

# Numerik MA0008: Zusammenfassung

Jonas Treplin

February 15, 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  $S_i := K(a_{ii}, \sum_{j=1, j \neq i}^n |a_{ij}|)$ . Wobei mindestens ein Eigenwert jeder Zusammenhangskomponente zugeordnet ist.

## 2 Matrixfaktorisierung

**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 Rückwärtssubstitution. Diese sind aus dem Endschrift des Lösen von Gleichungssystemen mit dem Gauß-Algorithmus bekannt. Glücklicherweise kann jede invertierbare Matrix (fast eindeutig)

---

**Algorithm 1** Vorwärtssubstitution (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** Rückwärtssubstitution (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**

---

in solche Matrizen zerlegt werden. Dies geschieht Wahlweise durch eine LU-Zerlegung oder eine QR-Zerlegung. Mit Pivots erreicht man, dass jede invertierbare Matrix  $A \in GL(n)$  zerlegt werden kann, sodass  $PA = LU$ . Dazu wählt man in jedem Schritt  $i$  die Zeile  $j = \arg \max_{j \geq i} |a_{ji}^i|$  und vertauscht diese mit der  $i$ -ten Zeile.

---

**Algorithm 3** LU-Zerlegung ohne Pivots

---

**Require:**  $(a_{ij}) = A \in GL(n)$

```
for  $i \in 1 : n$  do
   $l_{ii} \leftarrow 1$ 
  for  $j \in i + 1 : n$  do
     $l_{ji} \leftarrow -\frac{a_{ji}}{a_{ii}}$ 
    for  $k \in i : n$  do
       $u_{jk} \leftarrow u_{jk} - a_{ji}a_{ki}$ 
    end for
  end for
end for
end for
```

---

**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 (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 4** Berechnung der Cholesky Zerlegung

---

**Require:**  $A$  s.p.d.

```
 $L, U \leftarrow \text{LU\_Zerlegung}(A)$   
 $D = (u_{ii})$  Diagonalmatrix.  
 $\tilde{L} = \sqrt{D}L$ .
```

---

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

---

**Algorithm 5** QR-Zerlegung mit Givens-Rotationen

---

**Require:**  $A \in \mathbb{R}^{n \times m}$

$Q \leftarrow I_n$

**for**  $i \in [n]$  **do**

**for**  $j \in (n, \dots, i+1)$  **do**

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

$Q \leftarrow GQ$

$A \leftarrow GA$

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

---

**Algorithm 6** QR-Zerlegung mit Householder Rotationen

---

**Require:**

**Require:**  $A \in \mathbb{R}^{n \times m}$

$Q \leftarrow I_n$

**for**  $i \in [n]$  **do**

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

$Q \leftarrow HQ$

$A \leftarrow HA$

**end for**

$Q \leftarrow Q^*$

---

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

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

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

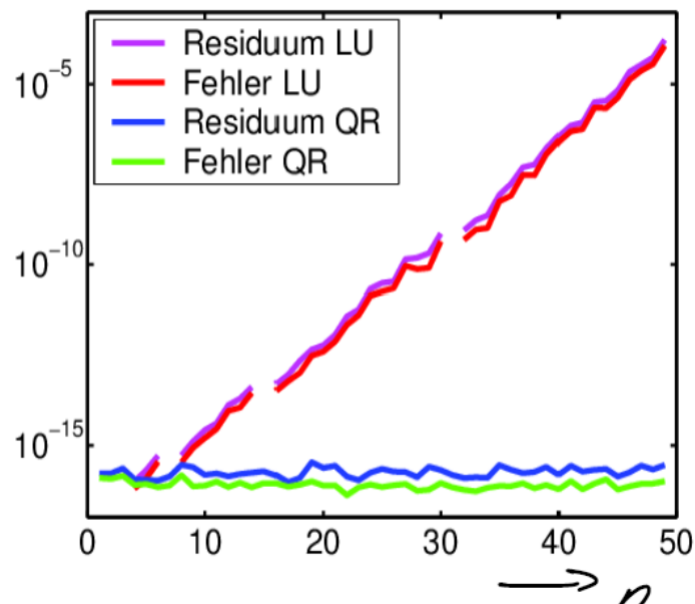


Figure 1: Fehler beim Lösen von  $W_n x = b_n$

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  $T = (A_1, A_2, \dots, A_n)$  mit Störung  $\tilde{T} = (A_1 + E_1, \dots, A_n + E_n)$ :

- das absolute Fehlermaß:

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

- das relative Fehlermaß:

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

**Definition 5 (Maschinenepsilon)** Das **Maschinen- $\epsilon$**  ist 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:

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

Man unterscheidet zwischen:

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

**Theorem 6** Die Kondition einer linearen Gleichung  $Ax = b$  hängt nur von  $A$  ab und ist:

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

## 4 Ausgleichsrechnung

**Definition 7 (Lineares Ausgleichsproblem)** Sei  $A \in \mathbb{R}^{n \times m}$ ,  $b \in \mathbb{R}^n$  mit  $m \leq n$ . Gesucht ist  $x \in \mathbb{R}^m$ . Sodass

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

minimiert wird. Es wird im Allgemeinen angenommen, dass  $\text{Rang}(A) = m$ .

**Definition 8 (Normalengleichung)** Gegeben ein Ausgleichsproblem  $A, b$ , nennt man:

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

die Normalengleichung.

**Theorem 7 (Lösung des Linearen Ausgleichproblems mittels Normalengleichung)**

Die Lösung der Normalengleichung ist das eindeutige gesuchte Minimum des Ausgleichsproblems.

---

**Algorithm 7** Lösen des Ausgleichproblems mittels Normalengleichung

---

**Require:** Ausgleichsproblem  $A \in \mathbb{R}^{n \times m}$ ,  $b \in \mathbb{R}^n$

$L \leftarrow \text{Cholesky}(A^T A)$

Löse  $LL^T x = A^T b$  durch Vorwärts und Rückwärtssubstitution.

---

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.

**Theorem 8 (Aufwand der Lösungsmethoden)** Der Aufwand beträgt:

1. Normalengleichung:  $nm^2 + \frac{m^3}{3}$ .

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

---

**Algorithm 8** Lösen des Ausgleichproblems mittels QR-Methode

---

**Require:** Ausgleichsproblem  $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n$

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  $(\cdot)_1$  die ersten  $m$  Elemente seien.

---

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

**Theorem 9 (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

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 9** Vektoriteration

---

**Require:**  $A \in \mathbb{R}^{n \times n}$  und Startvektor  $x^{(0)} \in \mathbb{R}^n$ .

**for**  $i = 1, 2, 3, \dots$  **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 10 (Konvergenz der Vektoriteration)** Sei  $A \in \mathbb{R}^{n \times n}$  mit EW  $|\lambda_n| \leq \dots \leq |\lambda_2| < |\lambda_1|$  und EV  $(v_i)_{i \in [n]}$ . Dann gilt:

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

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

Wobei  $\beta_1 := v_1^T x^{(0)}$  und  $C_1, C_2$  Konstanten sind die vom Startwert abhängen

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 Eigenwert von  $(A - \mu I)^{-1}$  der Kerwehrt des nächsten Eigenwerts an  $\mu$  ist.

---

**Algorithm 10** 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  $i = 1, 2, \dots$  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  $\mu$  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 Rückwärtssubstitution.

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

---

**Algorithm 11** 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  $i = 1, 2, 3, \dots$  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 11 (Konvergenz der QR-Iteration)** Sei  $A \in \mathbb{R}^n$  symmetrisch und mit EW  $|\lambda_1| > |\lambda_2| > \dots > |\lambda_m| > 0$ . Sei  $A = V\Lambda V^T$  diagonalisiert mit  $V$  orthogonal. Besitzt  $(Q_0^T V)^{-1}$  eine normierte LU-Zerlegung, dann gilt:

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

Der Beweis kann auf schwächere Bedingungen ausgeweitet 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 12** Berechnung von  $Q_0$ 

---

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

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

 $A \leftarrow GA$  $Q_0 \leftarrow GQ_0$ **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 13** 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**  $i = 1, 2, 3, \dots$  **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. **Wilkinson-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:** Wähle 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  $|(A_i)_{n,n-1}| \leq \text{TOL}$  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 12 (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 13 (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

$$A_k = Q_k^T \dots Q_1^T Q_0^T A Q_0 Q_1 \dots Q_k$$

Wir definieren  $\hat{Q}_k := Q_0 Q_1 \dots Q_k$ . 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  $A$ .