Numerik MA0008: Zusammenfassung

Jonas Treplin

February 19, 2023

1 Grundlagen

Theorem 1 (Satz von Gerschgorin) *Sei $(a_{ij}) = A \in \mathbb{R}^{n\times n}$ dann sind die Eigenwerte von A enthalten in $\bigcup_{i=1}^{n} S_i \subset \mathbb{C}$, dabei sind die $S_i := K(a_{ii}, \sum_{j=1, i\neq j}^{n} |a_{ij}|)$. Wobei mindestens ein Eigenwert jeder Zusammenhangskomponente zugeordnet ist.*

2 Matrixfaktorisierung

2.1 Lösen einfacher Matrizen

Theorem 2 *Die Permutationsmatrizen, die unitären Matrizen, die invertierbaren Matrizen und die unteren/oberen Dreiecksmatrizen bilden jeweils unter Matrixmultiplikation eine Gruppe. Insbesondere sind ihre Inverse von der selben Klasse von Matrizen.*

Gleichungssysteme für die unitären Matrizen (und damit auch Permutationsmatrizen) lassen sich einfach durch adjungieren lösen. Für untere und obere Dreiecksmatrizen existieren Vorwärts- und Rückwärtssubsitution. Diese sind aus dem Endschritt des Lösens von Gleichungssystemen mit dem Gauß-Algorithmus bekannt.

Algorithm 1 Vorwärtssubsitution (Lösen einer unteren Dreiecksmatrix)

Require: $(l_{ij}) = L \in \mathbb{R}^{n\times n}$ Untere Dreiecksmatrix, $b \in \mathbb{R}^n$.
  for $i \in 1 : n$ **do**
    $x_i \leftarrow \frac{1}{l_{ii}}(b_i - \sum_{j=1}^{i-1} l_{ij} * x_j)$
  end for

Algorithm 2 Rückwärtsssubsitution (Lösen einer oberen Dreiecksmatrix)

Require: $(u_{ij}) = U \in \mathbb{R}^{n\times n}$ Obere Dreiecksmatrix, $b \in \mathbb{R}^n$.
  for $i \in n : 1$ **do**
    $x_i \leftarrow \frac{1}{u_{ii}}(b_i - \sum_{j=i+1}^{n} u_{ij} * x_j)$
  end for

2.2 LU-Zerlegung

Glücklicherweise kann jede invertierbare Matrix (fast eindeutig) in solche Matrizen zerlegt werden. Dies geschieht Wahlweise durch eine LU-Zerlegung oder eine QR-Zerlegung. Mit Pivots erreicht man, dass jede invertierbare Matrix

Algorithm 3 LU-Zerlegung ohne Pivots

Require: $(a_{ij}) = A \in GL(n)$
  for $i \in 1 : n$ **do**
    for $j \in i + 1 : n$ **do**
      $a_{ji} \leftarrow \frac{a_{ji}}{a_{ii}}$
      for $k \in i + 1 : n$ **do**
        $a_{jk} \leftarrow a_{jk} - a_{ji}a_{ji}$
      end for
    end for
  end for
  $U = triu(A)$
  $L = tril(A, -1) + Id$

$A \in GL(n)$ zerlegt werden kann, sodass $PA = LU$. Dazu wählt man in jedem Schritt i die Zeile $j = \arg\max_{j \geq i} |a^i_{ji}|$ und vertauscht diese mit der i-ten Zeile.

Algorithm 4 LU-Zerlegung mit Pivots

Require: $(a_{ij}) = A \in GL(n)$
  $P = Id$
  for $i \in 1 : n$ **do**
    $p \leftarrow \arg\max_{i \leq k \leq n} |a_{ki}|$
    $P, A, \leftarrow P^{(p,i)}P, P^{(p,i)}A$
    for $j \in i + 1 : n$ **do**
      $a_{ji} \leftarrow \frac{a_{ji}}{a_{ii}}$
      for $k \in i + 1 : n$ **do**
        $a_{jk} \leftarrow a_{jk} - a_{ji}a_{ji}$
      end for
    end for
  end for
  $U = triu(A)$
  $L = tril(A, -1) + Id$

Theorem 3 (LU-ZErlegung mit Pivots) *Sei $A \in GL(n)$. Dann existieren eine eindeutige Permutationsmatrix P, sowie untere (obere) Dreiecksmatrix L (U, sodass $PA = LU$. Dabei ist L normiert also $l_{ii} = 1$.*

Theorem 4 *Für s.p.d. Matrizen sowie spalten-/zeilendiagonaldominante Matrizen ist keine Pivotsuche notwendig.*

Theorem 5 (Cholesky-Zerlegung) *Sei A symmetrisch positiv definit dann lässt sich eine nicht normierte untere Dreiecksmatrix $\tilde{L}$ finden, sodass $A = \tilde{L}\tilde{L}^T$.*

Algorithm 5 Berechnung der Cholesky Zerlegung

Require: A s.p.d.
$L, U \leftarrow$ LU_Zerlegung(A)
$D = (u_{ii})$ Diagonalmatrix.
$\tilde{L} = \sqrt{D}L$.

2.3 QR-Zerlegung

Eine weitere Möglichkeit ist die der QR Zerlegung.

Definition 1 (Givensrotation) *Für ein* $a \in \mathbb{R}^2$ *sei* $Q = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$. *Wobei* $c = \frac{1}{\sqrt{1+\tau^2}}$ *und* $s = c\tau$ *mit* $\tau = \frac{v_2}{v_1}$ *wenn* $|v_1| \geq |v_2|$ *und* $s = \frac{1}{\sqrt{1+\tau^2}}$ *und* $c = s\tau$ *mit* $\tau = \frac{v_1}{v_2}$ *wenn* $|v_1| < |v_2|$.

Diese Fallunterscheidung ist so gewählt, dass $||Q|| \leq 1$, damit sich Rundungsfehler nicht akkumulieren.

Theorem 6 (Givens-Rotation) *Es gilt* $Qa = \xi e_1$.

Algorithm 6 QR-Zerlegung mit Givens-Rotationen

Require: $A \in \mathbb{R}^{n \times m}$
$Q \leftarrow I_n$
for $i \in [n]$ **do**
for $j \in (n, ..., i+1)$ **do**

$$G := \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & s & & \\ & & -s & c & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \leftarrow Givens(\begin{bmatrix} A_{j-1,i} \\ A_{j-1,i} \end{bmatrix})$$

$Q \leftarrow GQ$
$A \leftarrow GA$
end for
end for
$Q \leftarrow Q*$

Definition 2 (Householder Spiegelung) *Die Householder Spiegelung für einen Vektor* $a \in \mathbb{R}^n$ *ist:*

$$Q := Id - \frac{2}{v^T v} v v^T$$

Wobei $v := a + sign(a_1)||a||e_1$ *Sie erfüllt ebenfalls* $Qa = \alpha e_1$

Die QR-Zerlegung ist der LU-Zerlegung hinsichtlich numerischer Stabilität überlegen, besonders bei Betrachtung der Wilkinson-Matrix:

Algorithm 7 QR-Zerlegung mit Householder Rotationen

Require:
Require: $A \in \mathbb{R}^{n \times m}$
  $Q \leftarrow I_n$
  for $i \in [n]$ **do**

$$H \leftarrow \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & \text{Householder}((a_i, ..., a_n)) \end{bmatrix}$$

    $Q \leftarrow HQ$
    $A \leftarrow HA$
  end for
  $Q \leftarrow Q*$

Definition 3 (Wilkinson-Matrix) *Die Wilinson-Matrix ist definiert als:*

$$W_n := \begin{bmatrix} 1 & 0 & ... & 0 & 1 \\ -1 & \ddots & ... & \vdots & \vdots \\ \vdots & & & \ddots & \vdots \\ -1 & ... & ... & \dots & 1 \end{bmatrix}$$

Ein besonders instabiler Lösungsvektor ist $b_n = \begin{bmatrix} 0 \\ \frac{1}{n} \\ \vdots \\ \frac{n-2}{n} \\ 1 \end{bmatrix}$

3 Fehlerrechnung

Definition 4 (Fehlermaße) *Wir definieren für eine Tupel* $T = (A_1, A_2, ..., A_n)$ *mit Störung* $\tilde{T} = (A_+E_1, ..., A_n + E_n)$*:*

- *das absolute Fehlermaß:*

$$[[E]]_{abs} := \max ||E_i||$$

- *das relative Fehlermaß:*

$$[[E]]_{rel} := \max \frac{||E_i||}{||A_i||}$$

Definition 5 (Maschinenepsilon) *Das* ***Maschinen-***ϵ *is der relative Fehler, der bei Addition und Multiplikation von SKalaren auftritt. Er liegt für IEEE double-precision bei ca.* 10^{-16}

Definition 6 (Kondition) *Die Kondition einer Abbildung* f *im Punkt* x *ist definiert als:*

$$\kappa(f, x) = \limsup_{y \to x} \frac{[[f(y) - f(x)]]}{||y - x||}$$

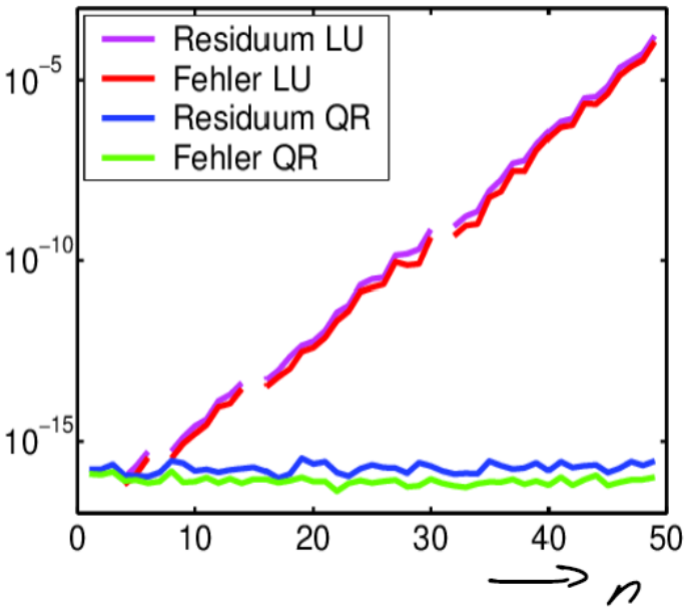


Figure 1: Fehler beim Lösen von $W_n x = b_n$

Man unterscheidet zwischen:

- ***gut konditionierten*** *Problemen:* $\kappa(f,x)$ $O(1)$
- ***schlecht konditionierten*** *Problemen:* $\kappa(f,x) >> 1$
- ***schlecht gestellten*** *Problemen:* $\kappa(f,x) = \infty$

Theorem 7 *Die Kondition einer linearen Gleichung* $Ax = b$ *hängt nur von* A *ab und ist:*

$$\kappa(A) = \|A\|\|A^{-1}\|$$

Theorem 8 (Kondition einer C^1-Funktion) *Sei* $f \in C^1(D), D \subset \mathbb{R}^n$ *mit* $f(x) \neq 0$*, dann gilt für die realtive Kondition:*

$$\kappa_{rel}(f,x) = \frac{\sum_{i=1}^{n} |\partial_i f(x)||x_i|}{|f(x)|}$$

Definition 7 (Vorwärts- und Rückwärtsfehler) *Sei* $\hat{f}$ *eine numerische Näherung der Funktion* f*, dann ist an* x*:*

1. *Der* ***Vorwärtsfehler****:*
$$[[f(x) - \hat{f}(x)]]$$
2. *Der* ***Rückwärtsfehler****:*
$$\inf_{\Delta x}\{[[\Delta x]] | f(x + \Delta x) = \hat{f}(x)\}$$

Definition 8 (Stabilität) *Ein Algorithmus $\hat{f}$ zur Näherung von f heißt* ***stabil*** *an x bezüglich des Fehlermaßes $[[.]]$ falls ein $\tilde{x}$ existiert mit:*

$$[[x - \tilde{x}]] = O(\epsilon_m)$$

$$[[\hat{f}(x) - f(\tilde{x})]] = O(\epsilon_m)$$

Dabei sei ϵ_m das Maschinenepsilon.

Definition 9 (Rückwärtsstabilität) *Eine Nöherung $\hat{f}$ zu f heißt* ***rückwärtsstabil*** *an an x bezüglich des Fehlermaßes $[[.]]$ falls ein $\tilde{x} = x + \Delta x$ existiert mit:*

$$[[x - \tilde{x}]] = O(\epsilon_m)$$

$$\hat{f}(x) = f(\tilde{x})$$

Theorem 9 (Fehler rückwärtsstabiler Algorithmen) *Der Fehler eines Rückwärtsstabilen Algorithmus $\hat{f}$ hängt nur von der Konidion von f ab:*

$$[[\hat{f}(x) - f(x)]] = [[f(\tilde{x}) - f(x)]] \leq c\kappa(f, x)\epsilon_m$$

Theorem 10 (Rückwärtsfehler beim Lösen linearer Gleichungssysteme) *Sei $Ax = b$ ein zu lösendes Gleichungssystem. Definiere*

$$w(\tilde{x}) = \inf_E \{\frac{\|E\|_2}{\|A\|_2} : (A + E)\tilde{x} = b\}$$

als den relativen Rückwärtsfehler bezüglich der Matrix A. Dann gilt:

$$w(\tilde{x}) = \frac{\|b - A\tilde{x}\|_2}{\|A\|_2\|\tilde{x}\|_2}$$

Theorem 11 (Stabilität der Matrixmultiplikation) *Sei $MN = A$ und $\tilde{M}, \tilde{N}$ fehlerbehaftet mit relativen Fehler kleiner als (δ), dann ist:*

$$\|\tilde{M}\tilde{N} - A\| \leq (2\delta + \delta^2)\|M\|\|N\|$$

. Daraus folgt, dass die QR-Zerlegung stabil bezüglich des relativen Fehlermaßes ist:

$$\frac{\|\tilde{Q}\tilde{R} - A\|}{\|A\|} \leq \frac{1\|A\|}{\|A\|} = 1$$

Ähnliches gilt auch für die Cholesky-Zerlegung, jedoch nciht für die LU-Zerlegung

Algorithm 8 Nachiteration

Require: Gleichungssystem $Ax = b$
  Zerlege $LU = PA$
  Berechne $LUx^{(0)} \tilde{=} Pb$ durch Vorwärts-/Rückwärtsssubsitution
  for $i = 1, ..., k$ **do**
    Berechne $LU\Delta x^{(i)} = P(b - Ax^{(i-1)})$
    $x^{(i)} \leftarrow x^{(i-1)} + \Delta x^{(i)}$
  end for

4 Ausgleichsrechnung

Definition 10 (Lineares Ausgleichsproblem) *Sei $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^n$ mit $m \leq n$. Gesucht ist $x \in \mathbb{R}^m$. Sodass*

$$||Ax - b||_2$$

minimiert wird. Es wird im Generellen angenommen, dass $Rang(A) = m$.

Definition 11 (Normalengleichung) *Gegeben ein Ausgleichsproblem A, b, nennt man:*

$$A^T Ax = A^T b$$

die Normalengleichung.

Theorem 12 (Lösung des Linearen Ausgleichproblems mittels Normalengleichung) *Die Lösung der Normalengleichung ist das eindeutige gesuchte Minimum des Ausgleichsproblems.*

Algorithm 9 Lösen des Ausgleichproblems mittels Normalengleichung

Require: Ausgleichsproblem $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^m, m \leq n$
$L \leftarrow$ Cholesky$(A^T A)$
Löse $LL^T x = A^T b$ durch Vorwärts und Rückwärtsssubsitution.

Die Stabilität des Algorithmus hängt von der Stabilität der Matrixmultiplikation $A^T A$ ab. Falls $\kappa(A^T A)$ groß ist und $||Ax - b||$ klein treten hier Stabilitätsprobleme auf. Um diesen entgegen zu wirken, kann man die Orthogonalisierungsmethode verwenden.

Algorithm 10 Lösen des Ausgleichproblems mittels QR-Methode

Require: Ausgleichsproblem $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^m, m \leq n$
Zerlege $A = Q\hat{R}$ mit $\hat{R} = \begin{bmatrix} R \\ 0 \end{bmatrix}$ wobei $R \in \mathbb{R}^{m \times m}$ obere Dreiecksmatrix sei.
Löse $Rx = (Q^T b)_1$ wobei $(.)_1$ die ersten m Elemente seien.

Theorem 13 (Aufwand der Lösungsmethoden) *Der Aufwand beträgt:*

1. *Normalengleichung: $nm^2 + \frac{m^3}{3}$.*
2. *QR-Methode: $2nm^2 - 2\frac{m^3}{3}$*

Für $n >> m$ ist also die QR-Methode doppelt so teuer wie der Ansatz der Normalengleichung.

Theorem 14 (Kondition der Normalengleichung) *Sei $A \in \mathbb{R}^{n \times m}$ und R wie in der Zerlegung für die QR-Methode. Es gilt:*

$$\kappa(A^T A) = \kappa(R)^2$$

5 Eigenwertapproximation

5.1 Vektoriteration

Eine Einfache Idee um einen einzelnen Eigenwert mit Eigenwert einer Matrix $A \in \mathbb{R}^{n\times n}$ zu bestimmen, ist die Vektoriteration. Diese beruht darauf, dass Eigenräume hoffentlich anziehende Fixpunkte sind.

Algorithm 11 Vektoriteration

Require: $A \in \mathbb{R}^{n\times n}$ und Startvektor $x^{(0)} \in \mathbb{R}^n$.
  for $i = 1, 2, 3, ...$ **do**
    $y^{(i)} \leftarrow Ax^{(i-1)}$
    $\lambda^{(i-1)} \leftarrow (x^{(i-1)})^T y^{(i)}$
    $x^{(i)} \leftarrow \frac{y^{(i)}}{||y^{(i)}||}$
  end for

Theorem 15 (Konvergenz der Vektoriteration) *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch mit EW $|\lambda_n| \leq ... \leq |\lambda_2| < |\lambda_1|$ und EV $(v_i)_{i\in[n]}$ und $\beta_1 := v_1^T x^{(0)} \neq 0$. Dann gilt:*

$$|\lambda^{(i)} - \lambda_1| \leq C_1(x^{(0)})|\frac{\lambda_2}{\lambda_1}|^{2i}$$

$$\|sign(\lambda_1)^i x^{(i)} - sign(\beta_1)v_1\|_2 \leq C_2(x^{(0)})|\frac{\lambda_2}{\lambda_1}|^i$$

und C_1, C_2 Konstanten sind die vom Startwertabhängen. Ist A nicht symmetrisch aber noch normal gilt nur noch Konvergenz:

$$|\lambda^{(i)} - \lambda_1| \leq C_1(x^{(0)})|\frac{\lambda_2}{\lambda_1}|^i$$

Durch diese Methode lässt sich nur ein einzelner Eigenvektor und auch nur der zum Betragsmäßig größten Eigenwert bestimmen. Um andere Eigenvektor zu berechnen, nutzen wir, dass der Betragsmäßig größte Eignewert von $(A-\mu I)^{-1}$ der Kerwehrt des nächsten Egenwerts an μ ist.

Algorithm 12 Inverse Vektoriteration

Require: $A \in \mathbb{R}^{n\times n}$ und Startvektor $x^{(0)} \in \mathbb{R}^n$, sowie shift $\mu \in \mathbb{R}$.
  for $i = 1, 2, ...$ **do**
    $(A - \mu I)\omega^{(i)} = x^{(i-1)}$
    $\eta \leftarrow \|\omega^{(i)}\|_2$
    $x^{(i)} \leftarrow \omega^{(i)}/\eta$
    $\rho \leftarrow {x^{(i)}}^T x^{(i-1)}/\eta$
    $\lambda^{(i)} \leftarrow \mu + \rho$
  end for

Der Aufwand hängt hier davon ab, wie oft der Shiftparameter μ neu berechnet wird, dann ist jedes Mal eine LU-Zerlegung mit $O(n^3)$ Schritten nötig. Ansonsten kostet jeder Schritt $O(n^2)$ Operation hauptsächlich für Vorwärts und Rückwärtsssubsitution.

5.2 QR-Iteration

Um alle Eigenwerte gleichzeitig zu bestimmen kann iterativ eine Schur-Zerlegung berechnet werden. Dies geschieht mit der QR-Iteration.

Algorithm 13 QR-Iteration ohne Shift

Require: $A \in \mathbb{R}^{n \times n}$
  $A_0 \leftarrow Q_0^T A Q_0$ mit $Q \in U(n)$.
  for $i = 1, 2, 3, ...$ **do**
    Bestimme $Q_i R_i = A_{i-1}$
    $A_i \leftarrow R_i Q_i$
  end for

Der Algorithmus beruht auf der Tatsache, dass $RQ \sim A$ also die gleichen EW wie A hat.

Theorem 16 (Konvergenz der QR-Iteration) *Sei $A \in \mathbb{R}^n$ symmetrisch und mit EW $|\lambda_1| > |\lambda_2| > ... > |\lambda_m| > 0$. Sei $A = V \Lambda V^T$ diagonalisiert mit V orthogonal. Besitzt $(Q_0^T V)^{-1}$ eine normierte LU-Zerlegung, dann gilt:*

1. $\lim_{k \to \infty} |(Q_k)_{ij}| = \delta_{ij}$
2. $\lim_{k \to \infty} |(R_k)_{ij}| = \delta_{ij} |\lambda_i|$
3. $\lim_{k \to \infty} (A_k)_{ij} = \delta_{ij} \lambda_i$

Der Beweis kann auf schwächere Bedingungen ausgweitet werden zum Beispiel sind mehrfache Eigenwerte auch erlaubt. Häufig kann die QR-Iteration auch auf nicht-symmetrische Matrizen angewendet werden, wenn sie in diesem Fall konvergiert dann allerdings nicht mehr gegen eine Diagonal-, sondern obere Dreiecksmatrix.
Der erste Schritt die Transformation mit Q_0 transformiert die Matrix in eine obere Hessenberg Matrix. Für eine Obere Hessenberg-Matrix ist die QR-Zerlegung in $O(n^2)$ berechenbar. Da auch ein einzelner Iterationschritt die Hessenberg Form erhält beschleunigt dies das Verfahren enorm.

Algorithm 14 Berechnung von Q_0

Require: $A \in \mathbb{R}^{n \times n}$
  $Q_0 \leftarrow Id$
  for $i = 1, ..., n-1$ **do**
    for $j = n, i+2$ **do**

$$G := \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & s & & \\ & & -s & c & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \leftarrow Givens(\begin{bmatrix} A_{j-1,i} \\ A_{j,i} \end{bmatrix})$$

    $A \leftarrow GAG^T$
    $Q_0 \leftarrow GQ_0$
    end for
  end for

Weiterhin kann ein Shiftparameter eingeführt werden um das Verfahren zu beschleunigen idealerweise liegt dieser Shiftparameter möglichst nah an einem Eigenwert der Matrix A.

Algorithm 15 QR-Iteration mit Shift

Require: $A \in \mathbb{R}^{n \times n}$
  $A_0 \leftarrow Q_0^T A Q_0$ mit $Q \in U(n)$.
  for $i = 1, 2, 3, ...$ **do**
    $\mu_i \leftarrow \text{Shift}(A_i)$
    Bestimme $Q_i R_i = A_{i-1} - \mu_i I$
    $A_i \leftarrow R_i Q_i + \mu_i I$
  end for

Dabei können verschiedene Shiftstrategien angewandt werden:

1. **Rayleighquotienten-Shift**: $\mu_i := (A_i)_{n,n}$
2. **Wilikinson-Shift**: Bestimme die Eigenwerte $\hat{\lambda}_1, \hat{\lambda}_2$ von $\begin{bmatrix} (A_i)_{n-1,n-1} & (A_i)_{n-1,n} \\ (A_i)_{n,n-1} & (A_i)_{n,n} \end{bmatrix}$.
   Wähle $\mu_i = \arg\max_{\mu \in \{\hat{\lambda}_1, \hat{\lambda_2}\}} \|\mu - (A_i)_{n,n}|$
3. **Random-Shift**: Wählre zufällig einen Shift aus.

Der Rayleighshift produziert bei $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ als shift 0 und landet somit in einem Deadend. Ähnlich schlägt der WIlkinson-Shift bei $A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ fehl.
In diesen Situationen kommt der Random-Shift zum Einsatz.
Weiterhin kann das Verfahren beschleunigt werden, indem man, sobald $|(A_i)_{n,n-1}| \leq$ TOL nur noch mit $(A_i)_{1:m-1,1:m-1}$ weiterrechnet. Diese Strategie heißt **Deflation**.
Um aus den berechneten Eigenwerten Eigenvektoren zu erhalten, kann man für wenige Eigenwerte die Inverse Vektoriteration mit Shift verwenden. Alternativ lassen sich die aus der QR-Zerlegung berechneten Q_i verwenden.

Theorem 17 (Eigenvektoren einer oberen Dreiecksmatrix) *Sei $R \in \mathbb{R}^n$ obere Dreiecksmatrix. Definiere:*

$$R^{(i)} := (R)_{1:i-1,1:i-1}$$

$$\omega := (R)_{1:i-1,i}$$

Dann hat der i-te Eigenvektor von R die Form:

$$v_i = \begin{pmatrix} (R^{(i)})^{-1}\omega \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Theorem 18 (Eigenvektoren der QR-Zerlegung) *Seien $(Q_i)_{i\in[k]}$ die berechneten Q_i einer QR-Iteration auf A mit Q_0 der Umformungsmatrix für die Hessenberg-Form. Dann ist*

$$A_k = Q_k^T ... Q_1^T Q_0^T A Q_0 Q_1 ... Q_k$$

Wir definieren $\hat{Q}_k := Q_0 Q_1 ... Q_k$. Bei genügend hohem k sollte A_k fast gleich einer oberen Dreiecksmatrix $\hat{R}_k$ sein. Deren Eigenvektoren lassen sich berechnen nach dem obigen Satz. Dann sind die $\hat{Q}_k v_i$ die Eigenvektoren von A.

Theorem 19 (Kondition des Eigenwertproblems für beliebige Matrizen) *Sei $A \in \mathbb{R}^{n\times n}$. Dann exisitiert eine Konstante C_A sodass für jede Störung ΔA:*

$$\forall \mu \in \sigma(A + \Delta A) \exists \lambda_\mu \in \sigma(A) : |\mu - \lambda_\mu| \leq C_A \max\{\|\Delta A|_2, \|\Delta A\|_2^{\frac{1}{n}}\}$$

Theorem 20 (Bauer-Fike, Kondition des Eigenwertproblems für diagonalisierbare Matrizen) *Sei $A \in \mathbb{R}^{n\times n}$ reell diagonalisierbar durch $T^{-1}DT$, dann existiert für jede Störung ΔA und jedes $\mu \in \sigma(A + \Delta A)$ ein $\lambda_\mu \in \sigma(A)$ mit*

$$|\mu - \lambda_\mu| \leq \|T\|_p \|T^{-1}\|_p \|\Delta A\|_p$$

Für alle $p \in [1, \infty]$.

6 Interpolation

Definition 12 (Interpolation) *Gegeben Stützpunkte $(x_1, f_1), ..., (x_n, f_n)$ finde f in einer bestimmten Klasse von Funktionen, sodass $f(x_i) = f_i \quad \forall i$*

Die bekanntesten Formen von Interpolation sind:

1. **Polynominterpolation**: $f \in \mathbb{P}^n$
2. **Spline-Interpolation**: f ist stückweise polynomiell und gesamt C^l.
3. **Rationale Interpolation**: $f \in R_{k,l}\{\frac{\sum_{i=0}^k a_i x^i}{\sum_{i=0}^l b_i x^i}\, a_i, b_j \in \mathbb{R}\}$
4. **Trigonometrische Interpolation**: $f \in T_n := \{b_0 + \sum_{k=1}^n a_k sin(2\pi kx) + \sum_{k=1}^n b_k sin(2\pi kx)\}$

6.1 Polynominterpolation

Definition 13 (Lagrange-Polynom) *Zu den $n+1$ verschiedenen Stützstellen $x_0, ..., x_n$ sind die Lagrange-Polynome von Grad n definiert als:*

$$L_i(x) := \prod_{k=0, k\neq i}^{n} \frac{x - x_k}{x_i - x_k}$$

Theorem 21 (Polynominterpolation) *Das Interpolationspolynom π_n zu den Stützstellen $x_0, ..., x_n$ ist eindeutig definiert als:*

$$\pi_n(x) = \sum_{i=0}^{n} f_i L_i(x)$$

Theorem 22 *Die Lagrange Polynome erfüllen die Partition der* 1*:*

$$\sum_{i=0}^{n} L_i(x) = 1$$

Definition 14 (Vandermonde-Matrix) *Die Matrix* $V_n \in \mathbb{R}^{n+1\times n+1}$ *mit Einträgen:*

$$(V_n)_{i,j} = x_{i-1}^{j-1}$$

Heißt Vandermonde-Matrix zu den Stützstellen $x_0, ..., x_n$*.*

Theorem 23 *Es gilt:*

$$\det V_n = \prod_{0\le i<j\le n} (x_j - x_i)$$

Theorem 24 *Das Interpolationspolynom ist auch durch* $\sum_{k=0}^{n} a_k x^k$ *gegeben wobei* $\begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} =: a = V_n \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix}$

Definition 15 (Knotenpolynom) *Zu den* $n+1$ *Stützstellen* $x_0, ..., x_n$ *definiert man das Knotenpolynom*

$$\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$$

Theorem 25 *Falls der Interpoland* f *aus* C^{n+1} *kommt ist der Interpolationsfehler beschränkt durch:*

$$\|f - \pi_n\| \le \frac{\|\omega_{n+1}\|}{(n+1)!}\|f^{(n+1)}\|$$

Theorem 26 *Es gilt für jede beliebige Wahl von Punkten:*

$$\|\omega\| \ge 2^{-n}$$

Bei äquidistanten Stützstellen treten beim Interpolieren von $f(x) = \frac{1}{1+25x^2}$ große Fehler in den Randbereichen auf. Generell lässt sich keine generell optimale Knotenfolge finden. Allerdings kann $\|\omega_n\|$ minimiert werden. Dies geschieht durch die Chebychev-Punkte:

Definition 16 *Die Chebychev-Polynome sind definiert als:*

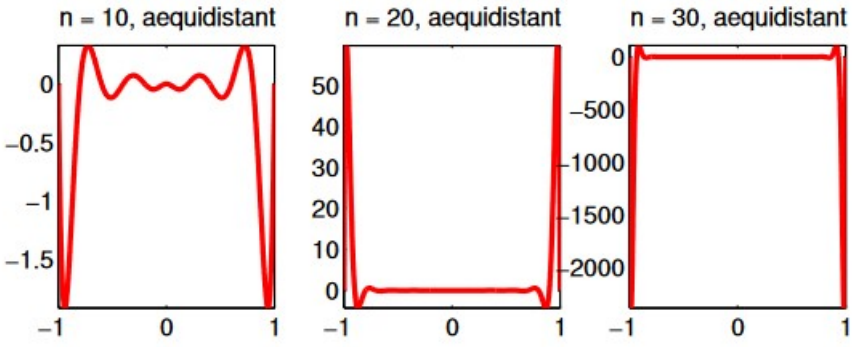
$$T_n(X) := \cos(n \arccos x)$$

Sie sind orthogonal bezüglich der Gewichtsfunktion $\frac{1}{\sqrt{1-x^2}}$*. Sie besitzen die Nullstellen:*

$$x_i^{(n+1)} = \cos(\frac{2i+1}{2n+2}\pi)$$

Diese werden auch als Chebychev-Punkte bezeichnet.

Figure 2: Interpolations Fehler für π_n



Theorem 27 (Dreiterm-Rekursion der Chebychev-Polynome) $T_0 = 1, T_1(x), T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$

Theorem 28 *Für das Knotenpolynom $\omega_{n+1,cheb}$ zu den Chebychevpunkten gilt:*

$$\|\omega_{n+1,cheb}\| = 2^{-n}$$

Die theoretischen Überlegungen zur Bestimmung des Interpolationspolynom sind numerisch nicht stabil.

6.2 Auswertungsschemata

Algorithm 16 Auswertungschema von Aitken-Neville

Require: $n+1$ Stützstellen $(x_0, f_0), ..., (x_n, f_n)$, Auswertungsstelle y.
  Setze $P_i(y) \leftarrow f_i$ für $i = 0, ..., n$
  for $i = 1, ..., n$ **do**
    for $j = 0, ..., n - i$ **do**
      $P_{j...j+i}(y) \leftarrow \frac{(y-x_j)P_{j+1...j+i}(y)-(y-x_{j+i})P_{j...j+i-1}(y)}{x_{j+i}-x_j}$
    end for
  end for

Figure 3: Darstellung des Aitken-Neville-Schema

Aitken–Neville-Schema für $n=2$, $n=3$, $n=4$ und $n=5$:

	$k=0$	1	2	3	4	5
x_0	$f_0 = P_0(x)$					
		$P_{01}(x)$				
x_1	$f_1 = P_1(x)$		$P_{012}(x)$			
		$P_{12}(x)$		$P_{0123}(x)$		
x_2	$f_2 = P_2(x)$		$P_{123}(x)$		$P_{01234}(x)$	
		$P_{23}(x)$		$P_{1234}(x)$		$P_{012345}(x)$
x_3	$f_3 = P_3(x)$		$P_{234}(x)$		$P_{12345}(x)$	
		$P_{34}(x)$		$P_{2345}(x)$		
x_4	$f_4 = P_4(x)$		$P_{345}(x)$			
		$P_{45}(x)$				
x_5	$f_5 = P_5(x)$					

Theorem 29 (Aufwand des Aitken-Neville-Verfahren) *Für l verschiedene Funktionen, m Auswertungsstellen und $n+1$ Stützstellen beträgt der Aufwand für das Aitken-Neville-Schema $O(lmn^2)$.*

Das Aitken-Neville Schema ist stabil aber langsam, wenn mehrere Stellen ausgwertet werden sollen. Um dies zu beschleunigen kann das

Definition 17 (Newton-Darstellung eines Polynoms) *Die Darstellung eines Polynoms in der Form:*

$$\pi_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + ... + c_n(x - x_0)...(x - x_{n-1})$$

*Heißt **Newton'sche Darstellung**.*

Algorithm 17 Newton'sche Dividierte Differenzen

Require: $n+1$ Stützstellen $(x_0, f_0), ..., (x_n, f_n)$
  Setze $f[x_i] := f_i$ für $i = 0, ..., n$
  for $i = 1, ..., n$ **do**
    for $j = 0, ..., n - i$ **do**
      $f[x_j ... x_{j+i}] \leftarrow \frac{f[x_{j+1} ... x_{n-i}] - f[x_j ... x_{n-i-1}]}{x_{j+i} - x_j}$
    end for
  end for
  $c_i \leftarrow f[x_0 ... f_i]$

Algorithm 18 Horner-Schema

Require: Polynom in Newton-Darstellung mit Koeffizienten $c_0, ..., c_n$, Auswertungsstelle y.
  $p \leftarrow c_n$
  for $k = n - 1, ..., 0$ **do**
    $p \leftarrow p(y - x_k) + c_k$
  end for

Theorem 30 (Aufwand des Horner Schema) *Für l verschiedene Funktionen, m Auswertungsstellen und $n+1$ Stützstellen beträgt der Aufwand für das Aitken-Neville-Schema $O(l(mn+n^2))$.*

Allerdings ist das Horner-Schema **nicht stabil** für große n. Der Zugang über die baryzentrische Darstellung bietet dem Abhilfe und noch größere Effizienz.

Definition 18 (baryzentrische Gewichte) *Wir definieren die **baryzentrischen Gewichte** zu den Stützstellen $x_0, ..., x_n$ als:*

$$\lambda_i := \frac{1}{\prod_{k=0,k\neq i}^{n}(x_i - x_k)} = \frac{1}{\omega'_{n+1}(x_i)}$$

Theorem 31 (baryzentrische Darstellung des Lagrange-Polynoms) *Für die Stützstellen $(x_0, f_0), ..., (x_n, f_n)$ hat das Interpolationspolynom die Form:*

$$\pi_n(x) = \omega_{n+1}(x)\sum_{i=1}^{n}\frac{\lambda_i f_i}{x - x_i} = \frac{\sum_{i=1}^{n}\frac{\lambda_i f_i}{x-x_i}}{\sum_{i=1}^{n}\frac{\lambda_i}{x-x_i}}$$

Theorem 32 (Formen der baryzentrischen Gewichte) *Die Gewichte haben folgende Form (abhängig von der Wahl der Stützstellen):*

- *Äquidistant: ($x_i = -1 + ih, h = 2/n$)*

$$\lambda_i = \frac{(-1)^{n-i}}{h^n(n-i)!i!}$$

- *Chebychev: ($x_i = \cos(\frac{2i+1}{2n+2}\pi)$)*

$$\lambda_i = (-1)^i \frac{2^n}{n+1}\sin(\frac{2i+1}{2n+2})$$

Theorem 33 *Die Berechnung von l Funktionen an m Auswertungsstellen kostet $O(n^2 + lmn)$ Schritte. Außerdem ist der Algorithmus stabil.*

6.3 Interpolationsfehler

Definition 19 (Lebesque-Konstante) *Wir definieren die **Lebesgue-Konstante** auf dem Intervall $[-1, 1]$ als:*

$$\Lambda_n := \|\sum_{i=0}^{n}|L_i(.)|\|_\infty$$

Theorem 34 *Sei π_n das Interpolationspolynom für f und $\tilde{\pi_n}$ das für $\tilde{f}$ dann gilt:*

$$\|\pi_n - \tilde{\pi_n}\| \leq \Lambda_n \|f - \tilde{f}\|$$

Definition 20 (Stetigkeitsmodul) *Der **Stetigkeitsmodul** fon f ist definiert als:*

$$\omega(f, \delta) := \sup_{|x-y|<\delta}|f(x) - f(y)|$$

Theorem 35 *Für Lipschitz-stetige f mit Lipschitzkonstante L gilt: $\omega(f, \delta) \leq L\delta$.*

Theorem 36 (Interpolationsfehlerschranke durch Stetigkeitsmodul) *Es gilt folgende Abschätzung für den Fehler der besten Approximation:*

$$\inf_{q \in \mathbb{P}^n} \|f - q\| \leq (1 + \frac{\pi^2}{2})\omega(f, 1/n)$$

Theorem 37 (Interpolationsfehlerschranke durch Bestappoximation) *Sei p^* die beste polynomielle Approximation an f, dann gilt folgende Fehlerabschätzung für das Interpolationspolynom π_n:*

$$\|f - \pi_n| \leq (1 + \Lambda_n)\|f - p^*\|$$

Theorem 38 (Wachstumsschranken der Lebesgue-Konstanten) *Die Lebesgue-Konstanten sind durch folgende Schranken begrenzt.*

- *Für Chebychev-Punkte gilt:*

$$\frac{2}{\pi}\ln(n+1) \leq \Lambda_n \leq \frac{2}{\pi}\ln(n+1) + 1$$

- *Für äquidistante Punkte gilt:*

$$Ce^{n/2} \leq \Lambda_n \sim \frac{2^{n+1}}{en \log n}$$

Theorem 39 (De la Vallée Poussin) *Sei $f \in C$ und $p \in \mathbb{P}^n$ außerdem seien $x_0, ...x_{n+1}$ sodass: $(f(x_i) - p(x_i))(f(x_{i+1}) - p(x_{i+1})) < 0$. Dann gilt für die Bestapproximation p^*:*

$$\|p^* - f\| \geq \min |f(x_i) - p(x_i)|$$

Theorem 40 (Oszillationstheorem) *Sei $f \in C$ und $p \in \mathbb{P}^n$ zusätzlich seien $x_0, ..., x_{n+1}$, sodass:*

- $f(x_i) - p(x_i) = -(f(x_{i+1}) - p(x_{i+1}))$
- $|f(x_0) - p(x_0)| = \|f - p\|$

*Dann ist p die Bestapproximation von f in $\mathbb{P}^n$. Man nennt p auch **MinMax-Polynom**.*

6.4 Spline-Interpolation

Definition 21 (Spline) *Sei $\Delta = \{a = x_0 < x_1 < ... < x_n = b\}$ ein Gitter auf $[a, b]$ Dann sind die Splineräume von Grad k definiert als:*

$$S_{\Delta,k} := \{v \in C^{k-1}([a, b]) : v|_{[x_i, x_{i+1}]} \in \mathbb{P}^k\}$$

$$S_{\Delta,0} := \{v|_{[x_i, x_{i+1}]} \in \mathbb{P}^0\}$$

Die Spline-Räume sind Vektorräume mit Dimension $n + k$.

Definition 22 (Unterarten des kubischen Splines) *Um einen kubischen Spline $s \in S_{\Delta,3}$ zu $f \in C^2$ zu berechnen benötigt man weitere Nebenbedingugnen: Dafür gibt es mehrere Möglichkeiten:*

- *Vollständiger Spline: $s'(x_0) = f'(x_0)$ und $s'(x_n) = f'(x_n)$*
- *Natürlicher Spline: $s''(x_0) = s''(x_n) = 0$*
- *Periodischer Spline: $s'(x_0) = s'(x_n)$ und $s''(x_0) = s''(x_n)$*

Definition 23 (Momentendarstellung eines kubischen Splines) *Für $s \in S_{\Delta,3}$ gilt $s'' \in S_{\Delta,1}$. Seien die N_i^2 Basisfunktionen von $S_{\Delta,1}$. Dann kann s'' dargestellt werden als:*

$$s''(x) = \sum_{i=0}^{n} m_i N_i^2(x)$$

*Die m_i heißen **Momente**.*

Definition 24 *Aus den Momenten kann s berechnet werden als:*

$$s(x)|_{[x_{i-1},x_i]} = c_i + d_i(x - \frac{x_{i-1}+x_i}{2}) + m_i\frac{(x-x_i)^3}{6h_i} - m_{i-1}\frac{(x-x_{i-1})^3}{6h_i}$$

Dabei sei $h_i = x_i - x_{i-1}$ und

$$c_i = \frac{f_i + f_{i-1}}{2} - \frac{h_i^2}{12}(m_i + m_{i-1})$$

$$d_i = \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(m_i - m_{i-1})$$

Die Momente lassen sich folgendermaßen ausrechnen:

Theorem 41 *Es gilt:*

$$\mu_i m_{i-1} + 2m_i + (1-\mu_i)m_{i+1} = b_i$$

mit $\mu_i := h_i/(h_i + h_{i+1})$ und:

$$b_i := \frac{6}{h_i + h_{i+1}}(\frac{f_{i+1}}{h_i} - f_i(1/h_i + 1/h_{i+1}) + f_{i-1}/h_i)$$

Es gilt:

1. *Vollständiger kubischer Spline:*

$$\begin{bmatrix} 2 & 1 & & & & \\ \mu_1 & 2 & 1-\mu_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \mu_{n-1} & 2 & 1-\mu_{n-1} & \\ & & & 1 & 2 & \end{bmatrix} \begin{pmatrix} m_0 \\ \vdots \\ \vdots \\ \vdots \\ m_n \end{pmatrix} = \begin{pmatrix} b_0 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

2. *Natürlicher kubischer Spline: Hier ist $m_0 = m_n 0$ und:*

$$\begin{bmatrix} 2 & 1-\mu_1 & & & \\ \mu_2 & 2 & 1-\mu_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & 1-\mu_{n-2} \\ & & & \mu_{n-1} & 2 \end{bmatrix} \begin{pmatrix} m_1 \\ \vdots \\ \vdots \\ \vdots \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ \vdots \\ b_{n-1} \end{pmatrix}$$

3. *periodischer Spline: Hier ist $m_n = m_0$*

$$\begin{bmatrix} 2 & 1-\mu_n & & & \mu_n \\ \mu_1 & 2 & 1-\mu_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & 1-\mu_{n-2} \\ 1-\mu_{n-1} & & & \mu_{n-1} & 2 \end{bmatrix} \begin{pmatrix} m_0 \\ \vdots \\ \vdots \\ \vdots \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ \vdots \\ \vdots \\ \vdots \\ b_{n-1} \end{pmatrix}$$

Theorem 42 (Fehler der Spline-Interpolation) *Für $f \in C^4$ und den vollständigen kubischen Spline $s \in S_{\Delta,3}$ gilt:*

$$\|(f-s)^{(k)}\| \leq Ch^{4-k}\|f^{(4)}\|$$

Dabei ist $h := \max |x_{i+1} - x_i|$.

Definition 25 (Kardinal-Spline-Basisfunktionen) *$S_{\Delta,3}$ kann durch folgende Basis dargestellt werden:*

- $K'_{-1}(x_0) = 1, K_{-1}(x_i) = 0, K'_{-1}(x_n) = 0$
- $K_i(x_j) = \delta_{ij}, K'_i(x_0) = K'_i(x_n) = 0$
-
- $K'_{n+1}(x_0) = 1, K_{-1}(x_i) = 0, K'_{-1}(x_n) = 0$

Da diese globale Träger haben wird alternativ eine andere Basisdarstellung gewählt:

Definition 26 (B-Splines) *Die **B-Spline Basisfunktionen** von $S_{\Delta,l}$ sind definiert durch die $B_{i,l+1}, i = 1-l, n$:*

$$B_{i,1}(x) := \begin{cases} 1 & \textit{falls } x \in [x_{i-1}, x_i], \\ 0 & \textit{sonst} \end{cases}$$

$$B_{i,k} := \frac{x - x_{i-1}}{x_{i+k-2} - x_{i-1}} B_{i,k-1} + \frac{x_{i+k-1} - x}{x_{i+k-1} - x_i} B_{i+1,k-1}$$

Theorem 43 (Nützliche EIgenschaften der B-Spline Basisifunktionen) *Die B-Spline Basisfunktionen erfüllen folgende Eigenschaften:*

1. *supp$B_{i,k} \subset [s_{i-1}, x_{i+k-1}]$*
2. *$B_{i,k} \geq 0$*
3. *$\sum_{i=2-k}^{n} B_{i,k} = 1$*

7 Quadratur

Definition 27 (Quadraturformel) *Eine **Quadraturformel** hat im allgemeinen die Form:*

$$Q_b^a(f) := \sum_{i=0}^{n} w_i f(x_i)$$

*Sind die Gewichte $w_i := \int_a^b L_i(x)dx$, so ist die Quadraturformel vom **Interpolationstyp**.*

Definition 28 (Exaktheitsgrad) *Eine Quadraturformel hat den **Exaktheitsgrad** n falls sie für alle Polynome in $\mathbb{P}^n$ exakt ist, aber nicht für x^{n+1}*

Definition 29 (symmetrische Quadraturformel) *Eine **symmetrische QF** erfüllt:*

$$w_i = w_{n-i}, x_i - a = b - x_{n-i}$$

Theorem 44 (Aussagen über den Exaktheitsgrad)

1. *Jede QF vom Interpolationstyp und mit $n+1$ Knoten hat mind. Exaktheitsgrad n.*
2. *Außerdem ist der Exaktheitsgrad einer symmetrischen Quadraturformel ungerade.*
3. *Jede QF mit $n+1$ Knoten und mindestens Exaktheitsgrad n muss vom Interpolationstyp sein.*
4. *Falls die Gewichte einer QF durch Interpolation bestimmt sind und die Knoten symmetrisch angeordnet sind. Dann ist die QF auch symmetrisch*
5. *Eine QF mit $n+1$ Knoten hat maximal den Exaktheitsgrad $2n+1$.*
6. *Die QF mit Exaktheitsgrad $2n+1$ bei $n+1$ Knoten ist eindeutig.*
7. *Für eine QF mit Exaktheitsgrad $\geq 2n$ müssen alle Gewichte positiv sein.*

7.1 Newton-Cotes-Formel

Definition 30 (Newton-Cotes QF) *Bei der Newton Cotes QF sind die Punkte äquidistant verteilt, man unterscheidet zwischen:*

1. *Der geschlossenen Newton Cotes Formel:*

$$x_i = a + ih, h = \frac{b-a}{n}$$

2. *Der offenen Newton Cotes Formel:*

$$x_i = a + (i+1)h, h = \frac{b-a}{n+2}$$

Geschlossene Quadraturformeln:

Name	Gewichte	Fehler	Exaktheitsgrad
Trapez	$w_1 = w_2 = \frac{1}{2}$	$-\frac{h^3}{12}f''(\xi)$	1
Cavalieri-Simpson	$w_0 = \frac{1}{6}(b-a), w_1 = \frac{4}{6}(b-a)$	$-\frac{h^5}{90}f^{(4)}(\xi)$	3
Simpson 3/8	$w_0 = \frac{3}{8}\frac{b-a}{3}, w_1 = \frac{9}{8}\frac{b-a}{3}$	$-\frac{3h^5}{80}f^{(4)}(\xi)$	3
Milne	$w_1 = \frac{7}{90}, w_2 = \frac{32}{90}, w_3 = \frac{12}{90}$	$-\frac{8}{945}h^7 f^{(6)}(\xi)$	5

Theorem 45 (Fehler allgemeiner Newton-Cotes-Formeln) *Sei n gerade und $f \in C^{n+2}$, dann gilt für den Quadraturfehler:*

$$E_n(f) = \frac{M_n}{(n+2)!}h^{n+3}f^{(n+2)}(\xi)$$

Dabei ist:

$$M_n = \begin{cases} \int_0^n t\hat{\omega}_{n+1}(t)dt & \text{für gechlossene QF} \\ \int_{-1}^{n+1} t\hat{\omega}_{n+1}(t)dt & \text{für offene QF} \end{cases}$$

Wobei $\hat{\omega}_{n+1}(t) := \prod_{i=0}^{n}(t-i)$. Für n ungerade gilt:

$$E_n(f) = \frac{K_n}{(n+1)!}h^{n+2}f^{(n+1)}(\xi)$$

Dabei ist:

$$K_n = \begin{cases} \int_0^n \hat{\omega}_{n+1}(t)dt & \text{für gechlossene QF} \\ \int_{-1}^{n+1} \hat{\omega}_{n+1}(t)dt & \text{für offene QF} \end{cases}$$

Die Newton-Cotes Formeln können auf Teilintervalle angewandt werden.

Theorem 46 *Für zusammengestzte Newton-Cotes Formeln mit m Teilintervallen verliert die Fehlerabschätzung eine h-Potenz es gilt mit $H = \frac{b-a}{m}$.*
Für n gerade:

$$E_n(f) = \frac{(b-a)M_n}{\gamma_n^{n+3}(n+2)!}H^{n+2}f^{(n+2)}(\xi)$$

Für n ungerade:

$$E_n(f) = \frac{(b-a)K_n}{\gamma_n^{n+2}(n+1)!}H^{n+1}f^{(n+1)}(\xi)$$

Wobei $\gamma_n = \begin{cases} n+2 & \text{für QF offen} \\ n & \text{für QF geschlossen} \end{cases}$.

7.2 Gauß-Quadratur

Theorem 47 (Dreiterm-Rekursion) *Die orthogonalen Polynome für ein Skalarprodukt $\langle .,. \rangle$ werden bestimmt durch:*

$$p_{k+1} = (x - \alpha_k)p_k(x) - \beta_k p_{k-1}(x)$$

$$p_{-1} = 0, p_0(x) = 1$$

und:

$$\alpha_k = \frac{\langle xp_k, p_k \rangle}{\langle p_k, p_k \rangle}, \beta_k = \frac{\langle p_k, p_k \rangle}{\langle p_{k-1}, p_{k-1} \rangle}$$

Definition 31 (Legendre Polynome) *Die Legendre Polynome sind orthogonal zu dem Skalarprodukt mit Gewichtugnsfunktion $w = 1$.*

$$L_0(x) = 1, L_1(x) = x$$

$$L_{k+1}(x) = \frac{2k+1}{k+1} x L_k(x) - \frac{k}{k+1} L_{k+1}(x)$$

Sie lassen sich auch folgendermaßen darstellen:

$$L_k(x) = \frac{(-1)^k}{2^k k!} \frac{d^k}{dx^k}(1-x^2)^k$$

Definition 32 (Gegenbauer-Polynome) *Die Gegenbauer-Polynome G_k^α sind die Orthogonalpolynome für $w(x) = (1-x^2)^{\alpha-1/2}$.*
Die Legendre und Chebchev-Polynome sind Spezialfälle für $\alpha = 1/2$ und $\alpha = 0$.
Außerdem gilt:

$$L_n' = G_{n-1}^{3/2}(x)$$

Theorem 48 *Die Dreiterm-Rekursion für die $G_k^{3/2}$ lautet:*

$$G_0^{3/2}(x) = 1, G_1^{3/2}(x) = 3x$$

$$G_{k+1}^{3/2}(x) = \frac{2k+3}{k+1} x G_k^{3/2}(x) - \frac{k+2}{k+1} G_{k+1}^{3/2}(x)$$

Definition 33 (Laguerre-Polynome) *Laguerre-Polynome L_k sind orthogonal zu $w(x) = e^{-x}$.*

Theorem 49 *Die Dreiterm-Rekursion für die Laguerre Polynome lautet:*

$$L_{-1}(x) = 0, L_1(x) = 1$$

$$L_{k+1}(x) = (2k+1-x)L_k(x) - k^2 L_{k-1}(x)$$

Sie lassen sich auch folgendermaßen darstellen:

$$L_k(x) = e^x \frac{d^k}{dx^k}(e^{-x}x^k)$$

Definition 34 (Hermite-Polynome) *Die Hermite-Polynome sind orthogonal zur Gewichtsfunktion $w(x) = e^{-x^2}$.*

Theorem 50 *Die Hermite-Polynome folgen der Dreiterm-Rekursion:*

$$H_{-1}(0), H_0(x) = 1$$

$$H_{k+1}(x) = 2xH_k(x) - 2kH_{k-1}(x)$$

Theorem 51 (Quadratur-Satz von Jacobi) *Eine QF mit $n+1$ Knoten ist had Exaktheitsgrad $n+m$, genau dann wenn das Knotenpolynom ω_{n+1} orthogonal auf $\mathbb{P}^{m-1}$ steht.*

Definition 35 (Gauß-(Lobatto)-Quadratur) *Bei der Gaußquadratur sind die Stützstellen die Nullstellen von L_{n+1} und die Gewichte:*

$$w_i = \frac{2}{(1-x_i^2)(L'_{n+1}(x_i))^2}$$

Für die Gauß-Lobatto Quadratur setzt man $x_0 = -1, x_n = 1$ an die Endpunkte des Intervalls. Die anderen Punkte sind die Nullstellen des $G_{n-1}^{3/2} = L'_n(x)$-Polynoms. Die Gewichte sind:

$$w_i = \frac{2}{n(n+1)} \frac{1}{[L_n(x)]^2}$$

Definition 36 (Laguerre-Quadratur) *Um auf dem Intervall $(0, \infty)$ zu integrieren benutzt man die Laguerre-Quadratur. Dazu definiert man $\phi(x) = e^x f(x)$ und es gilt $\int_0^\infty e^{-x}\phi(x) = I(f)$ Man quadriere nun ϕ mit den Stützstellen als den Nullstellen des Laguerre-Polynoms: L_{n+1} und den Gewichten:*

$$w_i = \frac{x_i}{(n+2)^2[L_{n+2}(x_i)]^2}$$

Definition 37 (Hermite-Quadratur) *Um auf dem Intervall $(-\infty, \infty)$ zu integrieren benutzt man die Laguerre-Quadratur. Dazu definiert man $\phi(x) = e^{-x^2} f(x)$ und es gilt $\int_0^\infty e^{-x^2}\phi(x) = I(f)$ Man quadriere nun ϕ mit den Stützstellen als den Nullstellen des Hermite-Polynoms: H_{n+1} und den Gewichten:*

$$w_i = \frac{2^{n+2}(n+1)!\sqrt(\pi)}{[H_{n+2}(x_i)]^2}$$

7.3 Extrapolation

Definition 38 (Richardson-Extrapolation) *Ein numerisches Schema A mit einer Entwicklung*

$$A(h) = q_0 + q_1 h + ... + q_k h^k + O(h^{k+1})$$

lässt sich durch extrapolation verbessern. Das Extrapolationsschema ist:

$$A_{k,0} = A(\delta^k h)$$

$$A_{k,l+1} = \frac{A_{k,l} - \delta^{l+1} A_{k-1,l}}{1 - \delta^{l+1}}$$

Theorem 52 *Durch das Anwenden des Extrapolationsschemas mit $\delta \in (0,1)$ erhält man:*

$$A_{k,l}(h) = q_0 + O((\delta^k h)^{l+1})$$

Theorem 53 (Euler-MacLaurin-Formel) *Sei $f \in C^{2n+2}$ und $q_0 = I(f)$ dann gilt für die zusammengestzte Trapezregel:*

$$I_{1,m}(f) = q_0 + \sum_{i=1}^{n} \alpha_{2i} h_m^{2i} + O(h_m^{2n+2})$$

Algorithm 19 Romberg-Quadratur

Require: f
  for $k = 1, ..., n$ **do**
    $A_{k,0} \leftarrow I_{1,2^k}(f)$
  end for
  for $l = 1, ..., n$ **do**
    for $k = l, ..., n$ **do**
      $A_{k,l} \leftarrow \frac{4^{l+1} A_{k,l-1} - A_{k-1,l-1}}{4^{l+1}-1}$
    end for
  end for

8 Lösung nichtlinearer Gleichungssysteme

Algorithm 20 Bisektionsverfahren

Require: f, x_0, x_1 und $f(x_0)f(x_1) < 0$
  for $k = 1, 2, ...$ **do**
    $x_{k+1} \leftarrow \frac{1}{2}(x_k + x_{k-1})$
    if $f(x_{k+1})f(x_{k-1}) < 0$ **then**
      $x_k \leftarrow x_{k-1}$
    end if
  end for

Theorem 54 (Konvergenz des Bisektionsverfahren) *Das Bisektionsverfahren konvergiert gegen eine Nullstelle x^*:*

$$|x^* - x_{k+1}| \leq 2^{-k}|x_0 - x_1|$$

Algorithm 21 Sekantenverfahren

Require: f, x_0, x_1
  for $k = 1, 2, ...$ **do**
    $x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$
  end for

Das Sekantenverfahren konvergiert, falls die Startwerte bereits nahe an einer Nullstelle x^* liegen recht schnell, allerdings können Fehler auftreten, falls $f(x_{k-1}) = f(x_k)$.

Algorithm 22 Regula Falsi

Require: f, x_0, x_1 und $f(x_0)f(x_1) < 0$
  for $k = 1, 2, ...$ **do**
    $x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k)$
    if $f(x_{k+1})f(x_{k-1}) < 0$ **then**
      $x_k \leftarrow x_{k-1}$
    end if
  end for

Theorem 55 *Das Regula Falsi Verfahren konvergiert immer.*

Definition 39 (Konvergenzordnung) *Eine Folge $(x_k)_{k\in\mathbb{N}}$ heißt konvergent von Ordnung $p \geq 1$ gegen x^* falls $c < \infty$ exisitiert, sodass:*

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

Für $p = 1$ muss zusätzlich $c < 1$ gelten.

Theorem 56 (Konvergenz einer Fixpunktiteration) *Sei $\Phi \in C^1$ und x^* ein Fixpunkt. Die Fixpunktiteration von Φ in der Umgebung von x^* ist:*

- *konvergent falls $|\Phi'(x^*)| < 1$*
- *divergent falls $|\Phi'(x^*)| > 1$*
- *konvergent von ordnung $p \geq 2$ falls $\Phi \in C^p$ und:*

$$\Phi^{(l)}(x^*) = 0, \quad l = 1, 2, ..., p-1$$

Algorithm 23 Newton-Verfahren

Require: $F \in C^1$, Startwert x_0.
  for $k = 0, 1, ...$ **do**
    Löse $DF(x_k)\Delta x_k = F(x_k)$
    $x_{k+1} \leftarrow x_k - \Delta x_k$
  end for

Theorem 57 *Falls für DF ein ω existiert, sodass:*

$$\|DF(x)^{-1}(DF(x + sv) - DF(x))v\| \leq s\omega\|v\|^2$$

Dann gilt für alle Startwerte $x_0 \in B_{\frac{2}{\omega}(x^*)}$, dass das Newton-Verfahren mindestens quadratisch gegen x^* konvergiert.

Algorithm 24 gedämpftes Newton-Verfahren

Require: $F \in C^1$, Startwert x_0, Dämpfungsparameter γ.
  for $k = 0, 1, ...$ **do**
    Löse $DF(x_k)\Delta x_k = F(x_k)$
    $x_{k+1} \leftarrow x_k - \gamma\Delta x_k$
  end for

Der Dämpfungsparameter kann natürlich in jedem Schritt angepasst werden.

Theorem 58 *Das gedämpfte Newtonverfahren für Φ mit Parameter γ ist äquivalent zum Newtonverfahren für $\Phi_\gamma(x) = (1-\gamma)x + \gamma\Phi(x)$*

Definition 40 (Inexaktes Newtonverfahren) *Hier wird $DF(x_k)\Delta x_k = F(x_k)$ in jedem Schritt nur näherungsweise gelöst.*

Definition 41 (Newton-artiges Verfahren) *Hier wird $DF(x_k)$ durch M_k approximiert.*

Theorem 59 (Dennis-Moré-Bedingugn) *Für ein Newton-artiges Verfahren ist folgendes equivalent:*

- *Die Konvergenz ist superlinear und $F(x^*) = 0$*
- $\|(M_k - DF(x_k)\Delta x_k)\| = o(\|\Delta x_k\|)$

Theorem 60 (Sherman-Morrison-Formel) *Sei A invertierber, dann ist $A+uv^T$ genau dann invertierbar, wenn $1 + v^T Au \neq 0$. Dann gilt:*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}$$

Algorithm 25 BFGS-Algorithmus

Require: Startwert x_0 und Startinverse B_0^{-1} (Approximation von $DF^{-1}(x_0)$, oder häufig auch identität), Schrittweiten λ_k
  for $k = 0, 1, 2, ...$ **do**
    $\Delta x \leftarrow -\lambda_k B_k^{-1} F(x_k)$
    $x_{k+1} \leftarrow x_k + \Delta x$
    $\Delta F \leftarrow F(x_{k+1}) - F(x_k)$
    $B_{k+1}^{-1} \leftarrow B_{k+1}^{-1} + (\Delta x - B_k^{-1}\Delta F)\frac{\Delta x^T B_k^{-1}}{\Delta x^T B_k^{-1} \Delta F}$
  end for
