^2}{2}u''(x_k) - \frac{h^3}{3!}u'''(x_k) + \frac{h^4}{4!}u^{(iv)} + \mathcal{O}(h^5)$$

$$-2u(x_k) - h^2 u''(x_k) - \frac{h^4}{12} u^{(iv)}(x_k) + \mathcal{O}(h^5) = h^2 f(x_k) + h^4 * const$$

$$T\underline{u} = \underline{b} \quad \underline{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_{n-1} \end{bmatrix} \quad \text{Naherung} \quad \underline{b} = (h^2 f(x_k))_{k=1}^{n-1}$$

$$T\underline{u}_{ex} = \underline{b} \quad \underline{u}_{ex} = \begin{bmatrix} u(x_1) \\ \vdots \\ u(x_{n-1}) \end{bmatrix} \quad \text{exakt}$$

$$T(\underline{u} - \underline{u}_{ex}) = \frac{h^4}{12} (u^{(iv)}(x_k) + \mathcal{O}(h)) \quad \text{heißt Approximation}$$

**Bemerkung 8.3**

Achtung  $k \rightarrow 0$  nicht so einfach moglich, da T sonst wachst.

aus oben  $\implies \|\underline{u} - \underline{u}_{ex}\| \leq \|T^{-1}\| \cdot \frac{h^4}{12} \|\underline{u}^{(iv)}\| + \mathcal{O}(h^5) \|T^{-1}\|$  hangt von  $h$  ab!

$$T \text{ symmetrisch} \implies \|T\|_2 = \lambda_{\max}(T) \\ \|T^{-1}\|_2 = \lambda_{\min}(T)^{-1}$$

$$\lambda_{\min}(T) = 4 \sin^2 \frac{k\pi}{n} \\ = 4 \frac{\pi}{b-a} h^2 + \mathcal{O}(h^3)$$

$$\implies \|\underline{u} - \underline{u}_{ex}\|_2 = \mathcal{O}(h^2) \rightarrow 0 \quad \text{fur } h \rightarrow 0$$

besser benutze Maximumnorm:  $\|\underline{u} - \underline{u}_{ex}\|_\infty \leq \|T^{-1}\|_\infty \|\frac{h^4}{12} u^{(iv)}(x_k) + \mathcal{O}(h)\|_\infty$

$\implies$

$$\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} h^2 f(x_1) + u_a \\ h^2 f(x_2) \\ \vdots \\ h^2 f(x_{n-2}) \\ h^2 f(x_{n-1}) + u_b \end{bmatrix}$$

Systemmatrix ist tridiagonal mit tridiag  $(-1, 2, -1)$  regular, symmetrisch und diagonal dominant

$\implies u_1 \dots u_{n-1} \forall n (\forall h)$  stabil berechenbar  $h \rightarrow 0$

Ahnliche Vorgehensweise im mehrdimensionalen Beispiel:

$$\begin{aligned}
 -\Delta u(x, y) &= f(x, y)|_{\Omega} \\
 u &= g(x, y)|_{\partial\Omega}
 \end{aligned}$$

Diskretisierung in beide Raumrichtungen

$x$	$k-1$	$k$	$k+1$	
$j+1$	·	·	·	$u(x_k, y_j) \approx u_{kj}$
$j$	·	·	·	
$j-1$	·	·	·	

$$\begin{aligned}
 -\frac{\partial^2 u}{\partial x^2} \Big|_{(x_k, y_j)} &\approx \frac{-u_{k-1j} + 2u_{kj} + u_{k+1j}}{h^2} \\
 -\frac{\partial^2 u}{\partial y^2} \Big|_{(x_k, y_j)} &\approx \frac{-u_{kj-1} + 2u_{kj} - u_{kj+1}}{h^2}
 \end{aligned}$$

⇒ entsteht der sogenannte 5-Punkte-Stern im 2 Dimensionalen

$$-\Delta u|_{x_k, y_j} \approx \left( \frac{1}{h^2} 4u_{kj} - u_{k-1j} - u_{k+1j} - u_{kj-1} - u_{kj+1} \right)$$

analog im 3 Dimensionalen.

Schwierigkeiten bei der Approximation machen hier die Randbedingungen, wenn  $\Omega$  nicht z.B. ein Rechteck o.ä. ist.

### 8.3 Finite Elemente Methode - FEM

**Grundidee:**

betrachten wieder Diskretisierungspunkte um Näherungswerte von  $u(x_k)$  zu erhalten. Aber: es wird eine Näherungsfunktion berechnet mit Hilfe eines Projektionsverfahrens in einen endlich-dimensionalen Teilraum. Dieser Teilraum wird bei FEM durch einfache Spline- Funktion mit kleinem Träger aufgespannt.

**Vorgehensweise:**

- a) Bilineares Funktional  $a(u, v)$  (sogenannte schwache Formulierung) zur Repräsentation der Differentialgleichung.
- b) Festlegen der Basis, Projektion von  $a(u, v)$  in endlich-dimensionalen Teilraum
- c) Lösen des Gleichungssystems  $A\underline{a} = \underline{b}$

**Beispiel 8.4** zu a)

$$\text{Aufgabe} \begin{cases} -u''(x) = f(x)|_{[a,b]} \\ u(a) = u(b) = 0 \end{cases} \quad (1)$$

$$\implies -u''(x)v(x) = f(x)v(x) \quad \forall v \in C[a, b]$$

$$-\int_a^b u''v dx = \int_a^b f v dx$$

wählen  $v(a) = v(b) = 0$  und sei  $v \in C$  stückweise diffbar

$$\implies \text{ges. } u(x) \text{ mit } u(a) = u(b) = 0 \text{ nach partieller Integration}$$

$$\int_a^b u'(x)v'(x) dx = \int_a^b f(x)v(x) dx \quad \forall v \in C^1[a, b], v(a) = v(b) = 0$$

<u>Aufgabe:</u>	$\begin{aligned} a(u, v) &= \langle f, v \rangle \\ u(a) &= u(b) = 0 \\ \text{mit } \langle f, v \rangle &= \int_a^b f v dx \end{aligned}$	(2)
-----------------	--	-----

**Bemerkung 8.5**

Abschließung dieser Funktionenräume ergibt eine Formulierung in allgemeinen Räumen (sogenannte verallgemeinerte Ableitungen für  $u$  und  $v$  existieren also auch stückweise diffbare Funktionen sind zugelassen!) bezeichnet mit  $\mathbb{V}_0$ .

b) jetzt: Näherungslösung  $u_h \in \mathbb{V}_0$  dadurch, daß (1) in endlichen dimensionalen Teilraum  $\mathbb{V}_h \subset \mathbb{V}_0$  projiziert wird.

**Beispiel 8.6** zu b)

$$\mathbb{V}_h = \text{span} (\varphi_1, \dots, \varphi_{n-1})$$

$\varphi_k$  sogenannte Hütchenfunktion (B-Spline)

$$a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$$

$\varphi_k(x)$  sei linear in allen Teilintervallen  $[x_{i-1}, x_i]$  und  $\varphi_k(x_j) = \delta_{ij}$

$\varphi_k$  stetig, stückweise diffbar mit  $k = 1, \dots, n-1$

$$u_h(x) = \sum_{j=1}^{n-1} \alpha_j \varphi_j(x)$$

Projektion von (1) in  $\mathbb{V}_h$  heißt:

ges.:  $u_h(x) \in \mathbb{V}_h : a(u_h, v) = \langle f, v \rangle \quad \forall v \in \mathbb{V}_h \subset \mathbb{V}_0$

**Bemerkung 8.7** zu c)

1. Die gesuchten DGL- Koeffizienten  $\alpha_k$  sind häufig gleichzeitig Funktionswerte der Näherungsfunktion

$$u_h(x_h) = \sum \alpha_i \varphi_i(x_h) = \alpha_k$$

2. Berechnung der  $\alpha_k$  aus einem Gleichungssystem

$$a\left(\sum \alpha_j \varphi_j, \varphi_i\right) = \langle f, \varphi_i \rangle \quad \forall i = 1, \dots, n-1$$

$$A \underline{\mathbf{a}} = \underline{\mathbf{b}}$$

$$\text{mit } A = (a(\varphi_j, \varphi_i))_{i,j=1}^{k-1}$$

$$\underline{\mathbf{b}} = (\langle f, \varphi_i \rangle)_{i=1}^{k-1}$$

$$\underline{\mathbf{a}} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{bmatrix} = \begin{bmatrix} u_k(x_1) \\ \vdots \\ u_k(x_{n-1}) \end{bmatrix}$$

**Beispiel 8.8**

$$A : a(\varphi_i, \varphi_j) = \int_a^b \varphi_i' \varphi_j' dx \quad \text{und} \quad a_{kk} = a(\varphi_i, \varphi_j) = \begin{cases} -\frac{1}{h} & j = i \pm 1 \\ \frac{2}{h} & j = i \\ 0 & \text{sonst} \end{cases}$$

$$\begin{aligned}
\text{mit } x_{i+1} - x_i &= h \\
\Rightarrow a_{kk} &= \int_{x_{k-1}}^{x_k} \left(\frac{1}{h}\right)^2 dx + \int_{x_k}^{x_{k+1}} \left(-\frac{1}{h}\right)^2 dx = \frac{2}{h} \\
a_{kk+1} &= \int_{x_k}^{x_{k+1}} \left(-\frac{1}{h}\right) \left(\frac{1}{h}\right) dx = -\frac{1}{h} \\
(a_{kj} &= 0 \quad \forall j > k+1) \\
\Rightarrow A &= \frac{1}{h} \begin{bmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & -1 & \\ & & -1 & 2 & \end{bmatrix}
\end{aligned}$$

wie bei Differenzenverfahren aber leicht andere rechte Seite b.

# Kapitel 9

## Eigenwertprobleme bei Matrizen

### 9.1 Grundlagen der linearen Algebra

#### Definition 9.1

$A \in \mathbb{C}^{n,n}$  ( $n \times n$  Matrix)

$(\lambda, w)$  nennt man Eigenpaar zu  $A$  ( $\lambda$  Eigenwert,  $w \in \mathbb{C}^{n,n}$  Eigenvektor)  
mit  $Aw = \lambda w$

**Bemerkung 9.2** i.A.  $\lambda \in \mathbb{C}^{n,n}$  bei  $A = A^\top$   $\lambda \in \mathbb{R}, w \in \mathbb{R}^n$

#### Definition 9.3

Eigenwert nennt man  $\lambda$  als Nullstelle des charakteristischen Polynoms

$$\varphi(t) = \det(tI - A) \in \Pi_n$$

- wenn  $\lambda$   $\alpha$ -fache Nullstelle, dann nennt man  $\alpha$  algebraische Vielfachheit von  $\lambda$
- bei  $\alpha > 1$  gibt es  $\beta$  linear unabhängige Eigenvektoren von  $\lambda$  mit  $1 \leq \beta \leq \alpha$  und  $\beta$  nennt man geometrische Vielfachheit von  $\lambda$

#### Definition 9.4

Falls  $\beta < \alpha$  für mindestens einen Eigenwert  $\lambda$  von  $A$ , dann heißt  $A$  defektiv.

#### Satz 9.5

$A$  ist diagonalisierbar, falls  $\beta = \alpha \quad \forall \lambda$  und es existiert

$$W = (w_1 | \dots | w_n)^\top \quad W^{(-1)}AW = \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$$

**Definition 9.6**

$A$  heißt normal  $\Leftrightarrow AA^* = A^*A \Leftrightarrow$  Eigenvektoren bilden Orthonormalbasis

**Bemerkung 9.7**

falls  $A = A^*$  hermitesch oder  $A = A^\top$  symmetrisch, so ist  $A$  auch normal

**Satz 9.8**

Alle normalen Matrizen sind diagonalisierbar.

**Satz 9.9** Cayley - Hamilton

Sei  $K$  ein Körper und  $A \in K^{n,n}$  mit dem charakteristischen Polynom  $\varphi_A(\lambda)$ .  
Dann erfüllt  $A$  die Gleichung

$$\varphi_A(A) = A^n + a_1 A^{(n-1)} + \dots + a_{n-1} A + a_n I_n = 0$$

## 9.2 Ungeeignete Verfahrensideen

**Bemerkung 9.10**

Fast alle Verfahren die etwas mit dem charakteristische Polynom  $\varphi(\lambda)$  zu tun haben sind numerisch instabil oder zu aufwendig.

- (a) Verfahren die Koeffizienten von  $\varphi(\lambda)$  berechnen:

$$\varphi(t) = t^n + \sum_{i=0}^{n-1} a_i t^i$$

- (b) Verfahren zur Nullstellenbestimmung:  $\varphi(t) = 0$  (ohne Koeffizienten zu bestimmen)

z.B.: Sekantenverfahren

$$t_{k+1} = \frac{t_k \varphi(t_{k-1}) - t_{k-1} \varphi(t_k)}{\varphi(t_{k-1}) - \varphi(t_k)}$$

$\Rightarrow$  jeweils eine Determinante pro Iteration

- (1) Aufwand  $\mathcal{O}(n^3)$
- (2) Funktionswert mittels  $\det(t_k I - A)$  ist eventuell sehr kleine/große Zahl ausserhalb des Zahlendarstellung

Bsp.:

$$\begin{bmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ & 2 & 1 & \cdots & 1 \\ & & \ddots & & \vdots \\ & & & \ddots & 1 \\ & & & & n \end{bmatrix} \quad \text{hat } \det(A) = n!$$

### Bemerkung 9.11 Ausnahme

Bei Tridiagonalmatrizen existiert ein stabiler Ausweg (siehe 9.3)

#### veraltetes Verfahren

basiert auf Satz 9.9 (Verfahren von DERVINDUE)

$$\begin{aligned} \varphi(A) = 0 & \quad \varphi(A)x^{(0)} = 0 \\ \left( A^n + \sum_{i=0}^{n-1} a_i A^i \right) x^{(0)} = 0 \end{aligned}$$

setzen  $x^{(i+1)} = A^{(i)}x^{(i)} \quad i = 0, \dots, n-1$

$$x^{(n)} + \sum_{i=0}^n a_i x^{(i)} = 0$$

$$X \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = -x^{(n)} \quad \text{mit } X = (x^{(0)} | x^{(1)} | \dots | x^{(n-1)})$$

also ist  $(a_0, a_1, \dots, a_{n-1})^\top$  ist Lösung dieses Gleichungssystems instabil:

- (a) Überlauf, Unterlauf bei  $x^{(i+1)} = Ax^{(i)}$
- (b)  $X$  extrem schlecht konditioniert

## 9.3 Iterative Berechnung von Eigenwerten bei symm. Tridiagonalmatrizen

Bezeichnung:  $T_k$  führende Hauptuntermatrix von

$$T = T_n = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_n & \\ & & & \beta_n & \alpha_n \end{bmatrix}$$

und  $\varphi_k(t) = \det(tI - T_k)$   
 $\Rightarrow$  für gegebene Zahl läßt sich  $\varphi(t) = \varphi_1(t)$  rekursiv berechnen (eventuell instabil)

$$\begin{aligned} \varphi_0(t) &= 1 \\ \varphi_1(t) &= t - \alpha_1 \\ &\vdots \\ \varphi_k(t) &= (t - \alpha_k)\varphi_{k-1}(t) - \beta_k^2\varphi_{k-2}(t) \quad \forall k = 2, \dots, n \end{aligned}$$

**Bemerkung 9.12**

Somit iterative Verfahren zur Bestimmung einer Nullstelle von  $\varphi_n(t)$  möglich. Jetzt nur noch  $O(n)$  Operationen pro Schritt aber  $\varphi_k(t)$  evtl. sehr große/kleine Zahlen  $\rightarrow$  instabil.

Vermeidung der Instabilität durch  
 definiere:  $f_k(t) = \frac{\varphi_k(t)}{\varphi_{k-1}(t)}$  stabil  $\forall t$  berechenbar

$\begin{aligned} f_k(t) &= (t - \alpha_k) - \beta_k^2 \frac{1}{f_{k-1}(t)} \\ f_1(t) &= t - \alpha_1 \end{aligned}$
---

**Satz 9.13**

Benutzen dies für  $T$  mit  $\beta_i \neq 0 \forall i$   
 (sonst berechnen Eigenwerte von 2 Teilblöcken einzeln)

$$\begin{array}{ccc|ccc} a_1 & \ddots & & & & \\ \ddots & \ddots & \ddots & & & \\ & \ddots & \ddots & & & \\ \hline & & & 0 & & \\ & 0 & & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & \ddots & \alpha_n \end{array}$$

$\rightarrow \beta_k \neq 0 \quad \forall i$

- alle Eigenwerte von  $T$  sind einfach
- Nullstellen von  $\varphi_k(t)$  separieren sich

**Folgerung 9.14**

Benutzen also Nullstellenbestimmungsverfahren für  $f_n(t) = 0$

- (a) Sekantenverfahren
- (b) Newton-Verfahren (noch  $f'_k(t) = 1 - \beta_k^2 \frac{f'_{k-1}(t)}{(f_{k-1}(t))^2}$  mit Start  $f'_1(t) = 1$  mit Rekursion für  $f_k(t)$  mitrechnen)
- (c) Bisektion (Intervallhalbierung) zur Berechnung z.B.: des n-ten Eigenwertes mit

$$\lambda_1 < \lambda_2 < \dots < \lambda_n \text{ Eigenwerte}$$

Ausnutzen  $\{\varphi_k(t)\}_{k=0}^n$  bilden Sturmsche Kette

### Satz 9.15

Betrachten  $t = \bar{t}$  fest und  $\varphi_0(\bar{t}), \dots, \varphi_n(\bar{t})$

Anzahl der Vorzeichenwechsel dieser Zahlen  $\equiv$  Anzahl der der Eigenwerte oberhalb von  $\bar{t} \Leftrightarrow f_1(\bar{t}), \dots, f_n(\bar{t})$  berechenbar

Anzahl der negativen  $f_k(\bar{t}) \equiv$  Anzahl der Eigenwerte mit  $\lambda > \bar{t}$

### Verfahrensidee

betrachten Intervall  $[a, b]$  mit  $\{f_k(a)\}$  und  $\{f_k(b)\}$

$$c = \frac{a+b}{2} \quad \{f_k(c)\}$$

m-ter Eigenwert in  $[c, b] \Rightarrow [a, b] := [c, b]$

m-ter Eigenwert in  $[a, c] \Rightarrow [a, b] := [a, c]$

$\Rightarrow \{f_k(a)\}$  und  $\{f_k(b)\}$

## 9.4 Berechnung von Eigenwerten und Eigenvektoren durch Transformation

Seien  $w_1 \dots w_n$  Eigenvektoren zu  $\lambda_1 \dots \lambda_n$  von A

$$W = (w_1 | w_2 | \dots | w_n) \quad \Lambda = W^{-1}AW = \text{diag}(\lambda_1 \dots \lambda_n)$$

### Verfahrensidee

$$A^0 = A \quad A^{(k+1)} = (X^{(k)})^{-1}A^{(k)}X^{(k)}$$

mit X so daß A näher an Diagonalgestalt

### einfache Fehlerbetrachtung

$$A^{(k)} + F^{(k)} \quad \text{mit } \|F^{(k)}\| \text{ klein}$$

$$\begin{aligned} A^{(k+1)} &= (X^{(k)})^{-1}(A^{(k)} + F^{(k)})X^{(k)} \\ &= (X^{(k)})^{-1}A^{(k)}X^{(k)} + (X^{(k)})^{-1}F^{(k)}X^{(k)} \end{aligned}$$

hätten gern

$$\|(X^{(k)})^{-1}F^{(k)}X^{(k)}\|_2 \leq \|F^{(k)}\|_2 \kappa(X^{(k)}) \quad \text{mit Konditionszahl } \kappa(X^{(k)}) > 1$$

**Ziel:**  $\kappa(X^{(k)}) = 1$

nur bei orthogonalen/unitären Matrizen

→ also benutzen nur orthogonale Matrizen zur Transformation

bei  $A \neq A^\top$  Diagonalisierung nicht immer möglich

→ SCHURsche Normalform

### Satz 9.16

$\exists U$  unitäre Matrix mit

$$U^*AU = \Delta = \text{obere Dreiecksmatrix}$$

sogenannte Blockschurform falls A reell:

$\exists Q$  orthogonal mit

$$Q^\top A Q = \begin{bmatrix} A_1 & B_i & \cdots & B_j \\ & \ddots & \ddots & \vdots \\ & & \ddots & B_m \\ & & & A_n \end{bmatrix}$$

wobei  $A_i$  eine  $(1 \times 1)$  Matrix bei reellen Eigenwerten oder  $(2 \times 2)$  Matrix mit konjugiert komplexen Eigenwerten

z.B.:

$$\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \text{ hat Eigenwerte } \alpha \pm i\beta$$

bester Vertreter dieser Kategorie ist **QR- Algorithmus**

### Algorithmus 9.17 Grundalgorithmus

- (1)  $A^{(k)} = Q^{(k)}R^{(k)}$  QR- Zerlegung  
mit  $Q^{(k)}(Q^{(k)})^\top = I$  und  $R^{(k)}$  obere Dreiecksmatrix
- (2)  $A^{(k+1)} = R^{(k)}Q^{(k)}$

### Bemerkung 9.18

zu (1)  $R^{(k)} = (Q^{(k)})^\top A^{(k)}$

zu (2)  $A^{(k+1)} = (Q^{(k)})^\top A^{(k)}Q^{(k)}$

also  $\{A^{(k)}\}$  Folgen ähnlicher Matrizen mit  $A^{(k)} \rightarrow$  obere Blockdreiecksmatrix (bei gewissen Voraussetzungen an das Spektrum von A)

### Bemerkung 9.19

Konvergenzgeschwindigkeit hängt von Quotienten  $\left| \frac{\lambda_i}{\lambda_j} \right|$   $i > j$  ab, wenn

$|\lambda_1| \geq \dots \geq |\lambda_n|$  (diese Anordnung auf der Diagonalen der Dreiecksmatrix geordnet)

**Bemerkung 9.20**

Konvergenzgeschwindigkeit ist bei Nebendiagonalen am langsamsten

Hieraus wurden durch folgende Zusätze höchsteffektive Verfahren entwickelt:

**(1) Verschiebungsstrategie**

$$(1) A^{(k)} - s_k I = Q^{(k)} R^{(k)}$$

$$(2) A^{(k)} = Q^{(k)} R^{(k)} + s_k I$$

$$\begin{aligned} \text{NR.: } R^{(k)} &= Q^{(k)}(A^{(k)} - s_k I) \\ &= (Q^{(k)})^\top (A^{(k)} - s_k I) Q^{(k)} + s_k I \\ &= (Q^{(k)})^\top (A^{(k)} Q^{(k)} - s_k I + s_k I) \end{aligned}$$

**Bemerkung 9.21**

Aus  $\frac{|\lambda_i - s_k|}{|\lambda_j - s_k|}$  folgt, daß wenn  $s_k$  sehr nahe am Eigenwert  $\lambda_i$  ist, sich der Eigenwert  $\lambda_i$  sehr schnell abspaltet, da der Konvergenzgeschwindigkeitsquotient sehr groß ist.  $\rightarrow$  letzte Zeile von  $A^{(k+1)}$  sehr klein  
Verfahren aber aufwendig.

**(2) Hessenberg Form****Definition 9.22**

H heißt Hessenbergmatrix wenn Einträge  $h_{i,j} = 0 \quad \forall j < i - 1$

Algorithmus wird nicht auf volle Matrix angewendet sondern auf Hessenberg - Matrix.

$$A^{(0)} = Q^\top A Q = H$$

**Bemerkung 9.23**

Der QR Algorithmus ist invariant bezüglich dieser Struktur  
 $\Rightarrow$  alle  $A^{(k)}$  sind Matrizen mit Hessenbergstruktur.

### 9.4.1 Spiegelungsmatrizen

#### Definition 9.24

Eine Matrix  $S(v)$  heißt Spiegelungsmatrix wenn sie die Form  $S(v) = I - 2vv^\top$  für  $\|v\| = 1$  hat (hier immer  $\|\cdot\| = \|\cdot\|_2$ )

#### Satz 9.25

Diese Matrix hat folgende Eigenschaften:

- $S(v) = S^\top(v)$
- $S^\top S = S^2 = I - 4vv^\top + 4v(v^\top v)v^\top = I$
- $S(v)v = v - 2vv^\top v = -v$
- $x \perp v : S(v)x = x - 2vv^\top x = xy$

#### Nutzen

gegebener Vektor  $a \in \mathbb{R}^n$  dann  $\exists S(v)a = \pm \|a\|e_1$

Konstruktion  $v = \frac{u}{\|u\|}$

$$S(v)a = a - 2\frac{uu^\top a}{\|u\|^2} = a - \left(\frac{2u^\top a}{\|u\|^2}\right)u = \begin{bmatrix} \pm \|a\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

setzen  $(u_2, \dots, u_n)^\top = (a_2, \dots, a_n)^\top$

$$u = \begin{bmatrix} u_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \text{und Abkürzung: } s^2 = \sum_{i=2}^n a_i^2$$

$$a - \frac{2u^\top a}{\|u\|^2}u = a - \frac{2(u_1 a_1 + s^2)}{u_1^2 + s^2}u$$

soll ab Position 2 Nullen ergeben

$$\begin{aligned} 1 - \frac{2(u_1 a_1 + s^2)}{u_1^2 + s^2} &= 0 \\ u_1^2 + s^2 - 2u_1 a_1 - 2s^2 &= 0 \\ u_1 &= a_1 \pm \sqrt{a_1^2 + s^2} \\ &= a_1 \pm \|a\| \end{aligned}$$

beide Vorzeichen sind erlaubt aber zur Vermeidung von Stellenauslöschung

$$u_1 = a_1 + \|a\| \operatorname{sign}(a_1)$$

**Fazit**

Geg.:  $a \in \mathbb{R}^n$

def.:  $u_1 = a_1 + \|a\| \operatorname{sign}(a_1)$

$u_i = a_i \quad i \geq 2$

$\rightarrow S \left( \frac{u}{\|u\|} \right) a = a - \frac{2u^T a}{\|u\|^2} u = \pm \|a\| e_1$  mit festen Vorzeichen

**9.4.2 Nutzung der Spiegelungen zur QR- Zerlegung**

**Ziel:**  $A = QR$  (mit Q,R wie oben)

**Idee:**  $(n - 1)$  Spiegelungen von links

(1)

$$S_1 A = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ \vdots & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix}$$

$$\text{durch } S_1 = \left( \frac{u^{(1)}}{\|u^{(1)}\|} \right) \quad u^{(1)} = \begin{bmatrix} a_{11} + \|a^{(1)}\| \operatorname{sign}(a_{11}) \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix}$$

(2)

$$S_2 S_1 A = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ \vdots & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \text{REST} & \\ 0 & 0 & & \end{bmatrix}$$

$$\text{durch } S_2 = \left( \frac{u^{(2)}}{\|u^{(2)}\|} \right) \quad u^{(2)} = \begin{bmatrix} 0 \\ \tilde{a}_{22} + \|\tilde{a}^{(1)}\| \operatorname{sign}(\tilde{a}_{22}) \\ \tilde{a}_{23} \\ \vdots \\ \tilde{a}_{2n} \end{bmatrix}$$

$$\text{usw. } \underbrace{S_{n-1} \cdots S_2 S_1}_{\text{orthogonal}} A = R \Leftrightarrow A = \underbrace{S_1 \cdots S_{n-1}}_Q R$$

So würde die QR- Zerlegung in 9.4 aussehen.

ABER 1 Zerlegung kostet  $O(n^3)$  Operationen  $\rightarrow$  Gesamtaufwand ist mit mindestens  $O(n^4)$  zu teuer

**Bemerkung 9.26**

QR- Zerlegung ist auch für lineare Gleichungssystem interessant.

$$\begin{aligned} Ax &= b \\ \underbrace{S_{n-1} \cdots S_1 A}_R x &= \underbrace{S_{n-1} \cdots S_1}_\tilde{S} b \end{aligned}$$

**9.4.3 Transformation auf Hessenberg Gestalt**

$$\begin{bmatrix} h_{11} & & & & \\ h_{21} & \ddots & & h_{ij} & \\ 0 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & h_{nn-1} & h_{nn} \end{bmatrix} = H = Q^T A Q \quad \text{durch n-2 Spiegelungen}$$

$$S_1 \text{ so, daß } S_1 A = \begin{bmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ h_{21} & \cdots & \cdots & h_{2n} \\ 0 & \ulcorner & & \lrcorner \\ \vdots & & \tilde{A} & \\ 0 & \llcorner & & \lrcorner \end{bmatrix}$$

$$\text{mit } S_1 = \begin{pmatrix} u_1 \\ \|u_1\| \end{pmatrix} \quad u_1 = \begin{bmatrix} 0 \\ \tilde{a}_{21} + \|\tilde{a}^{(1)}\| \text{sign}(\tilde{a}_{21}) \\ \tilde{a}_{31} \\ \vdots \\ \tilde{a}_{n1} \end{bmatrix} \quad \text{ändert Zeilen 2 bis n}$$

$S_1 A S_1^{(-1)} = S_1 A S_1$  ändert nur noch Spalten 2 bis n

$$\begin{aligned} H &= \underbrace{S_{n-2} \cdots S_1}_Q A \underbrace{S_1 \cdots S_{n-2}}_{Q^T} \\ H &= \underbrace{Q}_Q A \underbrace{Q^T}_{Q^T} \quad (\text{Ähnlichkeitstransformationen}) \end{aligned}$$

**Bemerkung 9.27**

Ist  $A = A^\top \rightarrow H^\top = (Q^\top A Q)^\top = Q^\top A^\top Q = H$  dh.  $H$  ist Tridiagonalmatrix.

**Aufwand:**

- $\frac{2}{3}n^3$  bei  $A = A^\top$
- $\frac{5}{3}n^3$  sonst

**9.4.4 Der Doppelschritt QR nach FRANCIS****Bemerkung 9.28**

$A = A^\top \rightarrow$  1.)  $A$  tridiagonalisiert  
2.) QR Algorithmus für Tridiagonalmatrix

Weiterhin jetzt  $A \neq A^\top$ :

(1)  $A \rightarrow H$  Hessenberg Matrix

(2) QR Algorithmus für  $H$

Grundalgorithmus:

$$\begin{aligned} H^{(k)} - s_k I &= Q^{(k)} R^{(k)} \\ H^{(k+1)} &:= R^{(k)} Q^{(k)} + s_k I \end{aligned}$$

**Doppelschritt:**

$$\left. \begin{aligned} H_1 - s_1 I &= Q_1 R_1 \\ H_2 &= R_1 Q_1 + s_1 I \end{aligned} \right\} \text{1. Schritt} \quad H_2 = Q_1^\top H_1 Q_1$$

$$\left. \begin{aligned} H_2 - s_2 I &= Q_2 R_2 \\ H_3 &= R_2 Q_2 + s_2 I \end{aligned} \right\} \text{2. Schritt} \quad H_3 = Q_2^\top Q_1^\top H_1 Q_1 Q_2$$

Suchen Matrix um direkt von  $H_1 \rightarrow H_3$  zu kommen.

Betrachten

$$\begin{aligned} P &= (H_1 - s_1 I)(H_1 - s_2 I) = Q_1 R_1 (Q_1 R_1 + (s_1 - s_2) I) \\ &= Q_1 \underbrace{R_1 Q_1}_{R_1} R_1 + (s_1 - s_2) Q_1 R_1 \\ &= Q (H_2 - s_1 I) R_1 + (s_1 - s_2) Q_1 R_1 \\ &= Q_1 ((H_2 - s_2 I) R_1) \\ &= (Q_1 Q_2) (R_2 R_1) \\ &= \tilde{Q} \tilde{R} \end{aligned}$$

Die beiden Verschiebungen  $s_{1/2}$  sind stets die EW der  $(2 \times 2)$  Matrix

$$\text{in } H = \begin{bmatrix} \ddots & & & & & \\ & \ddots & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \lrcorner & \lrcorner \\ & & & & \lrcorner & \lrcorner \end{bmatrix} \quad \begin{bmatrix} h_{n-1n-1} & h_{n-1n} \\ h_{nn-1} & h_{nn} \end{bmatrix}$$

evtl.  $s_1 = s$   $s_2 = \tilde{s}$  (konjugiert komplexes Pärchen)  
 $\implies P = H^2 - (2\text{Res})H + |s|^2 I$  (reelle Matrix)

**Idee**

für  $H_k$  definiere  $Q_k$  als Q-Faktor von  $P_k = (H_k - s_1 I)(H_k - s_2 I)$

$$H_{k+1} := Q_k^\top H_k Q_k$$

**Exaktes Vorgehen**

- (1) Berechnung der 1. Spalte von  $P_k$
- (2) Berechnen der Spiegelung  $S_1$  die  $S_1 q = |q| e_1$
- (3)  $\hat{H} := S_1 H_k S_1$
- (4) Transformiere  $\hat{H}$  wieder auf Hessenberg Gestalt  $\implies H_{k+1}$

**Aufwand**

- pro Schritt:  $O(n^2)$
- Gesamt:  $O(n^3)$

## 9.5 Vektoriteration

Zur Berechnung von einem Eigenwert und Eigenvektor.

**Potenzmethode**

$$\begin{aligned} x^0 & \text{ Startvektor mit } \|x^0\| = 0 \\ y^{(k)} & := Ax^{(k)} \\ x^{(k+1)} & := t_k y^{(k)} \quad k = 0, 1, \dots \end{aligned}$$

$t_k$  ist Normierungsfaktor:

- (a)  $t_k = \|y^{(k)}\|^{-1}$

$$(b) t_k = \langle Ax^{(k)}, x^{(k)} \rangle^{-1} = \langle y^{(k)}, x^{(k)} \rangle^{-1}$$

**Satz 9.29** Konvergenz

- Sei  $A$  diagonalisierbar
- Sei  $\lambda_1$  der einzige Eigenwert mit

$$\varphi(a) = |\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$$

$\implies \{x^{(k)}\}$  konvergiert gegen  $w^{(1)}$  (Eigenvektor zu  $\lambda_1$ )

Beweis:

$$\begin{aligned} x^{(0)} &= \sum_{i=1}^n \alpha_i w^{(i)} \quad \text{sei } \alpha_1 \neq 0 \\ x^{(k)} &= t_{k-1} A x^{(k-1)} = t_{k-1} t_{k-2} A^2 x^{(k-2)} \\ &= \prod_{j=0}^{k-1} t_j A^k x^{(0)} \\ x^{(k)} &= \left( \prod_{j=1}^k t_j \right) \sum_{i=1}^n \alpha_i \lambda_i^k w^{(i)} \\ &= \lambda_1^k \prod_{j=1}^k t_j (\alpha_1 w^{(1)} + \sum_{i=2}^n \left( \frac{\lambda_i}{\lambda_1} \right)^k w^{(i)}) \\ &= \implies \text{lineare Konvergenz} \quad \left| \frac{\lambda_2}{\lambda_1} \right|^2 \rightarrow 0 \end{aligned}$$

Schnellere Konvergenz und Berechnung von anderen Eigenwerten mittels Inverser Iteration (WIELANDT- Iteration)

$$\text{Sei } \mu \text{ eine Naherung fur } \lambda_k : |\mu - \lambda_k| < |\mu - \lambda_i| \quad \forall i \geq k$$

Benutzen die Potenzmethode fur  $(A - \mu I)^{-1}$

$$\text{Iteration: Lose } \left. \begin{aligned} (A - \mu I)y^{(k)} &= x^{(k)} \\ x^{(k+1)} &= t_k y^{(k)} \end{aligned} \right\} \quad k = 0, 1, \dots$$

$$\text{Ist } |\mu - \lambda_k| \ll |\mu - \lambda_i| \quad \forall i \geq k$$

$$\implies \frac{|\mu - \lambda_k|}{|\mu - \lambda_i|} \ll 1 \quad \implies \text{schnellere Konvergenz}$$

**Bemerkung 9.30**

$\kappa(A - \mu I)$  sehr gro, wenn  $\mu$  gute Eigenwertnaherung

$\implies$  hier trotzdem sehr gute Ergebnisse (Fehler in Richtung des gesuchten Eigenvektors)

notwendig: Losungstechnik fur Gleichungssysteme  $(A - \mu I)$  mittels Gau besser QR- Zerlegung

## 9.6 Der Lanczos Algorithmus für große Eigenwertprobleme

Sei  $A = A^\top \in \mathbb{R}^{n,n}$  mit  $n$  groß und üblicherweise  $A$  schwach besetzt (Tridiagonalisierung entsprechen 9.4.3 ist aus Speicherplatzgründen nicht möglich)

betrachten **Lanczos Prozess**

betrachte  $z^{(0)} \in \mathbb{R}^n$ , mit  $\|z^{(0)}\| = 1$  beliebiger Startvektor

$$\left. \begin{aligned} \tilde{z}^{(k+1)} &= Az^{(k)} - \alpha_k z^{(k)} - \beta_k z^{(k-1)} \\ z^{(k+1)} &= \frac{\tilde{z}^{(k+1)}}{\|\tilde{z}^{(k+1)}\|} \end{aligned} \right\} \quad k = 0, 1, \dots \quad \text{mit } B_0 = 0$$

mit  $\langle z^{(i)}, z^{(j)} \rangle = \delta_{ij}$

$$\begin{aligned} \text{durch: } \alpha_k &= \langle Az^{(k)}, z^{(k)} \rangle \\ \beta_k &= \langle Az^{(k)}, z^{(k-1)} \rangle \\ &= \langle z^{(k)}, Az^{(k-1)} \rangle \\ &= \langle z^{(k)}, \tilde{z}^{(k)} \rangle = \|\tilde{z}^{(k)}\| \end{aligned}$$

analog zu orthogonalen Polynomen folgt

$$\langle z^{(k+1)}, z^{(j)} \rangle = 0 \quad \forall j < k - 1$$

### Bemerkung 9.31

Der Grund für die Analogie zur Konstruktion orthogonaler Polynome ist:

$xp(x)$  symmetrischer Operator bzgl.  $\langle \cdot, \cdot \rangle = \int \dots$

$AZ$  ist symmetrischer Operator bzgl.  $\langle \cdot, \cdot \rangle =$  euklidisches Skalarprodukt

### Bemerkung 9.32

Implizit werden auch hier orthogonale Polynome erzeugt.

$$\begin{aligned} z^{(k+1)} &= p_k(A)z^{(0)} \quad p_k(t) \in \Pi_k \\ \text{z.B.: } p_0(t) &= 1 \\ p_1(t) &= (t - \alpha_0)/\beta_1 \\ \implies \langle z^{(i)}, z^{(j)} \rangle &= \delta_{ij} = \langle p_i(A)z^{(0)}, p_j(A)z^{(0)} \rangle \end{aligned}$$

### Bemerkung 9.33

$\{z^{(0)}, z^{(1)}, \dots, z^{(k)}\}$  bilden eine Orthonormalbasis im sogenannten Krylovraum

$$\mathbb{K}(A, z^{(0)}) = \text{span}(z^{(0)}, Az^{(0)}, A^2z^{(0)}, \dots, A^kz^{(0)})$$

**Fazit:**

$$\begin{aligned}
 Az^{(k)} &= b_{k+1}z^{(k+1)} + a_k z^{(k)} + B_k z^{(k-1)} \quad \forall k = 0, \dots \\
 Z_k &= (z^{(0)} | z^{(1)} | \dots | z^{(k-1)}) \\
 \mathbb{K}_{k-1}(A, z^{(0)}) &= \text{span}(z^{(k)}) \\
 AZ_k &= Z_k T_k + (0 | 0 \dots | B_k z^{(k)}) \quad \text{mit } T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{n-1} & \\ & & & \beta_{n-1} & \alpha_{n-1} \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 AZ_k &= Z_k T_k + \beta_k z^{(k)} e_k^\top \quad \text{und } Z_k^\top Z_k = I \\
 \longrightarrow Z_k^\top AZ_k &= T_k + \underbrace{\beta_k Z_k^\top z^{(k)} e_k^\top}_0 = T_k
 \end{aligned}$$

• Fall 1:  $k = n$

$z_n = (z^{(0)} \dots z^{(n-1)})$  Orthonormalbasis im  $\mathbb{R}^n$

jetzt  $Z_n$  orthogonale ( $n \times n$ ) Matrix

$\longrightarrow Z_n^\top AZ_n = T_n$   $T_n$  ähnlich zu  $A$

$T_n$  und  $A$  haben gleiche Eigenwerte und die Eigenwerte von  $T_k$  sind gut berechenbar z.B. mittels QR- Algorithmus für Tridiagonalmatrizen

$\implies$  Algorithmus ist nicht zu gebrauchen da obige Identität dadurch gestört, daß die  $\{z^{(k)}\}$  ab gewissen  $k$  kein exaktes Orthonormalsystem bilden

• Fall 2: Lanczos Phänomen

– schon für kleine  $k$  sind einige Eigenwerte von  $T_k$  gute Näherungen für die Eigenwerte von  $A$

– Instabilitäten werden insbesondere durch gut konvergierte Eigenwerte bewirkt

### Algorithmus 9.34 Lanczos Verfahren

- Bilde  $z^{(0)} \dots z^{(k)}$  und somit  $Z_k = (z^{(0)} | \dots | z^{(k-1)})$  und  $\alpha_i$  und  $\beta_i$  bilden obige Tridiagonalmatrix  $T_k$
- Berechnen Eigenwerte und Eigenvektoren von  $T_k$ ,  $T_k u = \delta u$   
 $\longrightarrow \langle \delta, Z_k u \rangle$  Näherung für ein Eigenpaar von  $A$  mit  $\|u\| = 1$

- Berechnen Residuum:

$$\begin{aligned} \|A(Z_k u) - \delta(Z_k u)\| &= \|Z_k T_k u + \beta_k z^{(k)} e_k^\top u - \delta Z_k u\| \\ &= \beta_k |e_k^\top u| \end{aligned}$$

$\implies B_k$  und die letzte Komponente des Eigenvektors von  $T_k$  bringen die Information ob der Eigenwert „konvergiert“ hat

also: - Berechnen von Eigenwerten und der letzten Komponenten der Eigenvektoren von  $T_k$

- Tests ob  $\beta_k |e_k^\top u|$  klein

$\implies \langle \delta, Z_k u \rangle$  akzeptables Eigenpaar von  $A$

- löse  $(T_k - \delta I)u = 0$  (mittels inverser Iteration)

- berechne  $Y = Z_k u$  als Eigenvektor von  $A$
- jetzt  $k:=k+1$   
+ weitere Lanczosvektoren  $z^{(i)}$  bzgl.  $y$  nachorthogonalisieren

### Bemerkung 9.35

Welche Eigenwerte von  $A$  spalten sich schnell ab ?

etwa solche die „ausen“ oder gut vom Restspektrum abgetrennt liegen