

# Numerisches Programmieren (IN0019)

Frank R. Schmidt

Winter Semester 2016/2017

9. Symmetrisches Eigenwertproblem . . . . .	2
<b>Eigenwert-Problem</b> . . . . .	<b>3</b>
Verallgemeinerte Fourier-Reihe . . . . .	4
Ableitungen als Lineare Abbildungen . . . . .	5
Diskretisierung periodischer Funktionen . . . . .	6
Diskretisierung der 2. Ableitung . . . . .	7
Eigenfunktionen (Bsp.) . . . . .	8
Eigenwerte (Wdh.) . . . . .	9
Nullstellen von Polynomen . . . . .	10
Lösen des Eigenwertproblems . . . . .	11
<b>Kondition</b> . . . . .	<b>12</b>
Beschreibung des Eigenwertproblems . . . . .	13
Partielle Ableitung des EW-Problems . . . . .	14
Absolute Kondition des EW-Problems. . . . .	15
Symmetrisches Eigenwert-Problem . . . . .	16
<b>Vektor-Iteration</b> . . . . .	<b>17</b>
Vektor-Iteration . . . . .	18

Konvergenz (Vektor-Iteration) . . . . .	19
Vektor-Iteration (Bsp.) . . . . .	20
Berechnung aller Eigenvektoren . . . . .	21
Zusammenfassung (Vektor-Iteration) . . . . .	22
Inverse Vektor-Iteration . . . . .	23
Inverse Vektor-Iteration (Bsp.) . . . . .	24
<b>QR-Verfahren</b> . . . . .	<b>25</b>
Berechnung aller Eigenwerte . . . . .	26
QR-Verfahren . . . . .	27
Eigenschaften des QR-Verfahrens . . . . .	28
Zusammenfassung . . . . .	29
<b>Gerschgorin-Kreise</b> . . . . .	<b>30</b>
Approximation von Eigenwerten . . . . .	31
Eigenwerte und Diagonaleinträge . . . . .	32
Gerschgorin-Kreise . . . . .	33
Gerschgorin-Kreise . . . . .	34
Gerschgorin-Kreise . . . . .	35
"Frequenzen" im $\mathbb{R}^2$ . . . . .	36
"Frequenzen" einer Katze . . . . .	37

**Verallgemeinerte Fourier-Reihe**

Das Berechnen von Eigenwerten wird bei viele praktische Anwendungen vorausgesetzt, z.B. bei der Verallgemeinerung der Fourier-Analyse.

Wir haben bereits die Vorteile der Fourier-Reihe gesehen. Ein wesentlicher Bestandteil ist die Orthogonalbasis

$$c_n(x) := \cos(n \cdot x)$$

$$s_n(x) := \sin(n \cdot x)$$

Diese Funktionen sind aber nur auf  $\mathbb{R}$  definiert. Um die Fourier-Reihe zu verallgemeinern, sollte man diese Funktionen eindeutiger charakterisieren.

Interessanterweise erfüllen die Funktionen folgende Gleichungen

$$c_n'' = -n^2 c_n$$

$$s_n'' = -n^2 s_n,$$

## Ableitungen als Lineare Abbildungen

Wir hatten bereits gesehen, dass man die Funktionen  $c_n$  und  $s_n$  als Elemente des Vektorraums  $C^0([-\pi; \pi])$  ansehen kann.

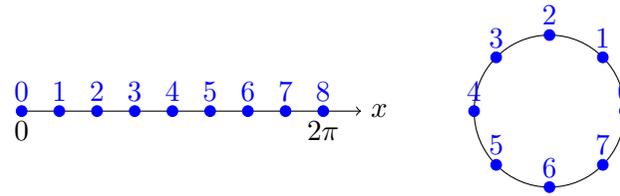
Weiter haben wir gesehen, dass man das Integral als lineare Abbildung  $\int: C^0([-\pi; \pi]) \rightarrow \mathbb{R}$  interpretieren kann.

Man kann aber auch Ableitungen als lineare Abbildungen verstehen

$$\begin{aligned} \frac{\partial^2}{\partial x^2}: C^\infty([-\pi; \pi]) &\rightarrow C^\infty([-\pi; \pi]) \\ f &\mapsto f'' \end{aligned}$$

Damit kann man  $c_n$  und  $s_n$  als **Eigenfunktionen** zu den **Eigenwerten**  $-n^2$  von der **linearen Abbildung**  $\frac{\partial^2}{\partial x^2}$  interpretieren.

## Diskretisierung periodischer Funktionen



Bei der Fourier-Reihe betrachtet man **periodische Funktionen**  $f: \mathbb{R} \rightarrow \mathbb{R}$

$$f(x + 2\pi) = f(x)$$

Solche Funktionen kann man sich auch als Funktionen  $f: \mathbb{S}^1 \rightarrow \mathbb{R}$  vorstellen, wobei  $\mathbb{S}^1$  den Kreisrand beschreibt:

$$\mathbb{S}^1 = \{x \in \mathbb{R}^2 \mid \|x\| = 1\}$$

Diskretisiert man  $f$  äquidistant, erhält man  $v = (v_0, \dots, v_{n-1}) \in \mathbb{R}^n$ .

## Diskretisierung der 2. Ableitung

Da  $f$  äquidistant diskretisiert ist, ist der Abstand zwischen zwei Stellen

$$h = \frac{2\pi}{n}$$

Die erste Ableitung diskretisieren wir wie folgt ( $x_i = i \cdot h$ )

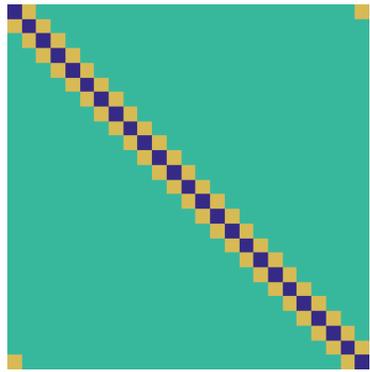
$$f' \left( x_i + \frac{h}{2} \right) \approx \frac{f(x_i + h) - f(x_i)}{h} = \frac{v_{i+1} - v_i}{h}$$

Für die zweite Ableitung gilt nun

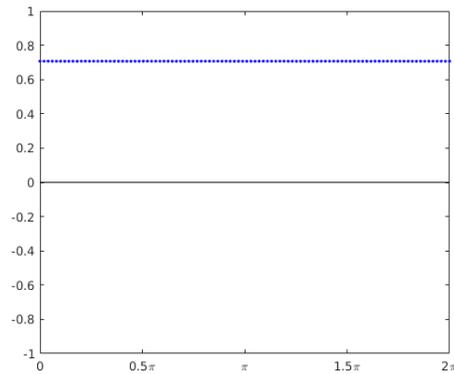
$$f''(x_i) \approx \frac{f \left( x_i + \frac{h}{2} \right) - f \left( x_i - \frac{h}{2} \right)}{h} = \frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} =: D_2 \cdot v$$

wobei  $i - 1$  und  $i + 1$  modulo  $n$  berechnet wird.

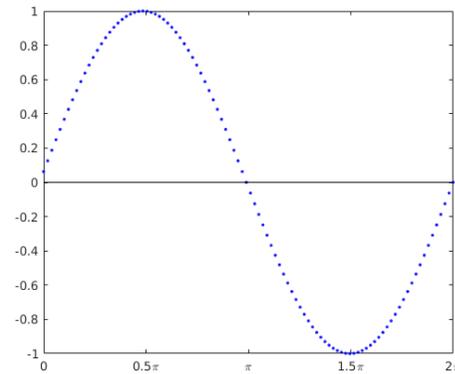
## Eigenfunktionen (Bsp.)



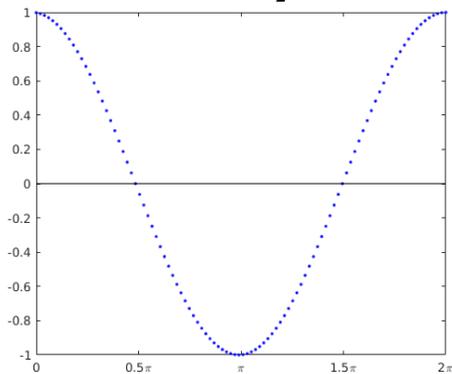
Matrix  $D_2$



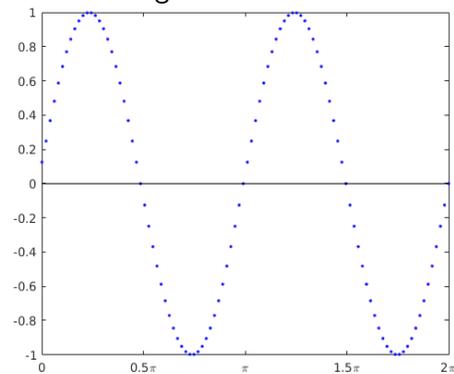
Eigenwert  $\lambda = 0$



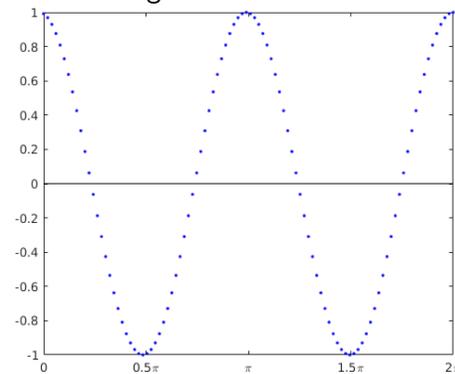
Eigenwert  $\lambda = -1$



Eigenwert  $\lambda = -1$



Eigenwert  $\lambda = -4$



Eigenwert  $\lambda = -4$

## Eigenwerte (Wdh.)

Ein Vektor  $x \in \mathbb{R}^n$  heißt **Eigenvektor** zum **Eigenwert**  $\lambda \in \mathbb{R}$  einer Matrix  $A \in \mathbb{R}^{n \times n}$ , wenn gilt:

$$A \cdot x = \lambda \cdot x \quad (x \neq 0),$$

d.h. die Richtung  $x$  beschreibt die **Fixgerade** der Abbildung  $x \mapsto A \cdot x$ .

Daher ist  $x \neq 0$  eine Lösung des Gleichungssystems

$$(A - \lambda I) \cdot x = 0.$$

Somit ist  $A - \lambda I$  eine singuläre Matrix und es gilt

$$p_A(\lambda) = 0 \quad p_A(t) := \det(A - t \cdot I)$$

Das Polynom  $p_A$  nennt man das **charakteristische Polynom bzgl.  $A$** .

## Nullstellen von Polynomen

Das Eigenwertproblem ist also äquivalent zum Berechnen von Nullstellen des charakteristischen Polynoms.

Abel bewies 1824, dass die Nullstellen eines Polynoms  $p$  vom Grad  $\deg(p) > 4$  im Allgemeinen nicht durch Elementaroperationen (Addition, Multiplikation, Wurzelziehen) darstellbar sind. Daher lösen wir das Eigenwertproblem **iterativ**.

Man könnte das **charakteristische Polynom** berechnen und die Nullstellen mit Hilfe des **Newton-Verfahrens** finden. Das ist aber oft sehr instabil.

So besitzt das Polynom

$$p_\varepsilon(t) = (t - 1) \cdot \dots \cdot (t - 20) - \varepsilon \cdot t^{19}$$

für  $\varepsilon = 0$  die ganzzahligen Nullstellen  $1, \dots, 20$   
und für  $\varepsilon = 2^{-23}$  insgesamt 10 komplexe Nullstellen ( $\Im \in [0.6; 2.0]$ ).

## Lösen des Eigenwertproblems

Möchte man also die Eigenwerte der Diagonalmatrix

$$A = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 20 \end{pmatrix}$$

berechnen, reicht es, diese Informationen **aus der Diagonalen abzulesen**, anstatt das numerisch instabile charakteristische Polynom aufzustellen und zu lösen.

Es ist also sinnvoller, für das Eigenwertproblem direkt mit  $A$  zu arbeiten.

Im Folgenden sei eine Matrix  $A \in \mathbb{R}^{n \times n}$  gegeben und wir sind auf der Suche nach einem Eigenwert  $\lambda \in \mathbb{R}$ . Um dieses Problem eindeutig zu beschreiben, gehen wir davon aus, dass  $\lambda$  eine einfache Nullstelle von  $p_A$  ist und eindeutig beschreibbar ist (z.B. kleinster Eigenwert, größter Eigenwert, 42ter Eigenwert).



**Beschreibung des Eigenwertproblems**

Die **absolute Kondition eines Problems** ist durch die Ableitung bestimmt.

Wir nehmen an, dass die Matrix  $A$  durch eine Matrix  $C \in \mathbb{R}^{n \times n}$  gestört wird:

$$A(t) = A + t \cdot C.$$

Damit erhalten wir eine Eigenwertfunktion  $\lambda(t)$  und eine Eigenvektorfunktion  $x(t)$ , so dass Folgendes gilt:

$$A(t) \cdot x(t) = \lambda(t) \cdot x(t)$$

Um die Notation zu vereinfachen, schreiben wir

$$\lambda_0 := \lambda(0)$$

$$x_0 := x(0)$$

## Partielle Ableitung des EW-Problems

Da  $p_{A^\top} = p_A$ , ist  $\lambda_0$  auch ein Eigenwert der Matrix  $A^\top$ . Es sei nun  $y_0$  ein Eigenvektor von  $A^\top$  zum Eigenwert  $\lambda_0$ .

Damit erhalten wir

$$\begin{aligned} 0 &= \left\langle \left. \frac{d}{dt} A(t)x(t) - \lambda(t)x(t) \right|_{t=0}, y_0 \right\rangle \\ &= \langle Cx_0 + Ax'(0) - \lambda'(0)x_0 - \lambda_0 x'(0), y_0 \rangle \\ &= \langle Cx_0 - \lambda'(0)x_0, y_0 \rangle + \underbrace{\langle Ax'(0) - \lambda_0 x'(0), y_0 \rangle}_{=0} \end{aligned}$$

Insgesamt erhalten wir also

$$\lambda'(0) = \frac{\langle Cx_0, y_0 \rangle}{\langle x_0, y_0 \rangle}.$$

## Absolute Kondition des EW-Problems

Betrachten wir nun das Eigenwertproblem als Funktion  $\lambda: \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ . Dann ist die absolute Kondition gerade:

$$\kappa_{\text{abs}} = \limsup_{C \rightarrow 0} \frac{|\lambda(A + C) - \lambda(A)|}{\|C\|_2} = \limsup_{C \neq 0} \frac{\frac{\langle Cx_0, y_0 \rangle}{\langle x_0, y_0 \rangle}}{\|C\|_2}$$

Wegen

$$|\langle Cx_0, y_0 \rangle| \leq \|Cx_0\| \|y_0\| \leq \|C\|_2 \|x_0\| \|y_0\|$$

erhalten wir (für  $C = y_0 \otimes x_0$  gilt Gleichheit)

$$\kappa_{\text{abs}} = \frac{1}{|\cos(\angle(x_0, y_0))|}.$$

## Symmetrisches Eigenwert-Problem

Ist  $A$  symmetrisch, so ist  $x_0 = y_0$  und wir erhalten für die absolute Kondition

$$\kappa_{\text{abs}} = \frac{1}{\cos(\angle(x_0, x_0))} = 1$$

Für die relative Kondition gilt dann

$$\kappa = \frac{\|A\|_2}{|\lambda_0|}.$$

Für nicht-symmetrische Matrizen kann die Kondition sehr groß werden. Für  $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$  und  $\tilde{A} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix}$  gilt  $\lambda_1 = 1$  und  $\tilde{\lambda}_{1,2} = 1 \pm \sqrt{\varepsilon}$  und somit

$$\kappa_{\text{abs}} \geq \frac{|\pm\sqrt{\varepsilon}|}{\left\| \begin{pmatrix} 0 & 0 \\ \varepsilon & 0 \end{pmatrix} \right\|_2} = \frac{\sqrt{\varepsilon}}{\varepsilon} = \frac{1}{\sqrt{\varepsilon}} \rightarrow \infty \quad \text{für } \varepsilon \rightarrow 0$$

**Vektor-Iteration**

Es sei eine symmetrische Matrix  $A \in \mathbb{R}^{n \times n}$  gegeben.

Weiter besitze  $A$  die Eigenwerte  $0 < |\lambda_1| < \dots < |\lambda_n|$  und die dazu gehörigen normierten Eigenvektoren  $v_1, \dots, v_n \in \mathbb{R}^n$ .

Betrachten wir nun die Folge der **Vektor-Iteration (eng. Power Method)**

$$x_0 \in \mathbb{R}^n, \|x_0\| = 1$$

$$x_{k+1} = \frac{Ax_k}{\|Ax_k\|}$$

Alle Vektoren  $x_k$  lassen sich wie folgt darstellen:

$$x_k = \sum_{i=1}^n \alpha_{k,i} v_i$$

Im Folgenden gehen wir davon aus, dass  $\forall i : \alpha_{0,i} \neq 0$  gilt.

Als Nächstes werden wir uns die Folgen  $\alpha_{k,i}$  ansehen.

Insbesondere sind wir an  $\lim_{k \rightarrow \infty} \alpha_{k,i}$  für jedes  $i = 1, \dots, n$  interessiert.

## Konvergenz (Vektor-Iteration)

Es gilt nun

$$Ax_k = \sum_{i=1}^n \lambda_i \alpha_{k,i} v_i \quad \Rightarrow \quad \alpha_{k+1,i} = \frac{\lambda_i}{\sqrt{\sum_{j=1}^n \lambda_j^2 \alpha_{k,j}^2}} \alpha_{k,i} \neq 0$$

Für  $i < n$  ergibt sich

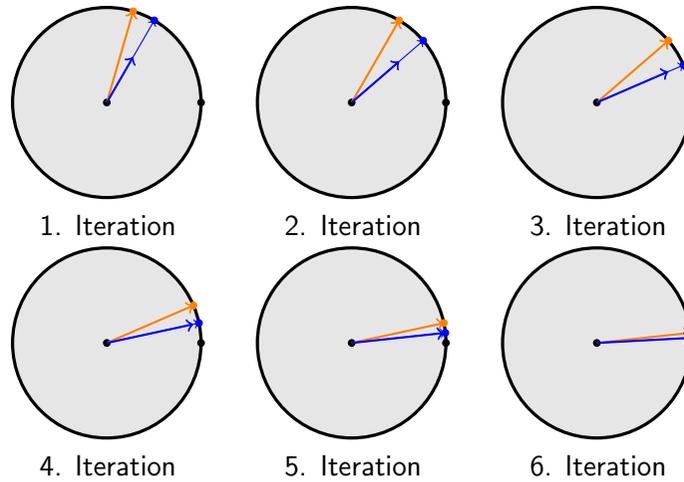
$$\frac{\alpha_{k+1,i}}{\alpha_{k+1,n}} = \frac{\lambda_i}{\underbrace{\lambda_n}_{<1}} \frac{\alpha_{k,i}}{\alpha_{k,n}}$$

Damit erhalten wir insgesamt

$$\begin{aligned} \lim_{k \rightarrow \infty} \alpha_{k,i} &= 0 & i < n \\ \lim_{k \rightarrow \infty} \alpha_{k,n} &= \pm 1 \\ \lim_{k \rightarrow \infty} x_k &= \pm v_n \end{aligned}$$

## Vektor-Iteration (Bsp.)

Betrachten wir die Matrix  $A = \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}$  und den Startvektor  $x_0 = \begin{pmatrix} 0.28 \\ 0.96 \end{pmatrix}$ :



## Berechnung aller Eigenvektoren

Die Vektoriteration konvergiert gegen den **größten Eigenvektor**, falls für den Startwert  $x_0$  Folgendes gilt:

$$x_0 = \sum_{i=1}^n \alpha_{0,i} v_i \quad \forall i : \alpha_{0,i} \neq 0$$

Wenn einige  $\alpha_{0,i} = 0$  sind, konvergiert die Vektoriteration gegen den größten Eigenvektor  $v_i$ , für den  $\alpha_{0,i} \neq 0$  gilt.

Da Eigenvektoren symmetrischer Matrizen **orthogonal** zu einander sind, kann man die Vektoriteration **mehrmals durchführen**, indem man nur solche Vektoren zulässt, die senkrecht auf **bereits gefundenen** Eigenvektoren stehen.

Wähle einen **zufälligen Vektor**  $x_0$ , der senkrecht auf  $v_1, \dots, v_\nu$  steht und führe die Power-Iteration mit  $x_0$  als **Startwert** durch.  
Damit können iterativ **alle** Eigenvektoren gefunden werden.

## Zusammenfassung (Vektor-Iteration)

Die Vektor-Iteration ist einfach zu implementieren, da sie nur **Matrix-Vektor-Multiplikation** und **Vektor-Normierung** benutzt.

Beginnt man mit einem zufälligen Vektor  $x_0$ , konvergiert die Iterationsfolge  $(x_k)_{k \in \mathbb{N}}$  zu einem Eigenvektor.

In den meisten Fällen ist dieser Eigenvektor  $v_i$  ein Eigenvektor zum größten Eigenwert  $\lambda_n$ .

Ist die Matrix  $A$  symmetrisch sind die Eigenvektoren orthogonal und die Vektor-Iteration kann benutzt werden, um nacheinander alle Eigenvektoren zu berechnen.

Dieses Verfahren kann numerisch instabil werden, da Differenzen benutzt werden, die zu Auslöschungen führen können.

## Inverse Vektor-Iteration

Wenn man nur an einem Eigenwert  $\lambda$  interessiert ist, für den man bereits eine gute Schätzung  $\tilde{\lambda}$  besitzt, macht es wenig Sinn, die Vektor-Iteration für alle Eigenvektoren durchzuführen.

Stattdessen, kann man das Problem so umformulieren, dass der gesuchte Eigenvektor  $v_i$  der größte Eigenvektor einer anderen Matrix ist.

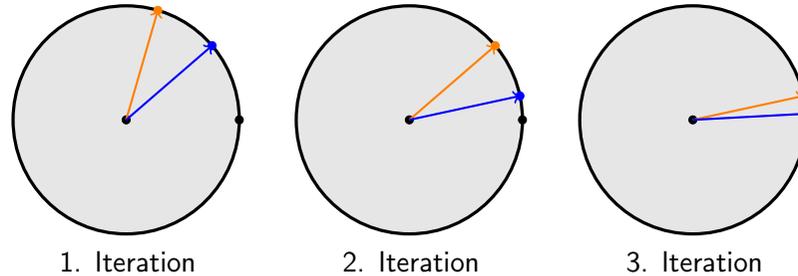
Dies wird von der **inversen Vektor-Iteration (eng. Inverse Power Method)** erreicht. Die Vektor-Iteration wird auf folgende Matrix angewandt

$$\tilde{A} := (A - \tilde{\lambda}I)^{-1}$$

Die Eigenwerte von  $\tilde{A}$  sind  $\tilde{\lambda}_i := (\lambda_i - \tilde{\lambda})^{-1}$ , wobei der größte Eigenwert gerade  $|\lambda - \tilde{\lambda}|^{-1}$  ist, d.h. man kann die inverse Vektor-Iteration benutzen, um einen vorher ausgewählten Eigenwert zu berechnen.

### Inverse Vektor-Iteration (Bsp.)

Betrachten wir  $A = \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}$ ,  $\tilde{\lambda} = 0.9$  und somit  $\tilde{A} = \begin{pmatrix} 10 & 0 \\ 0 & 2.5 \end{pmatrix}$ .  
Dann erhalten wir mit dem Startvektor  $x_0 = \begin{pmatrix} 0.28 \\ 0.96 \end{pmatrix}$ :



Die inverse Vektor-Iteration konvergiert schneller als die Vektor-Iteration, wenn ein  $\tilde{\lambda}$  eine gute Schätzung von  $\lambda$  ist.  
Allerdings muss die Matrix  $\tilde{A}$  invertiert werden.

**Berechnung aller Eigenwerte**

Die Vektoriterationen sind sehr hilfreich, wenn man nur an einem Eigenwert bzw. Eigenvektor interessiert ist. Wenn man aber alle Eigenwerte berechnen möchte, kann eine sukzessive Vektoriteration numerisch sehr instabil werden.

Es reicht daher eine orthogonale Matrix  $Q \in \mathbb{R}^{n \times n}$  zu finden, so dass:

$$A = Q\Lambda Q^T$$

Dies kann erreicht werden, indem man eine Folge  $(A_n)_{n \in \mathbb{N}}$  von Matrizen findet, so dass sie gegen eine Diagonalmatrix konvergiert und außerdem Folgendes gilt

$$A_{n+1} = Q_n^T A_n Q_n$$

Dies wird durch das sogenannte **QR-Verfahren** erreicht.

## QR-Verfahren

Das QR-Verfahren, berechnet in jeder Iteration die QR-Zerlegung einer Matrix  $A_n$  und multipliziert diese beiden Matrizen in umgekehrter Reihenfolge.

Das QR-Verfahren, erzeugt also die folgenden drei Matrizenfolgen

$$\begin{aligned} A_0 &:= A & A_0 & =: Q_0 \cdot R_0 \\ A_{n+1} &:= R_n \cdot Q_n & A_{n+1} & =: Q_{n+1} \cdot R_{n+1} \end{aligned}$$

Es gilt nun

$$A_{n+1} = R_n \cdot Q_n = Q_n^\top Q_n R_n \cdot Q_n = Q_n^\top A_n Q_n,$$

d.h.  $A_{n+1}$  hat die gleichen Eigenwerte wie  $A_n$  und damit (vollständige Induktion) ebenfalls die gleichen Eigenwerte wie  $A = A_0$ .

Das QR-Verfahren verwandelt also das symmetrische Eigenwertproblem in jedem Schritt in ein äquivalentes Eigenwertproblem.

## Eigenschaften des QR-Verfahrens

Die Analyse des QR-Verfahrens ist nicht ganz einfach. Allerdings kann man folgende Eigenschaften für den symmetrischen Startwert  $A_0 = A$  beweisen:

$$\begin{aligned} \lim_{n \rightarrow \infty} Q_n &= I \\ \lim_{n \rightarrow \infty} R_n &= \Lambda \end{aligned}$$

Definiert man außerdem  $U_n = \prod_{i=0}^n Q_i$ , so kann man zeigen, dass Folgendes gilt:

$$A = U^\top \Lambda U$$



## Zusammenfassung

Wir haben zwei iterative Verfahren kennen gelernt, um das symmetrische Eigenwertproblem zu lösen, die **Vektor-Iteration** und das **QR-Verfahren**.

Jede Iteration der Vektor-Iteration benötigt  $\mathcal{O}(n^2)$  Schritte, während jede Iteration des QR-Verfahrens  $\mathcal{O}(n^3)$  Schritte benötigt.

Wenn man nur einen Eigenvektor berechnen möchte, bietet sich die Vektor-Iteration an. Ist man an allen Eigenvektoren interessiert, ist das QR-Verfahren zu bevorzugen.

Das QR-Verfahren konvergiert auch, wenn ein Eigenwert mehrfach vorkommt, d.h. anstelle der Forderung

$$0 < |\lambda_1| < \dots < |\lambda_n|$$

wird lediglich Folgendes gefordert

$$0 < |\lambda_1| \leq \dots \leq |\lambda_n|$$

**Approximation von Eigenwerten**

Auch wenn wir mit dem QR-Verfahren eine Folge von Matrizen  $A_n$  generiert haben, die gegen die Diagonalmatrix  $\Lambda$  konvergiert, wird  $\Lambda$  nie exakt erreicht. Die Frage ist dann wie weit sich die Diagonaleinträge einer Matrix von ihren Eigenwerten unterscheidet.

Es sei also eine beliebige Matrix  $A \in \mathbb{R}^{n \times n}$  gegeben. Dann definieren wir mit

$$d_i := a_{ii}$$

die Diagonaleinträge der Matrix  $A$ .

Weiter beschreiben wir mit

$$\rho_i := \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|$$

wie weit sich die  $i$ -te Zeile von  $A$  von einer Diagonalmatrix unterscheidet.

## Eigenwerte und Diagonaleinträge

Sei nun  $v$  ein Eigenvektor von  $A$  zum Eigenwert  $\lambda$ .

Ohne Beschränkung der Allgemeinheit können wir Folgendes annehmen:

$$\max_{j=1,\dots,n} |v_j| = v_i = 1$$

Dann gilt

$$\lambda = \lambda v_i = (\lambda v)_i = (Av)_i = \sum_{j=1}^n a_{ij} v_j = a_{ii} + \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} v_j$$

Und somit

$$|\lambda - d_i| \leq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| = \rho_i$$

## Gerschgorin-Kreise

Wir haben also gesehen, dass es zu jedem **Eigenwert**  $\lambda$  eine Zeile  $i$  von  $A$  gibt, so dass sich  $\lambda$  von dem **Diagonaleintrag**  $d_i$  nicht mehr als den **Radius**  $\rho_i$  unterscheidet.

Damit haben wir das **Kreistheorem von Gerschgorin** bewiesen:

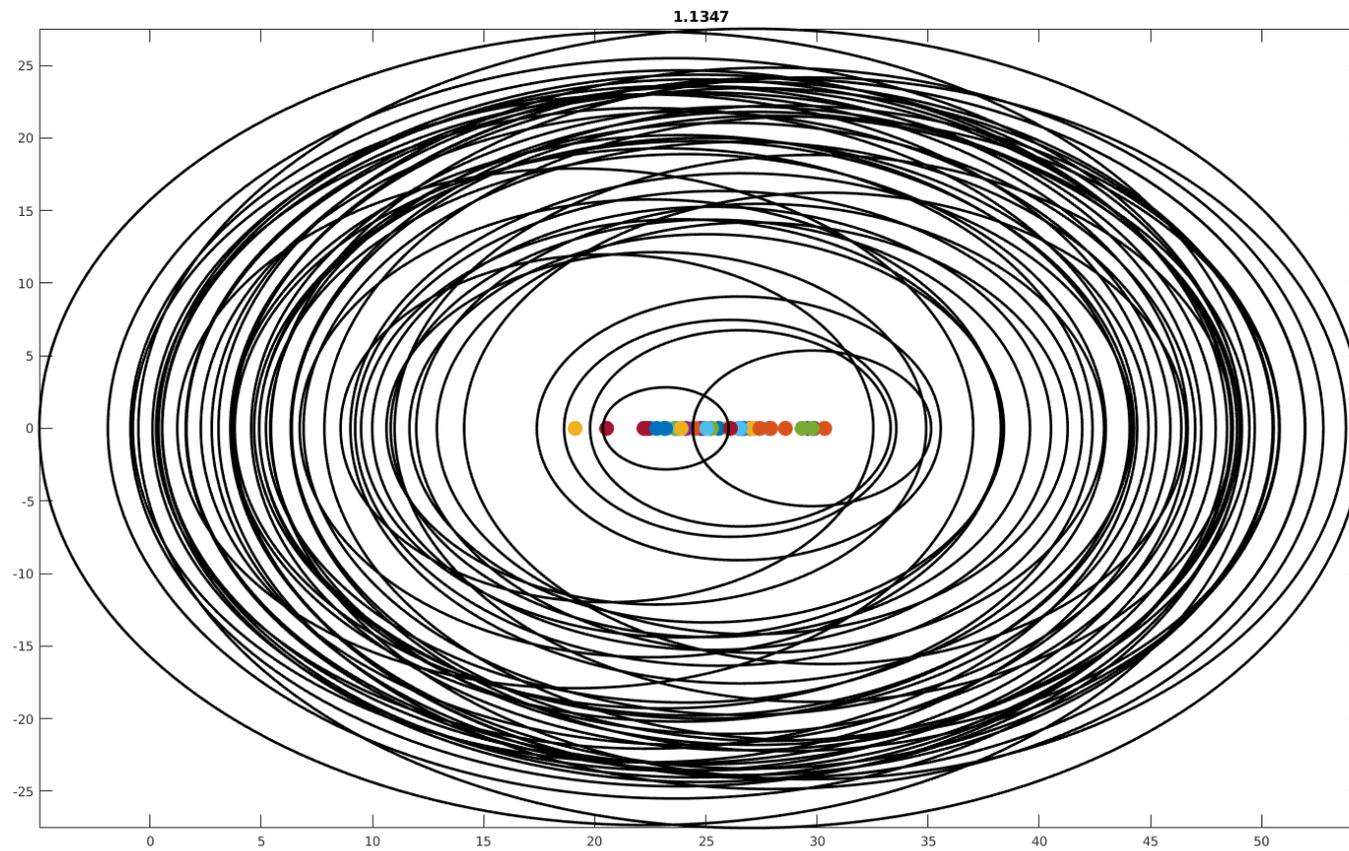
**Theorem 1** (Gerschgorin). Sei  $A \in \mathbb{C}^{n \times n}$  und  $d_i$  sowie  $\rho_i$  wie oben definiert. Dann gilt für die Eigenwerte  $\lambda_i$  von  $A$  stets:

$$\{\lambda_1, \dots, \lambda_n\} \subset \bigcup_{i=1}^n B_{\rho_i}(d_i)$$

Das Kreistheorem hilft uns bei der Analyse des QR-Verfahrens.

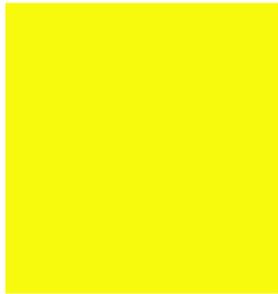
Die Diagonaleinträge von  $A_n$  approximieren die Eigenwerte und die Nebendiagonaleinträge geben die Approximationsgüte an.

# Gerschgorin-Kreise

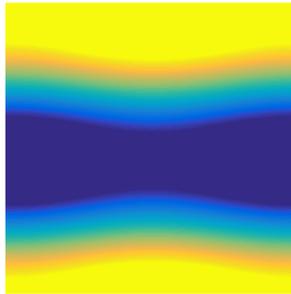




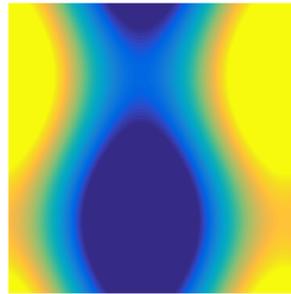
“Frequenzen” im  $\mathbb{R}^2$



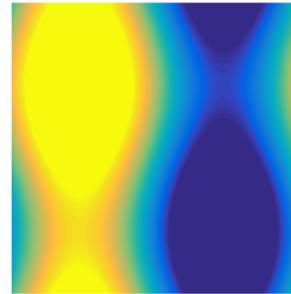
$\lambda = 0$



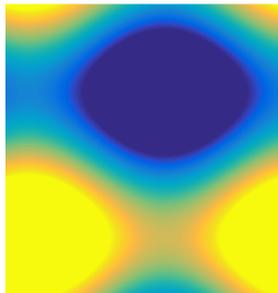
$\lambda = -1$



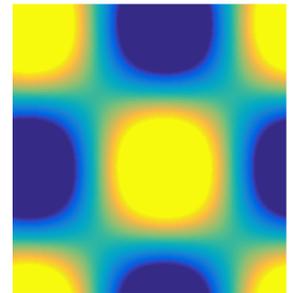
$\lambda = -1$



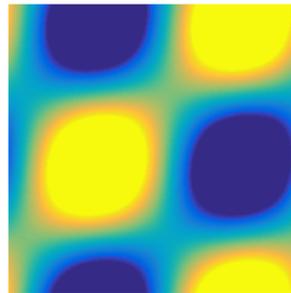
$\lambda = -1$



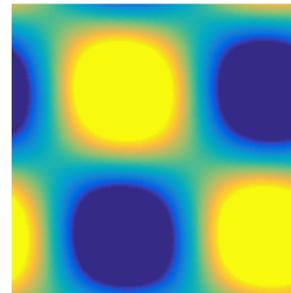
$\lambda = -1$



$\lambda = -2$



$\lambda = -2$



$\lambda = -2$

“Frequenzen” einer Katze

